2017-12-01

A Comparative Literature Review of Intelligent Tutoring Systems from 1990-2015

Brice Robert Colby
Brigham Young University

A Comparative Literature Review of Intelligent Tutoring Systems from 1990-2015

Brice Robert Colby

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Richard West, Chair
Peter J. Rich
Stephen Yanchar

Department of Instructional Psychology and Technology

Brigham Young University

ABSTRACT

A Comparative Literature Review of Intelligent Tutoring Systems from 1990-2015

Brice Robert Colby
Department of Instructional Psychology and Technology, BYU
Master of Science

This paper sought to accomplish three goals. First, it provided a systematic, comparative review of several intelligent tutoring systems (ITS). Second, it summarized problems and solutions presented and solved by developers of ITS by consolidating the knowledge of the field into a single review. Third, it provided a unified language from which ITS can be reviewed and understood in the same context. The findings of this review centered on the 5-Component Framework. The first component, the domain model, showed that most ITS are focused on science, technology, and mathematics. Within these fields, ITS generally have mastery learning as the desired level of understanding. The second component, the tutor model, showed that constructivism is the theoretical strategy that informs most ITS. The tutoring tactics employed in the ITS stem from this paradigm. The third component, the student model, describes the several ways ITS infer what a student knows. It described the variety of data that is collected by an ITS and how it is used to build the student model. The fourth component, the interface, revealed that most ITS are now web-based, but vary in their capacity to interact with students. It also showed that user experience is underreported and ought to be included more in the research. Finally, the fifth component, learning gains, demonstrated that ITS are capable of producing learning gains equivalent to a human tutor. However, reporting learning gains does not seem to be a focus of the literature.

Keywords: intelligent tutoring systems, literature review

TABLE OF CONTENTS

ABSTRACT ii
TABLE OF CONTENTS iii
LIST OF TABLES vii
LIST OF FIGURES viii
DESCRIPTION OF THESIS STRUCTURE ix
JOURNAL ARTICLE 1
A Comparative Literature Review of Intelligent Tutoring Systems from 1990-2015 1
Background 3
First Generation (1970-1990) 4
Second Generation (1990-2010) 5
Third Generation (2010-Present) 5
5-Component Framework Review 6
Domain Model 7
Tutoring Model 7
Student Model 8
Interface 8
Learning Gains 8
Review Methods 9
Searching for and Identifying ITS Cases 9
Findings 11
The Domain Model 11
The Tutoring Model 19
The Student Model 29
The Interface 36
Learning Gains 40
Conclusion 45
References 48
ITS CASE STUDIES 61
ActiveMath 61
Domain Model 61
Tutoring Model 62

Student Model 65
Interface 67
Learning Gains 68
References 69
ANDES 72
Domain Model 72
Tutoring Model 73
Student Model 77
Interface 78
Learning Gains 80
References 81
Aplusix 83
Domain Model 83
Tutoring Model 83
Student Model 85
Interface 87
Learning Gains 88
References 90
ASSISTments 92
Domain Model 92
Tutoring Model 93
Student Model 95
Interface 96
Learning Gains 98
References 99
AutoTutor 102
Domain Model 102
Tutoring Model 103
Student Model 107
Interface 109
Learning Gains 110
References 112

CIRCSIM-Tutor 115
Domain Model 115
Tutoring Model 115
Student Model 118
Interface 119
Learning Gains 121
References 122
Cognitive Tutor 124
Domain Model 124
Tutoring Model 125
Student Model 129
Interface 130
Learning Gains 131
References 133
Reading Tutor 136
Domain Model 136
Tutoring Model 137
Student Model 139
Interface 141
Learning Gains 141
References 142
SQL-Tutor 145
Domain Model 145
Tutoring Model 146
Student Model 148
Interface 150
Learning Gains 151
References 153
Wayang Outpost 156
Domain Model 156
Tutoring Model 157
Student Model 160

Interface 163
Learning Gains 164
References 166

LIST OF TABLES

Table 1. *List of ITS, Number of Articles from the First Phase and the Date Range*……13

Table 2. *A List of STEM Domains for Each ITS*……14

LIST OF FIGURES

Literature Review

Figure 1. Reported Average and ranges of effect sizes for the 10 ITS. 42

ActiveMath

Figure 1. The interface of ActiveMath (Andres et al., 2010). 67

ANDES

Figure 1. The interface of the web-based version of ANDES (VanLehn et al., 2010). 79

Aplusix

Figure 1. The user interface of Aplusix (Andallaza et al., 2012). 88

ASSISTments

Figure 1. A problem in ASSISTments including scaffolding and hints (Pedra et al., 2013). ... 97

AutoTutor

Figure 1. The interface of AutoTutor (D'Mello & Graesser, 2012). 110

CIRCSIM-Tutor

Figure 1. The user interface of CIRCSIM-Tutor (Michael et al., 2003). 120

Cognitive Tutor

Figure 1. The interface of Algebra Cognitive Tutor (Rolle et al., 2011). 131

Reading Tutor

Figure 1. The interface of Reading Tutor (Aist, Kort, Reilly, Mostow, & Picard, 2002). 141

SQL-Tutor

Figure 1. The SQL-Tutor interface (Hull & du Boulay, 2015). 151

Wayang Outpost

Figure 1. The user interface of Wayang Outpost (Arroyo et al., 2012). 164

DESCRIPTION OF THESIS STRUCTURE

This thesis was intended to be written in a hybrid format. The first part of the thesis is a journal-ready article that is a comprehensive literature review of intelligent tutoring systems from 1990 to 2015 and serves as the research article. The information for the review came from the second part of the thesis, which is a collection of intelligent tutoring system case studies. Each case is to be a stand-alone resource that will be provided on the Web. A 5-Component Framework that covers the domain content, student knowledge, tutoring pedagogy, interface, and learning gains was used to analyze both sections.

For publication of the journal-ready portion of my thesis, I would like to submit to the *Review of Educational Research* (RER). This journal focuses on publishing literature reviews about education from any discipline. As such, the article needs to be written at a level that is understandable by a broad audience. Publications need to follow APA formatting. There is no maximum word length.

JOURNAL ARTICLE

A Comparative Literature Review of Intelligent Tutoring Systems from 1990-2015

Intelligent Tutoring Systems (ITS) are computer programs "designed to incorporate techniques from the artificial intelligence (AI) community in order to provide tutors which know *what* they teach, *who* they teach, and *how* to teach it," (Nwana, 1990, p. 252). With such a loose definition, anything demonstrating some semblance of artificial intelligence that is used in education could be considered an intelligent tutoring system. For example, one tutoring system, ActiveMath, focuses on dynamically creating e-textbooks that are adapted to a student's skill level (Ullrich & Libbrecht, 2008). Another tutor, ANDES, does not adapt content at all but, instead, focuses on promoting transfer by designing a user experience that is like a paper-and-pencil format (VanLehn, van de Sande, Shelby, & Gershman, 2010). This loose definition makes it difficult to compare multiple ITS since their underlying architecture can differ greatly from one system to another. Consequently, it has been difficult to aggregate information about ITS systems and build upon previous knowledge. This has led to three significant challenges.

The first main challenge to fully understanding the research on ITS is that no systematic review has been done to compare ITS for many years. The last one that could be found was by Nwana (1990); however, in my search of over 800 articles relating to ITS, I found a few review articles that compared systems to each other or compared different student modeling techniques but no comprehensive reviews similar to Nwana's work.

Second, the field seems to be dominated by a core group of researchers who work on their own tutoring systems. One study looked at 815 papers from the years 2009-2012 and found that one-third of these papers were written by 12 groups (Nye, Graesser, & Hu, 2014). These papers seem to focus on the iterative development of their own tutoring systems which can

hinder the advancement of generalizable knowledge in the field since solutions are found that are context-dependent. Consider some of the following examples that highlight this hindrance.

ITS developers seem to be solving the same problems repeatedly and concurrently. For example, Reading Tutor is an oral reading fluency tutor (Lalle, Mostow, Luengo, & Guin, 2013). Initially, the collected data was not used to create a model of what a student knows (Beck, Jia, & Mostow, 2003; Beck, Jia, Sison, & Mostow 2003). Recognizing the utility in have a strong student model, the developers retrofitted data to infer student capabilities (Beck, et al., 2003). Eventually, they had to redesign their data collection system to have the capacity to collect the data they needed (Beck et. al., 2003). This oversight could have been avoided with an understanding of the role of a student model in ITS, derived from generalizable knowledge and theory developed from more rigorous research.

In contrast, some ITS recognize problems that should be considered by the field but are not. As an example, the developers of ASSISTments identified a concept called wheel-spinning (Beck & Gong, 2013). This concept describes the process of a student getting stuck in a mastery loop. In other words, the tutoring system determines that the student does not know a knowledge component well enough and offers remedial instruction and further problems to be solved. However, the student continues to get the problems wrong and is stuck in this loop until they become frustrated and quit.

Third, there doesn't appear to be a unifying language that allows researchers to talk about ITS in the same manner. One does not need to look far into the research to find similar concepts called by different names. For example, ontologies (Melis, Goguadze, Libbrecht, & Ullrich, 2009), Q-matrices (Birenbaum, Kelly, & Tatsuoka, 1993), and transfer models (Feng & Heffernan, 2006) are all names describing a similar process of mapping knowledge components

to problems which students solve. This may lead to researchers studying knowledge mapping processes by using only "ontology" as the search word and missing research that describes the same process under a different name.

This paper aims to provide a solution to these three problems by providing a systematic, comparative review that uses a standard language to draw comparisons and to consolidate the progress of ITS since the previous attempt by Nwana (1990). To do so, the paper is separated into the following sections. First, context is given by providing a brief overview of the first, second, and third generations of ITS—a distinction made by Nkambou, Mizoguchi, and Bourdeau (2010). Second, the methods section introduces the 5-Component Framework process by which the literature search was conducted, including inclusionary and exclusionary criteria. This 5-Component Framework is used in the paper to provide a unifying language. Third, the findings are reported for each of the five components of the framework.

Background

When talking about technological innovations, it is difficult to keep abreast of all the latest happenings since technological breakthroughs happen frequently. While ITS can benefit from these technological breakthroughs, they rely on advances not only in technology, but also fields like psychology and artificial intelligence. As such, ITS can be reviewed on a larger timescale than other technological innovations. One way they can be broken up is into generations (Nkambou et al., 2010). Nkambou et al. (2010) identified three generations of ITS. The first generational distinction made by Nkambou et al. (2010) described 20 years of research capped with seminal articles that related the state of the art. In similar fashion, the second generation is distinguished as 20 years of research, followed by the third generation which is ongoing.

First Generation (1970-1990)

Nwana (1990) **r**eviewed the first generation of artificial intelligence in education (AIED) and identified the steps that led to the realization of the first ITS. The precursor of ITS began with the development of computer-assisted instruction (CAI) in the 1950s. Computer-assisted instruction programs followed a linear approach, moving students step-by-step to the learning goal or desired behavior. These initial attempts at CAI did not afford any individualization of the instruction. The same material was presented in the same sequence while ignoring student responses and providing no feedback (Nwana, 1990).

These machines advanced to branching programs in the 1960s. This new type of CAI took into consideration student responses and provided a unique learning tree for each student. As students successfully or unsuccessfully answered questions, the sequence of material changed to provide tailored instruction. Further functionality was added to CAI in the form of generative CAI in the late 1960s. The new CAI was able to create and solve its own material (Uhr, 1969; Woods & Hartley, 1971). This progression of CAI from linear programs to generative became the precursor for ITS.

Leading researchers recognized the potential that CAI could bring to education but acknowledged that it was still lacking something—intelligence (Nwana, 1990). The demonstrable superiority of 1:1 human tutors was a well-known fact (Bloom, 1984). The 2-sigma tutoring effect (i.e., the ability of human tutors to produce learning gains of 2 sigma) identified by Bloom (1984) became the gold standard as the researchers searched for ways to use CAI to provide that 1:1 experience. Carbonell (1970) proposed putting the AI in CAI to meet this need. His work, called SCHOLAR, was the beginning of ITS in 1970. From 1970-1990,

Nwana (1990) identified 43 first-generation ITS that were developed. These first-generation ITS created a scientifically-based foundation for the second generation.

Second Generation (1990-2010)

A majority of the ITS identified by the current review come from the second generation. As such, a review of the development of ITS will not be given here, but it is supplied in the findings section of this paper. Instead, this section addresses Self's (1990) argument for the development of scientific foundations for AIED, including its own theories, techniques, and methods. Self also wanted theorizers to put into practice the ideas they had about ITS.

In answer to Self's call, biannual conferences and journals were created for the sole purpose of ITS development and research. Additionally, throughout the second generation, many ITS were built—another answer to Self's call. However, these ITS rarely used the same architecture, which made it difficult to have a unified language. Regardless, common components could still be found between them. These components are: the domain, tutor, and student models as well as the interface used to interact with the tutoring system. Together, these components are called the four-component architecture (Wenger, 1987).

Third Generation (2010-Present)

As ITS and other technologies become more commonplace in classrooms, Nye (2015) suggested that ITS will become less of comprehensive systems and more of an "ecosystem of reusable infrastructure and platforms" (p. 63). This suggests that ITS will become a shell that draws from services provided by other sources. Nye provided an example where other online technologies operate under that philosophy: basic blog site uses authentication services from several different sources (e.g., Google, Facebook).

This review corroborates this notion with the identification of some ITS that are domain-independent. This means that these ITS are designed such that they are flexible enough to create an intelligent tutoring system based on the needs of whoever is creating content. However, it is still too early to tell if ITS and AIED will make the shift to a service-based ecosystem even though Nye (2015) claimed it is an "existential necessity" (p. 64).

Even if ITS make the shift into a service-based ecosystem, thus obsoleting the need for a standardized architecture, much work has been done in the second and third generations that requires a consolidation of knowledge. By consolidating knowledge, gaps can be identified so that developers of ITS know which services ought to be provided and which are redundant. For example, the developers of Cognitive Tutor created a stand-alone Help Tutor which can provide metacognitive feedback to students and can be used in various systems (Roll, Aleven, McLaren, & Koedinger, 2011). Knowing that such a service already exists can help developers know that they should spend time on creating another service that fills a need like automatic affect detection.

5-Component Framework Review

As mentioned in our summary of the second generation of intelligent tutoring systems, Wenger's four component architecture (1987) was created to describe a typical architecture for ITS. This architecture mentions domain, tutoring, and student models, as well as the user interface of an ITS. The domain model answers the question of what ITS teach, the student model—who they teach, and the tutoring model—how they teach. The interface component describes what the human-computer interaction is like and whether users like the interface. Since these four components can typically be identified in most ITS (albeit sometimes under other names), this architecture is used as the unifying language for this paper. However, it has

been augmented with a fifth component: learning gains. This component can answer the question of whether the tutoring system is effective.

Each of these five components, we call the 5-Component Framework, identifies an area of particular concern when constructing ITS. Attention to each area is what allows ITS to properly teach the domain to a student, act as a tutor, infer what a student knows, create an enjoyable experience for users, and ultimately let the users know that the system is worth their time by improving learning. For clarification, each component will be described below.

Domain Model

The domain model refers to expert knowledge. It contains the concepts, rules, and problem-solving strategies of the domain (Nkambou et al., 2010). Building the domain model is what allows the tutor to be the teacher and is how the ITS know "*what* they teach**"** (Nwana, 1990, p. 252). Implicit to this model is the understanding that different types of knowledge require different types of teaching depending on the nature of the knowledge to be learned.

Tutoring Model

Whereas the domain model represents the expert knowledge of a teacher, the tutoring model represents the pedagogical knowledge of a tutor. This model will inform tutoring strategies and tactics and dictates "*how* to teach" (Nwana, 1990, p. 252). Like a human teacher, ITS are influenced by pedagogical paradigms like constructivism or behaviorism. Examples of pedagogical tactics employed by ITS include Socratic dialogue, scaffolding, and hint-giving. Careful attention is given to the development of the tutoring model to ensure that the ITS is built upon sound pedagogy and produces the desired learning gains that come from tutor interactions.

Student Model

The student model has been called the "core component" (Nkambou, et al., 2010). It is designed to contain as much information about a student as possible (to the extent that the tutor can make use of the data) and to record changes in this information. This is what allows ITS to know "*who* they teach" (Nwana, 1990). These changes include performance factors such as answers to questions or time to completion of lessons and problems. Additional factors include cognitive and affective states or even changes in belief patterns, knowledge, and misconceptions. Collecting this data about each student allows the ITS to proceed teaching in an informed manner and puts the student at the center of teaching.

Interface

The interface describes the interaction between the ITS and the student and includes many components from graphical representations of the tutor to different representations of the domain content to the amount of control afforded to the student. ITS vary greatly in this regard as some ITS are minimalistic in terms of the complexity of the human-computer interaction while others develop virtual tutors capable of expressing emotions like approval or disapproval.

Learning Gains

At their heart, ITS are meant to improve instruction and learning through adapting to learner needs. Therefore, any intelligent tutoring system framework that does not consider ITS' effect on learning would be incomplete. Thus, we added learning gains to the four-component architecture to gather information on what seems to be working in ITS development, and whether ITS are getting closer to achieving the 2-sigma effect of human tutors (Bloom, 1984). Inattention to learning gains can result in ineffective pedagogical techniques or gimmicky

interfaces being implemented rather than a well-honed ITS that eschews excess and provides a pointed learning experience.

Review Methods

In order to understand the state of ITS systems currently, as well as areas for potential development, this review addresses the problem of creating a unifying language to discuss ITS. In order to do so, first, we define how various ITS address each of the five components in the 5-Component Framework. Second, patterns across ITS are identified that demonstrate how these components are implemented in practice and to find areas of strength and areas that require further research and development. More specifically, the following questions are answered for each component:

1. **Domain model.** What content do ITS cover? What level of understanding is the goal? What is used to create the expert?
2. **Tutoring model.** What tutoring strategies and tactics are used?
3. **Student model.** How is the student model created? What issues are there regarding the development of the student model? What data is collected?
4. **Interface.** What are the common types of interfaces? What research is there on ITS user experience?
5. **Learning gains.** What learning gains are reported? Does the research suggest that ITS are close to achieving the 2-sigma effect of human tutors?

Searching for and Identifying ITS Cases

An initial, informal search was conducted to generate a list of ITS. A formal search was then conducted by looking through literature databases using the names of the tutoring systems as the search keyword. At times, due to the general nature of some ITS' names, the words

"intelligent tutoring system" were added at the end to hone the results. When a new system was mentioned in an article, it was also added to the list and used for its own literature search.

Since the field of AIED can span several educational databases (e.g., computer science, psychology, business), Google Scholar was a more effective database than any individual, discipline-based database.

Inclusionary/exclusionary criteria. Including and excluding articles involved two phases. First, we included results from 1990-2015, which covered the second and third generations of ITS. Only relevant articles were included, which was determined by reading through the abstract and searching the article to make sure that the system in question was the focus of the article and not a quick reference. Article collection for each ITS stopped after five pages of results on Google Scholar since the quality and relevance of the articles started to diminish at that point (i.e., articles were merely referencing the named system or results were articles not even relating to ITS). The first phase yielded a reasonably comprehensive list of ITS including 64 unique systems with a total of 823 articles for all ITS.

The second phase narrowed the number of articles/ITS based on an approximation of prominence in the field. To measure prominence, first, the system must have more than 25 articles collected over a 10+ years span. This produced a list of 10 ITS with a total of 333 articles combined. This represented 15.6% of the total ITS, but 40.5% of the published articles. This suggests that most of the research is dominated by these systems and supports the conclusion that these are influential systems.

Each article was then read and reviewed and all information pertaining to one of the five components was identified. Articles were read in chronological order starting with the oldest. This sequencing allowed the iterative development of each system to be more apparent as the

research built on older publications. If an article provided redundant information, it was excluded from the list. The 10 ITS described in more detail in the next section are ActiveMath, ANDES, Aplusix, ASSISTments, AutoTutor, CIRCSIM-Tutor, Cognitive Tutor, Reading Tutor, SQL-Tutor, and Wayang Outpost. Table 1 shows the full list of ITS, the number of articles found in the first phase, and the date range of the articles. Note that some ITS were referenced and are included in this list, but no articles were found using Google Scholar.

Findings

As was mentioned, of the ITS identified in Table 1, 10 will be used for further exploration in the following sections. These 10 systems offer a good coverage of different domains, modeling techniques, and other features of ITS. The following subsections will address the research questions presented earlier by using the 10 systems as examples.

The Domain Model

What content do ITS cover? To answer this question, each ITS was reviewed to see what subject matter the tutoring system focused on. This resulted in 31 unique domains such as thermodynamics, physics, and algebra. However, each of these domains could be classified under broader domains, namely STEM. In order to simplify the categories, the STEM domains (Science, Technology, Engineering, Mathematics) were used as classifiers with an "Other" domain for subject matter that did not fit in STEM. A final category of "Domain Independent" was used for those ITS which were created with the intent of being able to handle any domain content. The results are presented in Table 2.

This categorization shows a preference for the science, technology and mathematics domains. Science had 14 ITS that covered four subdomains (i.e., biology, thermodynamics, physics, and genetics). Of these subdomains, physics had the highest number of ITS with seven

dedicated solely to physics tutoring. This was also the highest number of ITS for any one subdomain that related to a specific knowledge domain. Technology had 12 ITS that covered nine subdomains (i.e., Java, Haskell, SQL, UML, database design, data normalization, functional dependencies, linked lists, and computer literacy). Of these subdomains, five ITS were developed to tutor Java while the remaining subdomains had one system each. Mathematics had 11 ITS that could be categorized in this domain. The subdomains included here are general mathematics, scatterplots, modelling dynamic systems, number factorization, geometry, algebra, statistics, and calculus. Algebra had the most ITS of this domain with six systems.

The remaining three domains, engineering, other, and domain independent are noteworthy. Engineering had a total of three ITS for two subdomains (i.e., electrical engineering and electricity and electronics) which made it the least represented STEM domain. Other had 10 ITS for seven subdomains (i.e., design, plantation decision making negotiated interviews for nursing, logic, reading, critical thinking, and cross-cultural negotiation). The domain-independent category had eight ITS.

The spread of the ITS over these domains seems to indicate a preference for designing ITS that tutor well-defined domains. These well-defined domains can be considered as domains where the relationships between knowledge, rules, and problem-solving strategies are more apparent and each can be easily formalized. First, the clear understanding of relationships among the domain's knowledge components makes it easier to develop the tutoring model since prerequisite knowledge can be identified and targeted for instruction. Second, the ease of formalization benefits the student model since knowledge components are explicitly defined and can be associated with homework problems in the tutoring system. This allows the ITS to determine what the student is learning or demonstrating as the student interacts with the system.

Table 1

List of ITS, Number of Articles from the First Phase, and the Date Range

ITS	# of Articles	Date Range	ITS	# of Articles	Date Range	ITS	# of Articles	Date Range
3ITS	1	2016	DeepTutor	19	2006-2016	NORMIT	13	2002-2013
ActiveMath	32	2001-2011	DM-Tutor	4	2009-2012	OOPS	1	2009
Algebra Tutor	15	1996-2015	Dr Vicky	1	2011	PAT	1	2012
Andes	30	1996-2014	Dragoon	5	2014-2016	Prime Climb	24	2002-2015
AnimalWatch	23	1999-2015	EcoLab II	15	2002-2016	Reading Tutor	33	1999-2013
Aplusix	31	1993-2014	EER-Tutor	1	2006	Protus	13	2011-2015
Ask-Elle	8	2012-2016	Electronix Tutor	1	2016	REALP	0	--
ASSISTments	36	2005-2015	eTeacher	5	2004-2013	Scatterplot Tutor	10	2005-2015
Atlas	0	--	FDTutor	1	2008	Scooter the Tutor	8	2006-2016
AutoTutor	35	1999-2014	FFDC	2	2009	SE-Coach	13	1997-2009
BEETLE II	21	2010-2016	iList	8	2008-2015	SHIECC	0	--
BILAT	23	2006-2013	ILMDA	5	2004-2010	SmartTutor	3	2001-2002
BlueJ	0	--	iStart	28	2004-2013	SQL-Tutor	38	1996-2016
CIMEL ITS	8	2005-2008	Java Sensei	4	2015	T-Algebra	14	2005-2011
CIRCSIM-Tutor	34	1991-2011	JavaTutor	25	2009-2016	The Incredible Machine	9	2001-2014
Cognitive Constructor	7	2008-2015	J-LATTE	2	2009-2016	Thermo-Tutor	1	2011
COLLECT-UML	0	--	KERMIT	20	2001-2012	TutorJ	10	2005-2013
Cognitive Tutor	38	1998-2014	Logiocando	5	2003-2016	Wayang Outpost	32	2003-2014
Conceptual Helper	3	2000-2008	Mag	2	2009-2011	Why2-Atlas	11	2002-2006
Cordillera	14	2007-2015	Mathematics Tutor	0	--	ZOSMAT	3	2009-2015
Crystal Island	32	2006-2012	Medical Imaging	1	2009			
Deep Thought	25	2004-2015	MetaTutor	32	2007-2016			

Table 2

A List of STEM Domains for Each ITS

ITS	S	T	E	M	Other	Domain Independent
3ITS						X
ActiveMath				X		
Algebra Tutor				X		
ANDES	X					
AnimalWatch				X		
Aplusix				X		
Ask-Elle		X				
ASSISTments						X
AutoTutor	X	X			X	
Beetle II			X			
BILAT					X	
BlueJ		X				
CIMEL ITS		X				
CIRCSIM-Tutor	X					
Cognitive Constructor						X
Cognitive Tutor	X			X		
Conceptual Helper	X					
Cordillera	X					
Crystal Island	X					
Deep Thought					X	
DeepTutor	X					
DM-Tutor					X	
Dr Vicky					X	
Dragoon				X		
EcoLab II	X					
Electronix Tutor			X			
eTeacher						X
FDTutor		X				
FFDC					X	
iList		X				
ILMDA						X
iStart					X	
Java Sensei		X				
JavaTutor		X				
J-LATTE		X				
KERMIT		X				
Logiocando					X	
MetaTutor	X					
NORMIT		X				
Prime Climb				X		
Project Listen					X	
Protus	X					
REALP						
Scatterplot Tutor				X		
SE-Coach	X					
SHIECC			X			
SmartTutor						X
SQL-Tutor		X				
T-Algebra				X		
The Incredible Machine					X	
Thermo-Tutor	X					
TutorJ						X
Wayang Outpost				X		
Why2-Atlas	X					
ZOSMAT						X
Category Totals	**14**	**12**	**3**	**11**	**10**	**8**

Another consideration is the amount of time developers spend on creating ITS. Estimates have said that it can take anywhere from 100-1000 hours to produce one hour of instruction (Heffernan et al., 2006). The developers of ITS also require extensive knowledge in cognitive science and the domains in order to produce an effective tutoring system. To alleviate the need for such highly-skilled people, authoring tools have been developed. These tools can be used by people who do not have a high level of computer expertise and have been shown to reduce the amount of time needed to produce one hour of content to 30-40 hours (Heffernan et al., 2006).

What level of understanding is the goal? Many ITS that were reviewed had mastery learning as its goal, but a few did not. Mastery learning is almost a natural consequence of ITS as the intelligent part of the tutoring systems aims to adapt to the learner. The tutoring system takes information it collects via the student model and makes decisions on whether or not certain knowledge components have been mastered (the mastery levels are dependent on the system). This can be accomplished since each knowledge component and its relations to other knowledge components can be mapped to problems presented by the tutoring system. This process of tagging problems with knowledge components has been called a Q-matrix (Birenbaum et al., 1993), transfer model (Feng & Heffernan, 2006), cognitive model (Feng & Heffernan 2006), and ontology (Melis et al., 2009). By tagging problems with knowledge components, the student model can estimate the likelihood that knowledge components have been mastered based on how a student solves a problem. While some ITS use probabilities to estimate the student's knowledge, other ITS simplify this process by setting a specific number of problems that need to be solved correctly before attaining mastery (Beck & Gong, 2013).

ActiveMath, as an example, is an intelligent tutoring system that focuses on mastery learning. It is unique in that it assesses mastery within the context of the first three levels of

Bloom's taxonomy: knowledge, comprehension, and application (Melis et al., 2001). Mastery of each knowledge component is then defined along each of these levels. If a student reads a text, then the knowledge mastery for that knowledge component is increased. If a student follows a worked-out example, then the comprehension mastery for the knowledge component is increased. Finally, if a student successfully solves an exercise, then the application mastery can be updated for that knowledge component.

While ActiveMath and others focus on teaching to the lower levels of Bloom's Taxonomy, some ITS shift the focus on the higher levels. AutoTutor (D'Mello & Graesser, 2012; Graesser, VanLehn, Rose, Jordan, & Harter, 2001) and CIRCSIM-Tutor (Woo et al., 2006) use simulated tutor dialogue to help students develop their reasoning skills in making judgments and inferences in problem-solving situations. Due to the nature of the higher levels of Bloom's Taxonomy, it seems that natural language dialogue may be better to facilitate the acquisition of those skills. Assessing whether a student is capable of synthesizing information is difficult to do when the only assessment tools available are multiple-choice questions. There was no research found that addresses this hypothesis though since most ITS do not seem to separate knowledge acquisition into any kind of taxonomy. It would be beneficial to test this hypothesis to see if different tutoring tactics are more conducive to teaching certain levels of knowledge. For example, basic mathematical skills may require a skill-and-drill approach to gain fluency whereas some advanced mathematical techniques would need a more conceptual approach and would benefit from talking it out with a tutor.

Going one step beyond mastery, Reading Tutor (Lalle, Mostow, Luengo, & Guin, 2013) and Wayang Outpost (Arroyo, Woolf, Cooper, Burleson, & Muldner, 2011) focus on developing oral reading fluency and math fluency. The distinction between mastery and fluency is that

mastery means that a student can reliably demonstrate a given skill whereas fluency adds a time component to it. Students must not only reliably demonstrate a skill but do so quickly. The reasoning for focusing on fluency is so that students can have reduced cognitive load when being presented with more difficult items.

Some ITS do not support mastery learning. Mastery learning assumes that students can freely interact with the system without time constraints in regard to how long it takes a student to master content. Since students are generally using ITS in the classroom, not all ITS can be self-paced. This means that some ITS are not concerned with presenting content adapted to the student, but instead they follow lesson plans created by teachers. ANDES is a physics tutor that exemplifies this point (VanLehn et al., 2005). Students are presented with a set of physics problems that are assigned by the teacher. ANDES' tutoring model is still intelligent in that hints are sequenced and feedback is adapted to the student, but it does not have a student model that utilizes the information collected about what the student knows. Therefore, its goal is not mastery learning but to be more of an intelligent homework aide.

One tutor falls in between supporting and not supporting mastery learning. Cognitive Tutor uses a modified mastery learning approach to fit the constraints of a classroom (Corbett, McLaughlin, & Scarpinatto, 2000; Pane, Griffin, McCaffrey, & Karam, 2014). Students have the opportunity to achieve mastery by solving problems and are considered to have graduated from that section if they do so. However, some students require more time to master and may fall behind the class if they are expected to reach mastery levels. As a compromise, Cognitive Tutor sets a maximum number of problems that students can attempt to solve for a given topic. If the student reaches the maximum number of problems without attaining mastery levels, then they move on to the next topic and are considered to have been promoted. This modification

allows the tutor to use mastery learning in a classroom without hindering the overall progression of the course.

What is used to create the expert? Developers of ITS have used many techniques of formalizing the knowledge, rules, and problem-solving strategies. Some ITS use subject-matter experts who create and organize the domain. ANDES (Schulze, 2000) had three physics professors who played an integral part as the tutoring system was developed. They used their expert knowledge to structure the domain and outline the feedback they would give to students. CIRCSIM-Tutor (Evens et al., 1997; Woo, Evens, Michael, & Rovick, 1991) also used a subject-matter expert to structure its domain with the added component that tutor dialogue was also based off transcripts of the subject-matter expert tutoring students.

Another approach to creating the expert is by using standardized test questions. Wayang Outpost (Arroyo et al., 2003) uses released questions from the SAT as its domain base. ASSISTments (Razzaq et al., 2005) did something similar by using questions that were released from a state standardized test to build the content base. Then, subject-matter experts annotated the questions to add tutoring. In a similar vein of using non-subject-matter experts to create the content, Reading Tutor used children's reading magazines to have age-appropriate content for students (Mostow & Aist, 1999).

Regardless of who or what is used to create the expert, domain models can be categorized into two prominent groups (i.e., model-tracing tutors or constraint-based tutors) with the rest belonging in an "other" group. The way the domain is constructed has implications for the student model as well and so each group will be discussed in more detail in the Student Model section.

The Tutoring Model

What informs tutoring strategies and tactics? Just like any human tutor, ITS are designed with philosophical underpinnings that explain why tutors tutor the way they do. This review question will be answered by looking at the strategies and tactics of ITS separately. The strategies here are defined as the philosophical paradigms that dictate the overall goals of a system. The tactics will describe the specific tutoring moves a system makes as it interacts with the student.

Strategies. Constructivism seems to be the paradigm of choice for ITS with eight of the 10 example systems using some form of a constructivist theory. While elements of other approaches are present in ITS like skill-and-drill and spaced repetition (Heffernan & Heffernan, 2014), the nature of a tutoring interaction lends itself to constructivism. A tutor is not a teacher. Their goal is not to instruct a student in new knowledge—a position explicitly stated by several ITS (Conati, Gertner, & VanLehn, 2002; Mitrovic, 1997). The instruction that is presented is generally remedial or to fill in knowledge gaps. To illustrate this point, consider ActiveMath. Andrews, Fleisher, and Liang (2010) described their constructivist approach as helping students "gain understanding and knowledge through their own experience and reflections on their own experiences" (p. 319). This approach informed the design of the system. Students are allowed relative freedom in navigating the course and setting goals (Melis et al., 2001). This interactivity provides the support a student needs while leaving room for the student to choose their own learning goals, context, and learning scenario. Once the student defines these parameters, ActiveMath generates a personalized e-book that the student can extend at any time if they feel there is additional knowledge they require (Melis et al., 2007).

Another system that uses a constructivist approach is AutoTutor. AutoTutor uses mixed-initiative dialogue to help scaffold students in explaining their reasoning, especially in the domain of physics (Nye, Graesser, & Hu, 2014). For example, if a student seems to be struggling with the concept of acceleration, AutoTutor may ask a question relating to acceleration within the context of the current problem. The student responds to the question by explaining their reasoning from which AutoTutor decides if the student has an accurate understanding of acceleration. If the perception is inaccurate, AutoTutor can ask additional questions to draw out the student's knowledge or provide hints and instruction. This type of interactivity puts the burden on the student to be responsible for their own knowledge.

Coached problem solving. This strategy is a sub-strategy of constructivism. ANDES implements this strategy in its tutoring design. VanLehn (1996) defined coached problem solving as the process a tutor and student go through to solve problems. Calling it a "kind of cognitive apprenticeship" (p. 29), VanLehn described the interaction as the coach (tutor) taking the lead and modeling cognitive processes for the student. Eventually, the coach fades its scaffolding until the student is solving problems on his own. In instances where the student requires help from the coach, ANDES provides hints that encourage the student to independently solve problems without explaining exactly which steps to take.

CIRCSIM-Tutor also uses a coached problem-solving approach for part of its instruction (Woo et al., 2006). Students are asked to fill out a prediction table that predicts physiological responses based on causal relationships. Once a student is done filling out the prediction table, the tutor and student engage in natural language dialogue to correct misunderstandings.

Zone of proximal development. Another strategy commonly cited, if not explicitly then certainly implied, is Vygotsky's zone of proximal development (ZPD). Here, tutoring systems

attempt to tailor problems to the student in such a way that each problem is just difficult enough that the student can't easily solve it but is still capable of doing it with help (Arroyo et al., 2003; Mitrovic, Martin, & Mayo, 2002; Wixon et al., 2014). Mastery learning and the ZPD seem to be intertwined in the design of ITS as the system tailors problems to the student in a way that the student remains engaged and challenged.

ACT-R. A prominent constructivist theory, ACT-R is used as a basis for several tutors to conceptualize knowledge (Koedinger & Anderson, 1998; Mitrovic, 1997). In making a distinction between declarative (i.e., factual knowledge) and procedural knowledge (i.e., application of declarative knowledge in performing a skill), ITS can make better tutoring decisions when providing remedial instruction. For example, if it is determined that a student is making errors in performing a mathematical formula (procedural knowledge), the system can determine if the student knows and understands the formula enough (declarative knowledge). If declarative knowledge is missing, those gaps can be filled in.

Learning from performance errors. SQL-Tutor expands the ACT-R theory to including learning from performance errors. This expansion is based off Ohlsson's theory (1996) where learning from errors involves two parts: error recognition and error correction. When someone does not have enough declarative knowledge to recognize an error, then that information needs to be taught to the student. This approach of learning from errors is the basis for constraint-based tutors (described in the Student Model section).

Natural language dialogue. Two of the 10 tutors used natural language dialogue as the main process of tutoring. This means that additional theories need to be used to guide the dialogue moves the tutor makes. AutoTutor and CIRCSIM-Tutor take different approaches to this.

AutoTutor uses explanation-based theories (Jackson, Mathews, Link, Olney, & Graesser, 2003), to draw out student knowledge as it requires them explain their reasoning. AutoTutor allows mixed-initiative dialogue meaning the student can ask questions and change the tutoring agenda based on their expressed needs (Person et al., 2000). This collaborative approach between tutor and student in constructing an answer belies the fact that AutoTutor does not actually use any sophisticated tutoring strategies (e.g., Socratic dialogue) like expert tutors use for dialogue (Rajan et al., 2001). Rather, AutoTutor relies on the tutoring strategies of unskilled tutors. The reasoning behind this is based on a previous meta-analysis in which tutor experience did not significantly predict learning gains (Cohen, J. Kulik, & C. Kulik, 1982; Graesser, Wiemer-Hastings, Wiemer-Hastings, Kreuz, & Tutoring Research Group, 1999). This finding led the authors of AutoTutor to hypothesize that, perhaps, it is not the sophisticated tutoring strategies or expertise of the tutor that matters, but rather the conversational properties of tutorial dialogue that are responsible for learning (Graesser et al., 2003).

In contrast to AutoTutor, CIRCSIM-Tutor does use sophisticated tutoring strategies for its dialogue approach (Woo et al., 2006). Relying on Socratic teaching, CIRCSIM-Tutor engages the student with questions and assertions that help the student develop qualitative reasoning skills. However, CIRCSIM-Tutor does not handle mixed-initiative dialogue well, preferring to restrict student responses by almost always ending a dialogue move with an imperative or question (Freedman, 1997).

Educational gaming and multimedia learning theory. Relying on educational gaming and multimedia to keep students engaged and to promote learning, Wayang Outpost is unique among the 10 in that it is an educational game. Set in at a research station in the Borneo Rainforest,

students receive assistance in solving SAT math problems and then go on adventures where similar math skills are contextualized within the game story (Wixon et al., 2014).

Non-constructivist approaches. Reading Tutor deviates from constructivist approaches. Reading Tutor is an ITS that develops reading fluency in first through fourth graders (Beck et al., 2003). Unlike the tutors that teach physics and mathematics, where conceptual knowledge plays a larger role, Reading Tutor requires more of a skill-and-drill approach for the intended audience—first through fourth graders. Mostow, Beck, Chalasani, Cuneo, and Jia (2002) explicitly stated that they "did not attempt to model the Reading Tutor's decision processes" (p. 5). Instead, they relied on expert reading interventions to assist students (Mostow & Aist, 1999; Zhang et al., 2008). This decision could suggest a reason for the predominant constructivist approach in other ITS. Students who have less knowledge benefit from more direct instruction and repeated practice opportunities. If this is true, it has implications for the design of ITS—the target audience should dictate which strategy to use.

If most of the 10 ITS are influenced by constructivism, it becomes difficult to tell what an intelligent tutoring system would look like if a different learning theory were used. This constraint on the basic design of a tutor prevents the exploration of new designs based on different learning theories. However, this constraint may also suggest that a constructivist strategy is what makes a tutoring system intelligent. After all, many direct instruction programs are used already by students. Take the popular, language-learning app, Duolingo, as an example. Gamification is used, content is adaptive, mastery criteria is used, feedback is given, and a student model is created and displayed (Duolingo, n.d.). Can Duolingo and similar direct-instruction programs be considered intelligent tutoring systems? Certainly, they can be considered intelligent systems, but do they meet the tutoring criterion? Perhaps the very nature

of tutoring requires a constructivist strategy. The tactics that are discussed next seem to indicate that a tutoring interaction is more than just a transfer of knowledge.

Tactics. Regardless of the strategies implemented by the tutoring system, many follow the same patterns of tutoring moves, most notably that of feedback and hint sequences. Some of the more unique tactics used include scaffolding and fading of feedback, worked-out examples (both correct and erroneous), and natural language dialogue used to structure self-explanation.

Feedback. One of the key features of intelligent tutoring is being able to give feedback that has been adapted to the student. Feedback can be generally categorized into flag feedback, step-based feedback, and answer-based feedback. Flag feedback is the simplest form of feedback. When implementing it, ITS flag an answer as simply right or wrong. ANDES (VanLehn et al., 2005) accomplishes this by coloring an answer or steps of an answer red if it is incorrect and green if it is correct. This does not provide much information to the student though since a student does not know why the answer was flagged. ActiveMath (Melis, Goguadze, Libbrecht, & Ullrich, 2009), Cognitive Tutor (Corbett et al., 2000), Reading Tutor (Mostow & Aist, 1999), SQL-Tutor (Mitrovic & Ohlsson, 1999), and Wayang Outpost (Arroyo, Woolf, et al., 2010) also mentioned some form of flag feedback similar to ANDES. Going a step further, step-based feedback increases the level of granularity by looking at the process a student goes through as he answers a question. From there, a step can be marked right or wrong and feedback can be given based on what is wrong. ANDES (VanLehn et al., 2005) does this by flagging a step right or wrong. A student can then ask for help which ANDES will provide—at times supplying general hints and at other times giving more specific feedback and remedial instruction. Finally, answer-based feedback gives feedback once a student submits an answer and not at all during the process of answering a question. ASSISTments (Koedinger,

McLaughlin, & Heffernan, 2010) demonstrates answer-based feedback. Once a student solves a problem, he inputs his answer. If the answer is correct, the student moves on. If the answer is incorrect, then the student goes through a scaffolded sequence that leads the student to the conclusion.

Some tutors also give feedback to students that is not based on student performance but rather desired metacognitive behaviors (Arroyo, Mehranian, & Woolf 2010; Roll, Aleven, McLaren, & Koedinger, 2011). Cognitive Tutor developed a standalone tutor called Help Tutor that assists students in developing ideal help-seeking behaviors (Roll et al., 2011). Wayang Outpost introduced pedagogical agents that give feedback that also promotes help-seeking behaviors but also promotes affective outcomes as well like helping students believe that intelligence is malleable (Arroyo, Mehranian, et al., 2010).

Hint sequences. Hint sequences generally follow the same pattern in each ITS, but the number of hints available varies from system to system. As an example, ANDES gives at first a general hint is given called a pointer hint (VanLehn et al., 2005). This hint points to what the system identifies as a potential problem in an answer or step without giving further instruction or explicitly saying what is wrong. If the student is still confused, he can ask for another hint. This hint is more specific and is called a teaching hint (VanLehn et al., 2005). This will provide further instruction. Lastly, the final hint in the sequence is called the bottom-out hint (Razzaq & Heffernan, 2006; VanLehn et al., 2005) and is a common, last hint in ITS (Aleven & Koedinger, 2001; Goguadze & Melis, 2008; Graesser, Hu, et al., 2001; Zhou et al., 1999). This gives the answer to the student, which can serve as a worked-example for some students. For other students, this prevents an endless, frustrating loop where the student does not have the requisite

knowledge components to answer the question successfully. The tutor can move on and adapt the instruction to address the student's knowledge gaps if needed.

Natural Language Dialogue. In an effort to more completely mimic human tutors, some ITS have implemented natural language processing. This dialogue, exemplified in AutoTutor (Nye et al., 2014), allows students to converse with a pedagogical agent. The agent can then respond as a tutor would. If AutoTutor detects knowledge gaps, it can fill those in. Alternatively, AutoTutor can have a student explain their thought process. This self-explanation has been demonstrated in other ITS and has proven effective in helping students learn (Gertner & VanLehn, 2000).

AutoTutor has myriad dialogue moves available, but follows the general framework of first, asking a question; second, the student answers the questions; third, the tutor gives short, immediate feedback on the quality of the answer; fourth, the tutor and the student collaboratively work together to improve the quality of the answer; and, lastly, the tutor assesses the student's understanding (Graesser et al., 1999). This framework is based on an analysis of nearly 100 hours of naturalistic tutoring sessions. In later development, the fifth step was dropped since it produced inconsistent student responses (Person, Graesser, Kreuz, & Pomeroy, 2003).

Scaffolding. The idea behind scaffolding is to remove assistance gradually as a student gains proficiency. In ITS, scaffolding takes the form of help from the tutor. While the ITS wean the students off the assistance, what is interesting to note is the timing of the scaffolding. Scaffolding can occur before, during, or after a problem.

Reading Tutor is an example of providing assistance before a problem (Beck et al., 2003). The Reading Tutor analyzes the words presented before a student starts reading. If it determines that some of the words would be exceptionally difficulty to the student (e.g., first

encounter of the word, previous poor performance), then it will assist the student by doing something like saying the word aloud. Beck et al., (2003) stated the rationale for this decision. If a student practices saying a word incorrectly, it will be more difficult to fix the pronunciation later—hence the preemptive assistance.

Another example, from Schulze et al., (2000), shows ANDES providing unsolicited help during a problem. In this case, the student model determined that the student does not understand a particular physics concept. A mini-lesson is then presented to the student to fill in the knowledge gap. ActiveMath (Melis & Andres, 2005) also suggests instructional intervention to a student mid-lesson. As a student is solving a math problem, if the student model determines there are missing knowledge components, it can detect if the content had been reviewed by the student or not. It can even detect if the student just skipped over a small part of the lesson. If ActiveMath detects this, it will suggest revisiting the material.

After student performance is measured, some ITS will give feedback at the end without any help in between. ASSISTments (Razzaq, Heffernan, & Lindeman, 2007) conducted an experiment to determine what level of tutor interaction is best. There were three conditions: scaffolding + hints, hints on demand, and delayed feedback. The results showed that those students with less knowledge benefited more from scaffolding + hints. Those students who had more knowledge (honors students) benefited more from the delayed feedback setting. This demonstrates that a one-size-fits-all to teaching does not work with all students. That leads to the next tactic: adaptivity.

Adaptivity. One of the main advantages of one-on-one tutoring is being able to adapt to the student's needs in real time. ITS accomplish this by the wealth of data they collect about a student. From these data, a system infers what the student knows and can adapt the presentation

of problems to a student. The mastery level inferred from the data influences the number of problems presented as well as the difficulty (Ullrich, 2003).

Spaced Practice. To promote retention, ASSISTments developed a system that assesses learning on a spaced schedule of 7, 14, 28, and 60 days after a student initially masters a knowledge component (Heffernan & Heffernan, 2014). If a student fails their reassessment, they are assigned a relearning activity and the schedule for spaced practice is reset.

Anti-gaming Tactics. Some students will disengage with the learning process and behave in a way that minimizes their effort and maximizes their progress with the tutor. This gaming behavior is problematic since the student is not learning the material. Some tutors have attempted to address this issue through various tutor interventions. ANDES encourages good behaviors by displaying a score on the screen (VanLehn, Van De Sande, Shelby, & Gershmann, 2010). The score decreases when students use bad physics problem-solving strategies or engage in gaming behaviors.

Cognitive Tutor's developers also addressed the issue by introducing a pedagogical agent in the form of an animated dog called Scooter the Tutor (Baker et al., 2006). Scooter promotes non-gaming behavior by looking happy and giving positive remarks when a student is behaving properly. But, if Scooter detects the student is gaming the system, he will look progressively more upset until he is displaying anger. This emotional response is meant to trigger social norms in the student so that the student no longer games. Scooter will also give supplementary exercises to the student when gaming is detected so that the student has another opportunity to learn the material. Unfortunately, it seems that as anti-gaming measures are introduced, students will find new ways of gaming the system (Baker et al., 2006). While gaming is not a common

behavior, many students do engage in it and thought needs to be put into the design of the ITS to prevent gaming.

The Student Model

How is the student model created? The goal of the student model is to map what a student knows by inferring student knowledge from student performance (Beck & Chang, 2007). This process, called *knowledge tracing*, is quite complex. There are popular techniques used to perform the knowledge tracing, but each system seems to adapt existing techniques to meet their system's needs and capabilities. Additionally, the underlying tutoring strategy seems to dictate modeling decisions, like whether or not to include forgetting as a variable in the model. Regardless of the differences in each system, the student models are classifiable in four categories: model tracing tutors, constraint-based tutors, example tracing tutors, and effort-based tutors. These categories answer the question of how a student model is constructed.

Model tracing tutors. Of the 10 systems, three were model-tracing tutors. Kodaganallur, Weitz, and Rosenthal (2005) described model-tracing tutors (MTT) as being process-centric. This means that the tutoring system attempts to infer the process or path that a student took to reach a solution. Where it detects errors in this path, remediation is provided. This is possible due to the domain knowledge being broken down into a series of production rules. Kodaganallur et al. (2005) identified two sets of rules: expert rules and buggy rules. Expert rules denote the solution path an expert would take to solve a problem. These rules are further broken down into planning rules which represent procedural knowledge and operator rules which represent declarative knowledge. Overall, these rules state that IF condition X is true, THEN do this thing. If the student did that thing, then the production rule is satisfied. Buggy rules represent the misconceptions a student could have. They are formalized the same way, it just demonstrates a

misapplication of knowledge in the THEN clause. So, IF condition X is true, THEN do this incorrect thing.

Model-tracing tutors have a model tracer that will trace the student's solution path by executing a series of rules that eventually lead to the solution the student inputs, hence the name model tracing. If the model tracer can reach the same solution, then it can be said to have successfully inferred the student's process. If the rule series includes buggy rules, then the tutor provides remedial instruction based on the knowledge component that was violated by the misconception.

Constraint-based tutors. Of the 10 systems, one was a constraint-based tutor and it was the first of its kind (Mitrovic, 2010). In contrast to MTT, constraint-based tutors (CBT) are considered product-centric (Kodaganallur et al., 2005). Rather than focusing on the process a student uses to identify buggy knowledge, CBT use the student's solution to identify misconceptions. Instead of identifying a series of rules that create a solution path, CBT have constraints. These constraints set the conditions that must be satisfied for a solution to be correct. In this case, the process that led to the solution is unimportant. Constraints can be formalized as an ordered pair <Cr, Cs>. Cr stands for the relevance condition (the IF-statement) and Cs is the satisfaction condition (the THEN-statement). That is, the relevance condition describes when the constraint should be applied, and the satisfaction condition describes another condition that should be true for any correct solution that meets Cr. Like production rules, constraints are formalized in IF-THEN clauses. The difference being that constraints say that IF condition X is true, THEN condition Y better also be true (rather than stating what needs to be done). Constraint-based tutors detect student misconceptions when a relevance condition is true,

but the satisfaction condition is not. From there, CBT can provide remediation based on the violated constraint.

Example-tracing tutors. Unlike MTT or CBT, example-tracing tutors (ETT) are not rules-based and are focused on a single problem rather than an entire domain. The ASSISTments system is an example of an ETT (Mendicino, Razzaq, & Heffernan, 2009). Within this system, a transfer model is created. This transfer model is a mapping of knowledge components to problems and scaffolding questions (Feng & Heffernan, 2006). While this may seem similar to MTT and CBT, the difference is that there are no rules or constraints associated with each knowledge component. Rather, ASSISTments tracks how many times a knowledge component has been applied to a question. Using this information, ASSISTments can infer mastery of knowledge components based on the student answering questions correctly. The scaffolding and hints provided in the system give remediation to the student if they answer the question incorrectly.

Effort-based tutors. Wayang Outpost pursued a unique route for its student model. Noting that typical knowledge tracing methods focus on performance in terms of incorrect answers and do not take into consideration help requests or timing, Wayang Outpost's developers created a new model that valued effort over performance (Arroyo, Mehranian, et al., 2010). Based on the number of hints requested, the number of attempts to solve a problem, and the time it took to solve a problem, Wayang Outpost can discern between behaviors that relate to student engagement in addition to behaviors related to skill mastery. For example, this model could detect when a student demonstrated mastery without much effort. In response, the tutor can increase the desired problem difficulty and maybe show the student's learning progress. Another example could be that the student is avoiding hints but trying hard. In that case,

Wayang Outpost can respond by reducing the problem difficulty and maybe offering hints on the next incorrect answer.

Other methods. In ill-defined domains, it seems that knowledge tracing is not as simple as creating a knowledge map and then tracing student performance against the mapping. Reading Tutor identifies with this hardship and used a different method of creating a student model. Rather than creating and relying on expert solution paths, Reading Tutor's developers took an inductive approach (Beck et al., 2003). They determined which variables predict student knowledge at a coarse level—reading latency (pauses between words) and help requests. They then used this information to let the tutor model know when to intervene (e.g., providing preemptive assistance).

What issues are there regarding the development of the student model? Several issues surround the construction of a student model. First, the developers of ANDES described an assignment of credit/blame problem (Conati, Gertner, & VanLehn, 2002). This problem describes the difficulty of ascertaining the amount of credit or blame should be assigned to solution. If a student answers a question incorrectly and several knowledge components were found to be missing, how much blame does each component receive for the student answering the question incorrectly? In other words, by how much is the mastery level of that knowledge component reduced? The same holds true when a student identifies a correct solution—how much credit will you give to each knowledge component present in the problem? This is especially problematic when students are not required to show their work and multiple solution paths exist.

A second problem is the identifiability issue. Beck and Change (2007) demonstrated this issue by creating separate student models that all predict with similar accuracy student

performance. The assumptions behind these models are wildly different. One model predicted that a student would need 24 practice opportunities to master a knowledge component whereas another model predicted 32 practice opportunities. If both accurately predict student performance, which model should be used? This identifiability issue has further implications for tutoring decisions since some tutoring moves are dependent on estimates of student knowledge (e.g., presenting remedial instruction on missing knowledge components).

A third problem involves ill-defined domains and the restriction of input. One of the reasons Reading Tutor had difficulties with its student model is that reading fluency isn't clearly defined as a domain and students were restricted in which input could be accepted (Beck, Jia, & Mostow, 2003). Student models that have been used in language learning have been able to rely on typed input. However, Reading Tutor is aimed at first through fourth graders who cannot type. And so, Reading Tutor had to rely on different data to assess student performance.

A fourth problem arises when ITS use standardized questions as content. Two of the 10 systems relied on these types of questions for their content. Wayang Outpost described the problem in more detail (Arroyo, Beal, Murray, Walles, & Woolf, 2004). First, there is no clear idea of what the problem difficulty will be on future versions of the SAT nor which skills will be implemented. Second, the skills that are used are not used in more than a few problems, which makes estimating student knowledge difficult. This takes some control over the domain model from Wayang Outpost which could explain why Wayang Outpost shifted from a traditional knowledge tracing model to one that focuses more on effort.

A final problem that was identified concerns the philosophical question, "What level of intelligence is necessary for ITS?" AutoTutor's developers have argued that since real tutors have a hard time gauging student understanding but can produce significant learning gains, then

it would be reasonable to expect AutoTutor to perform reasonably well given a shallow understanding of student knowledge (Graesser et al., 1999). SQL-Tutor's developers shared a similar thought stating that "if the goal is to model student's knowledge completely and precisely, student modeling is bound to be intractable," (Mitrovic, 1998, p. 1). Echoing the developers of AutoTutor, SQL-Tutor's developers say that human tutors have "very loose models of their learners, and yet are highly effective in what they do," (Mitrovic, 1998, p. 1).

What data is collected? When considering that millions of student interactions can be logged by an intelligent tutoring system over the course of a semester, it becomes a critical matter to understand which data are useful to collect. To answer this question, the grain-size of data will be discussed. Following that, different static and dynamic variables that are collected will be presented. Finally, less observable data (engagement and affect) are discussed.

Grain-size. Grain-size refers to the granularity of data. Generally, grain-size can be classified as fine-grained or coarse-grained. Fine-grained data can represent step-based or even substep-based data collected by the system. Examples of such data include timestamps, keystrokes, and mouse clicks. In contrast, coarse-grained data can be represented by answer-based data. That is, final solutions, affect over time, and user preferences are examples of coarse-grained data.

Static variables. Static variables are variables that are not prone to change throughout the use of a tutoring system. Such variables include field of study, learning scenario, goal concepts, and preferences (Melis et al., 2001).

Dynamic variables. Dynamic variables are variables that are prone to change throughout the use of a tutoring system. For example, knowledge mastery values for knowledge tracing are updated as students solve problem and receive assistance (Melis et al., 2001). Timestamps are

another variable that is collected. Reading Tutor (Joseph, 2005) uses timestamps to detect when a student is making hasty responses as a sign of disengagement from assessment questions. ActiveMath uses timestamps in a similar manner to estimate if a student is skimming through content or skipping it completely (Melis & Andres, 2005). Additional examples of dynamic variables include level of assistance/hints (Koedinger et al., 2010), reading latency (Mostow & Aist, 1999), and reading prosody (Mostow & Duong, 2009).

Engagement. Engagement is an important part of the student model to measure. Since the tutoring model can present assistance and hints on-demand at times, students who are not engaged are more likely to game the system (Joseph, 2005). Gaming the system can be defined as students minimizing their effort to maximize their gain. Examples of gaming the system include repeatedly hitting the help button until a bottom-out hint is provided. Other forms exist depending on the structure of the system. Those students who game the system do not learn as much, although there has been evidence to show that gaming behavior actually helps (Ringenberg & VanLehn, 2006**)**. In these cases, the researchers hypothesize that these students use the bottom-out hints as worked examples.

Affect. The emotional state of a student can impact his performance when working with a tutoring system. For example, anxiety can inhibit student performance (Ullrich & Libbrecht, 2008)**.** Affect-sensitive tutoring systems use a variety of methods to measure affect including facial recognition, a pressure sensitive mouse, posture-sensitive chairs, and self-report measures (Nye et al., 2014).

Being able to detect and respond to the affective states of students is a growing trend and focus in ITS. AutoTutor's work identified the six academic emotions of boredom, frustration, confusion, engagement, delight, and surprise (D'Mello et al., 2008). The emotions of delight and

surprise were eventually eliminated since they are not very common in academic settings. Cognitive Tutor could identify similar emotions with five affective states being detectable: boredom, confusion, engaged concentration, frustration, and "?" which represented everything else (Baker et al., 2012).

The Interface

The interfaces of the ITS reviewed can be distinguished by four dichotomous categories: (a) web- vs. desktop-based, (b) typed vs. spoken inputs, (c) including a pedagogical agent, (d) and being game-based. Each of these categories will be discussed. Afterwards, the research done on user experience will be addressed.

Web- vs. desktop-based. In the early 90s, ITS seemed to be primarily desktop-based. ANDES (Schulze et al., 2000) is an example of this. With the rise of the personal computers and faster internet connections, there has been a shift from desktop-based ITS to web-based ITS. ITS have taken advantage of this shift to provide intelligent tutoring both in the school and at home. ASSISTments provides an example of an ITS used both at home and school (Tvarozek, Kravcik, & Bielikova, 2008). At times, teachers use ASSISTments in the classroom, and, at other times, homework is assigned. This flexibility allows ITS to be accessed anywhere that has the necessary system requirements and an internet connection. This accessibility has implications for ITS development.

First, there are the physical limitations in terms of things like servers and router bandwidth. Care needs to be taken to properly manage databases and data access. For example, Reading Tutor did not develop its reading tutor with a strong student model in mind (Mostow et al., 2002). They collected data and stored it, but several articles described their struggles as they attempted to format old data to be interpretable and the adjustments they had to make in

collecting data they needed. Another system described the latency periods between calling info from a database and using it to generate student reports (Feng & Heffernan, 2006). Some of the tables required several minutes to generate the reports. Such latency periods would be unacceptable by users today.

Second, the accessibility can improve development times of content. Several ITS have built web-based authoring tools that simplify the development process (Heffernan et al., 2006; Schulze et al., 2000; Goguadze & Tsigler, 2007). This simplification allows teachers to create content catered to their needs. This crowdsourcing of content development can then be shared for others to use. As was described before, development time is a major hindrance to the development of ITS, but web-based solutions (i.e., authoring tools) seem to be helping (Heffernan et al., 2006).

Typed vs. spoken input. Another differentiation between ITS seems to be their capacity for typed and spoken input. ANDES takes a "pencil and paper" approach to their design (VanLehn et al., 2005). This design was used to promote transfer to physics problems outside the scope of the system (i.e., actual paper and pencil problems). In this case, students type their solutions step-by-step as if they were solving math problems on paper. They can even use the interface to draw and label diagrams. However, ANDES doesn't support verbal input.

Reading Tutor (Mostow & Aist, 1999) supports only verbal input. This makes sense considering the domain—oral reading fluency. Their design was based on the assumption that non-readers would not be able to write. So, while students can use a mouse to navigate the program and seek help, their input is limited to spoken word excluding some assessment questions (Hoppcetal, 2003).

In between the two extremes is AutoTutor. AutoTutor supports both typed input and verbal input. Nye et al., (2014) report comparable performance on these two types of input, but with a slight favor for typed input due to errors in speech recognition. In either case, students can input their answers however they want—especially when interacting with the pedagogical agent built into AutoTutor, the subject of the next section.

Including a pedagogical agent. Some ITS have explored the addition of a virtual teacher to their system. This virtual teacher is referred to as a pedagogical agent (PA). Reading Tutor includes a pedagogical agent in the simplest form—a smiley face (Mostow & Aist, 1999). AutoTutor goes to the other extreme in terms of sophistication, creating a 3D rendering of a virtual teacher capable of expressing emotion and talking with the student (Nye et al., 2014).

Reading Tutor's PA is used to provide feedback to the student. It has been described as a "persona that actively watches and patiently listens" (p. 410). Feedback and help appears in the form of a speech bubble above the PA. This interactivity is meant to give the students a sense that someone is listening to them.

AutoTutor's PA was designed to engage in dialogue with students and to provide feedback to the student. The dialogue encourages self-explanation and metacognitive scaffolding (Nye et al., 2014). Unlike typical feedback in other ITS that used flag feedback or textual assistance, AutoTutor's PA can use spoken feedback with intonation and facial expressions to demonstrate disapproval, approval, and other emotions to the student (Craig, Graesser, Sullins, & Gholson, 2004; D'Mello et al., 2008, Nye et al., 2014).

Some PAs do not focus on giving instruction or performance feedback. Rather, they help the student with metacognitive feedback. One that was mentioned earlier was Cognitive Tutor's Scooter the Tutor who discouraged students from gaming the system (Baker et al., 2006).

Wayang Outpost has a PA that acts more like a learning companion. A student working at a computer is displayed on the screen and can be a boy (Jake) or a girl (Jane) and can represent minority races. Jake and Jane are empathic agents that reflect the emotions of students (Arroyo, Woolf, Royer, & Tai, 2009), provide metacognitive or affective feedback, or direct the student to use the help button to request further assistance from the system (Cooper et al., 2009).

User experience. Studies on user experience (UX) have been sparse and mostly anecdotal. Of the ITS reviewed, only five had any mention of user experience and, even then, those were not thorough studies. ANDES had the most extensive information regarding what students thought about the system. In Schulze et al. (2000), ANDES' developers described anecdotal reports of students being frustrated by the interface. ANDES forces students to define variables, which, at the beginning, seemed useless and like extra work. They report that students eventually appreciated this constraint after extended use.

Another study from ANDES (Schulze et al., 2000) drew two conclusions regarding student attitudes. First, those who used ANDES frequently liked it, but those who did not use it as much did not like it. VanLehn et al. (2005) found that 40-50% of students liked ANDES and 40-55% of students disliked ANDES. Second, Schulze et al., (2000) concluded that problem sets should include challenging problems to demonstrate the utility of ANDES and justify the learning curve. To help with the learning curve, various instructional materials were created like tutorials, manual, and short videos. Of these, the short videos received positive reviews from the students.

Aside from the few reports from ANDES, not much research was found about user experience. Surely anecdotal evidence was used in the initial design and subsequent versions, but not much has been included in the literature to help the field in this regard. User experience

is important to the development of ITS. A tutoring system can have impressive learning gains, but unless students like working with it, the effectiveness of the tutoring system will remain unrealized. Additionally, it would be difficult to get buy-in from stakeholders (e.g., school administration, teachers, and parents) in order to implement ITS in a school if nobody liked the design. Further research, or at least the publication of research already done, needs to become more of a priority if ITS are to become more widespread in their use.

Cognitive Tutor did focus a lot on UX, but not from the perspective of students. Cognitive Tutor had the goal of creating a complete course experience that includes textbooks, supplementary materials, and teacher training (Ritter, Anderson, Koedinger, & Corbett, 2007). This decision allowed Cognitive Tutor to become a regular part of the classroom experience since it was designed with teachers in mind. Cognitive Tutor has been adopted by teachers throughout the country and is used by about 500,000 students regularly every year (Koedinger, McLaughlin, & Heffernan, 2010).

Learning Gains

The 2-sigma effect is a well-known problem in AIED. It comes from an article written by Bloom in 1984. Bloom describes research that attempts to measure the effect sizes of three different instructional conditions: (a) conventional instruction, (b) mastery learning, and (c) human tutoring. They found that the tutoring condition performed about two standard deviations above the conventional classroom condition. This means that the average, tutored student performed better than 98% of students in the conventional classroom. Mastery learning also had impressive results as the average student here performed better than the conventional classroom by about one standard deviation, or better than 84% of students in the conventional classroom. The appeal in these data is that students can reach the level of achievement reserved for the top

20% of a conventional classroom through innovative teaching practices (i.e., mastery learning). While Bloom's article focused on mastery learning benefits, the 2-sigma effect became a benchmark for excellence in AIED. How can we design instruction to reach that level of student performance without the 1:1 tutoring costs?

ITS attempt to tackle this problem, not primarily through innovative teaching practices, but by providing a computerized tutor to every student. It would follow that the benchmark for a successful computer tutor would be the effect size of a human tutor. Thus, it becomes necessary to address the effect sizes of ITS to see if they are close to achieving the 2-sigma effect. However, newer research challenges the notion that human tutors even achieve the 2-sigma effect.

VanLehn (2011) conducted a meta-analysis to review the relative effectiveness of different tutoring scenarios (e.g., human tutoring, computer-based tutoring, or no tutoring). After identifying 99 articles that met inclusionary criteria, VanLehn's results contradicted Bloom's (1984) findings. Bloom claimed that human tutoring had an effect size of 2.0. VanLehn found that human tutors were closer to 0.79 when compared to no tutoring. Additionally, VanLehn found that step-based ITS are much closer to approximating human tutors than previously thought. In his review, ITS had an effect size of 0.76 when compared to no tutoring. This new information has the potential to redefine the 2-sigma problem. If one ignores the two sigma that Bloom identified and focuses on the concept of ITS performing as well as human tutors, then it seems that some ITS have already reached that goal.

While VanLehn's (2011) study provides a mean effect size for step-based ITS, the current paper will provide the range and mean of effect sizes based on each ITS. Thus, we can

see a breakdown of the effectiveness of each tutoring system and whether they have solved the redefined 2-sigma problem.

Of the 10 ITS reviewed for this paper, only 6 of them had learning gains reported in terms of effect sizes. Figure 1 is a graph showing the reported ranges and averages of the effect sizes of the tutors in relation to the benchmarks described above (i.e., average tutor, 0.79 sigma, and 2 sigma). The four that did not (ActiveMath, Aplusix, Reading Tutor, and Wayang Outpost) may not have reported learning gains due to the nature of the tutoring system. ActiveMath's focus is to dynamically create textbooks for the students (Andres et al., 2010). The textbooks do include questions and answers, but due to the system being based on mastery learning, perhaps effect sizes are not a good measure of differences in student achievement. If each child is expected to master content to a certain level, this would artificially inflate the effect sizes when compared to a non-mastery learning course. Perhaps another measure to look at in these cases is the length of time the students required to gain mastery. A more efficient and effective system ought to help students achieve mastery quicker.

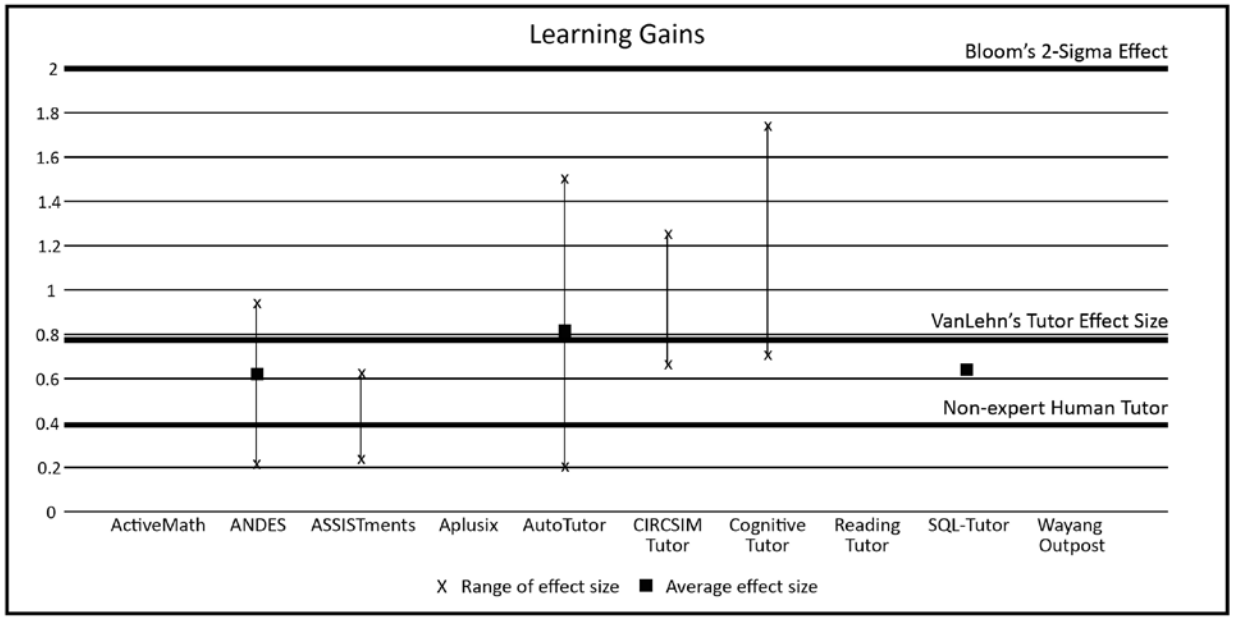

Figure 1. Reported average and ranges of effect sizes for the 10 ITS.

Reading Tutor's focus is on oral reading fluency. The data collected in this system (e.g., latency (Beck et al., 2003), prosody (Mostow & Duong, 2009)) is somewhat subjective. While they use audio recordings of adults to approximate appropriate answers, there is no standardized, objective test that a student can take to measure the optimal latency period between words or the best intonation of a sentence. This ties into reading being an ill-defined domain. Perhaps the reporting of learning gains is too difficult to be practical in ill-defined domains.

Wayang Outpost did report learning gains, but not in terms of effect sizes. Instead, learning gains were reported as percentage point gains from pre- to posttests. For example, one study found that there was a 27% increase from pre- to posttest (Arroyo, Woolf, & Beal, 2006). Students were divided into low/high performing groups using a median split technique based on the pre-test. Low-performers took more advantage of the system functions and demonstrated significant improvement from pre- to posttest while high-performers showed no significant change. It seems that the system is more valuable for those who start off at a disadvantage.

The remaining tutors boasted sizeable learning gains. ANDES reported on learning gains over a five-year period with effect sizes that ranged from 0.21 to 0.92 (VanLehn et al., 2005). Overall, VanLehn et al., reported an average effect size of 0.61. While this is a bit lower than 0.79 reported by VanLehn (2011), students would perform better than or as well as approximately 73% of students in a conventional classroom. VanLehn's effect size for tutors, as a comparison, would have students performing better than or as well as approximately 79% percent of students. So, while the effect sizes are considerably different, the practical effect (performing better than or as well as other students) seems to be less significant by comparison.

Looking at ASSISTments, effect sizes were reported within the range of 0.16 to 0.85 (Koedinger et al., 2010; Mendicino, et al., 2009; Razzaq & Heffernan, 2006; Razzaq &

Heffernan, 2009; Singh et al., 2011). No average effect size was found in the review for ASSISTments. Applying the same standard (0.79 effect size, 79th percentile) to ASSISTments, we find on the low end that ASSISTments did not perform as well when compared to human tutors. With a low effect size of 0.16, students would perform better than or as well as about 56% percent of students—about average.

AutoTutor also had a range of effect sizes reported ranging from .2 to 1.5 standard deviations with an average of 0.8 (Graesser, Chipman, Haynes, & Olney, 2005). These effect sizes are gathered from experiments on over 1000 students and almost always were significant. CIRCSIM-Tutor, the other dialogue-based tutor of the 10, only had one study that reported learning gains (Woo et al., 2006). A control group where students read from a textbook was compared to students who used CIRCSIM-Tutor. The control group outperformed CIRCSIM-Tutor in memorizing relationship information with a significant effect size of 1.27 compared to CIRCSIM-Tutor's significant 0.65 effect size for the same portion of the test. However, CIRCSIM-Tutor outperformed the control group in terms of problem-solving and making correct predictions with a significant effect size of 1.24 for CIRCSIM-Tutor and 0.48 for the control.

The developers of Cognitive Tutor (Corbett et al., 2000) and SQL-Tutor (Martin, Koedinger, Mitrovic, & Mathan, 2005) mention another aspect of learning gains that should be considered here and that is the amount of time saved. Cognitive Tutor's developers found that students using the system were able to complete problems in as little as one-third the amount of time than a control group of students (Anderson, Corbett, Koedinger, & Pelletier, 1995). SQL-Tutor focused on learning curves that put the number of times a knowledge component has been used on the x-axis and the error rate on the y-axis. Plotting this information, the curve demonstrates the speed at which the student is learning the material. The developers of SQL-

Tutor argue that this measure can be a more accurate way of performing a quantitative comparison between disparate systems since the system that has a better learning curve could arguably be superior (Martin et al., 2005)**.** Unfortunately, this isn't without its own difficulties. Martin et al., (2005) identified two problems with this approach. First, having different knowledge components being measured in two systems can skew the learning curves. For example, if one system has a "large number of trivially satisfied rules" (Martin et al., 2005, p. 7) that a student population already knows, then these could reduce the error rate for that population while another system foregoes similar rules. The second problem relates to problem difficulty. If one system consistently presents more challenging content, then it will naturally have a higher error rate than a system that does not.

While the learning gains may vary from system to system, it is encouraging to know that ITS, overall, are capable of producing learning gains equivalent to their human counterparts (VanLehn, 2011). This information can help ITS become more mainstream and accepted in classrooms. After all, the costs of providing a human tutor for each student would be infeasible in our current system. However, ITS can provide a scalable solution to achieving the learning gains of a human tutor.

Conclusion

This paper sought to accomplish three goals. First, it provided a systematic, comparative review of several ITS. Second, it summarizes problems and solutions presented and solved by developers of ITS by consolidating the knowledge of the field into a single review. Third, it provided a unified language from which ITS can be reviewed and understood in the same context.

The findings of this review centered on a new 5-Component Framework. The first component, the domain model, showed that most ITS are focused on science, technology, and mathematics. Within these fields, ITS generally have mastery learning as its desired level of understanding. The second component, the tutor model, showed that constructivism is the strategy that informs most ITS. The tactics employed in the ITS stem from this paradigm. The third component, the student model, identified the several ways ITS infer what a student knows. It described the variety of data that is collected by an ITS and how it is used to build the student model. The fourth component, the interface, revealed that most ITS are now web-based, but vary in their capacity to interact with students. It also showed that user experience is underreported and ought to be included more in the research. Finally, the fifth component, learning gains, demonstrated that ITS are capable of producing learning gains equivalent to a human tutor. However, reporting learning gains does not seem to be a focus of the literature.

From this review, we suggest that further research should focus on three things. First, development of ITS for domains other than science, technology, and math. Exploring ill-defined domains could lead the research community in a direction where new models and methodologies can be established as we struggle to represent the knowledge of those domains. Second, conducting or publishing more research on user experience. The research on user experience was sparse. Having more information available can help in the design of ITS. For example, if the research suggests that pedagogical agents add no significant benefit to the user experience or learning outcomes, then ITS can be slimmed down as they are excluded. Not only would this cut down on development costs and time, but also more resources can be devoted to the development of a data-backed interface that supports learning. Third, learning gains need to be more fully included in the research agenda for publications. If there is no clear focus on conducting valid

experiments that measure the effectiveness of ITS, then it becomes difficult to not only compare IT, but also to know what is effective. This is not to say that the effectiveness of learning gains can only be measured in effect sizes. It is a call to create a standardized way of measuring effectiveness so that developers and users of ITS can make comparisons.

Following these recommendations may lead to the broader acceptance and implementation of ITS in schools. If developers of ITS can demonstrate that ITS are a viable solution for more than science, technology, and math; if they can create ITS that are easy to use and visually pleasing; and if they can provide tutoring environments that are effective, worthwhile tools; then maybe ITS can be accepted as a regular tool in a classroom. Thus, ITS can provide a 1:1 tutoring experience for every student, equalizing the quality of education across the board.

References

Aleven, V., & Koedinger, K. R. (2001). Investigations into help seeking and learning with a cognitive tutor. In R. Luckin (Ed.), *Papers of the AIED-2001 workshop on help provision and help seeking in interactive learning environments* (pp. 47-58), San Antonio, Texas: St. Mary's University.

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.

Andres, E., Fleischer, R., & Liang, M. (2010). An adaptive theory of computation online course in ActiveMath. In *Computer Science and Education (ICCSE), 2010 5th International Conference on* (pp. 317-322). IEEE.

Arroyo, I., Beal, C., Bergman, A., Lindenmuth, M., Marshall, D., & Woolf, B. P. (2003). Intelligent tutoring for high-stakes achievement tests. *Artificial Intelligence in Education: Shaping the Future of Learning Through Intelligent Technologies*, *97*, 365-368.

Arroyo, I., Beal, C., Murray, T., Walles, R., & Woolf, B. P. (2004). Web-based intelligent multimedia tutoring for high stakes achievement tests. In J. C. Lester, R. M. Vicari, & F. Paraguacu (Eds.), *Intelligent tutoring systems* (pp. 468-477). Berlin/Heidelberg: Springer.

Arroyo, I., Mehranian, H., & Woolf, B. P. (2010). Effort-based tutoring: An empirical approach to intelligent tutoring. In R. Baker, A. Merceron, P. Pavlik (Eds.), *Educational Data Mining 2010* (pp. 1-10). Pittsburgh, PA: Carnegie Mellon University

Arroyo, I., Woolf, B. P., & Beal, C. R. (2006). Addressing cognitive differences and gender during problem solving. *International Journal of Technology, Instruction, Cognition and Learning*, *4*, 31-63.

Arroyo, I., Woolf, B. P., Cooper, D. G., Burleson, W., & Muldner, K. (2011). The impact of animated pedagogical agents on girls' and boys' emotions, attitudes, behaviors and learning. In I. Aedo et al., (Eds.), *Proceedings of Advanced Learning Technologies (ICALT), 2011 11th IEEE International Conference* (pp. 506-510), Athens, Georgia: IEEE.

Arroyo, I., Woolf, B. P., Royer, J. M., & Tai, M. (2009). Affective gendered learning companions. *AIED*, *200,* 41-48.

Arroyo, I., Woolf, B. P., Royer, J. M., Tai, M., Muldner, K., Burleson, W., & Cooper, D. (2010). *Gender matters: The impact of animated agents on students' affect, behavior and learning.* Amherst: UMASS.

Baker, R. S., Corbett, A. T., Koedinger, K. R., Evenson, S., Roll, I., Wagner, A. Z., ... & Beck, J. E. (2006). Adapting to when students game an intelligent tutoring system. In Ikeda et al. (Eds.), *International Conference on Intelligent Tutoring Systems* (pp. 392-401). Berlin/Heidelberg: Springer.

Baker, R. S., Gowda, S., Wixon, M., Kalka, J., Wagner, A., Salvi, A., ... & Rossi, L. (2012). Sensor-free automated detection of affect in a Cognitive Tutor for algebra. In K. Yacef et al., (Eds.), *Educational Data Mining 2012* (pp. 126-133). Chania, Greece: International Educational Data Mining Society.

Beck, J. E., & Chang, K. M. (2007). Identifiability: A fundamental problem of student modeling. In C. Conati, K. McCoy, & G. Paliouras (Eds.), *International Conference on User Modeling* (pp. 137-146). Berlin/Heidelberg: Springer.

Beck, J. E., & Gong, Y. (2013). Wheel-spinning: Students who fail to master a skill. In H. Lane, K. Yacef, J. Mostow, P. Pavlik (Eds.), *Artificial Intelligence in Education* (pp. 431-440). Berlin/Heidelberg: Springer.

Beck, J., Jia, P., & Mostow, J. (2003). Assessing student proficiency in a Reading Tutor that listens. In P. Brusilovski, A. Corbett, & F. de Rosis (Eds.), *International Conference on User Modeling* (pp. 323-327). Berlin/Heidelberg: Springer

Beck, J. E., Jia, P., Sison, J., & Mostow, J. (2003). Predicting student help-request behavior in an intelligent tutor for reading. In P. Brusilovsky, A. Corbett, & F. De Rosis (Eds.), *International Conference on User Modeling* (pp. 303-312). Berlin/Heidelberg: Springer.

Birenbaum, M., Kelly, A. E., & Tatsuoka, K. K. (1993). Diagnosing knowledge states in algebra using the rule-space model. *Journal for Research in Mathematics Education*, *24*(5), 442-459.

Bloom, B. S. (1984). The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring. *Educational Researcher*, *13*(6), 4-16.

Carbonell, J. R. (1970). AI in CAI: An artificial-intelligence approach to computer-assisted instruction. *IEEE Transactions on Man-Machine Systems, 11*(4), 190-202. doi:10.1109/TMMS.1970.299942

Cohen, P. A., Kulik, J. A., & Kulik, C. C. (1982). Educational outcomes of tutoring: A meta-analysis of findings. *American Educational Research Journal,19*(2)*,* 237-248

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

Cooper, D. G., Arroyo, I., Woolf, B. P., Muldner, K., Burleson, W., & Christopherson, R. (2009). Sensors model student self concept in the classroom. In G. J. Houben, G.

McCalla, F. Pianesi, & M. Zancanaro (Eds.), *User Modeling, Adaptation, and Personalization* (pp. 30-41). Berlin/Heidelberg: Springer.

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*(2), 81-108.

Craig, S., Graesser, A., Sullins, J., & Gholson, B. (2004). Affect and learning: An exploratory look into the role of affect in learning with AutoTutor. *Journal of Educational Media, 29*(3), 241-250. doi:10.1080/1358165042000283101

D'Mello, S., Jackson, T., Craig, S., Morgan, B., Chipman, P., White, H., ... & Graesser, A. (2008). AutoTutor detects and responds to learners affective and cognitive states. In *Workshop on Emotional and Cognitive Issues at the International Conference on Intelligent Tutoring Systems* (pp. 306-308). Montreal, Canada.

D'Mello, S., & Graesser, A. (2012). AutoTutor and affective AutoTutor: Learning by talking with cognitively and emotionally intelligent computers that talk back. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, *2*(4), 23-63.

Duolingo (n.d.) Retrieved October 21, 2017, from https://www.duolingo.com

Evens, M. W., Chang, R. C., Lee, Y. H., Shim, L. S., Woo, C. W., Zhang, Y., ... & Rovick, A. A. (1997). CIRCSIM-Tutor: An intelligent tutoring system using natural language dialogue. In *Proceedings of the fifth conference on Applied natural language processing: Descriptions of system demonstrations and videos* (pp. 13-14). Oxford, OH: Association for Computational Linguistics.

Feng, M., & Heffernan, N. T. (2006). Informing teachers live about student learning: Reporting in the ASSISTment system. *Technology Instruction Cognition and Learning, 3*(1/2), 63-77.

Freedman, R. (1997). Degrees of mixed-initiative interaction in an intelligent tutoring system. In *Working Notes of AAAI97 Spring Symposium on Mixed-Initiative Interaction* (pp. 44-49). Stanford, CA: AAAI.

Gertner, A. S., & VanLehn, K. (2000). Andes: A coached problem solving environment for physics. In G. Gauthier, C. Frasson, & K. VanLehn (Eds.), *International conference on intelligent tutoring systems* (pp. 133-142). Berlin/Heidelberg: Springer.

Goguadze, G., & Melis, E. (2008). Feedback in ActiveMath exercises. In *International Conference on Mathematics Education (ICME)* (pp. 1-7). ICME.

Goguadze, G., & Tsigler, J. (2007). Authoring interactive exercises in ActiveMath. In *Proceedings of the Mathematical User-Interfaces Workshop (MathUI-2007)* (pp. 1-10). Linz, Austria.

Graesser, A. C., Chipman, P., Haynes, B. C., & Olney, A. (2005). AutoTutor: An intelligent tutoring system with mixed-initiative dialogue. *IEEE Transactions on Education*, *48*(4), 612-618.

Graesser, A. C., Hu, X., Susarla, S., Harter, D., Person, N. K., Louwerse, M., & Olde, B. (2001). AutoTutor: An intelligent tutor and conversational tutoring scaffold. In *10th ICAI in Education* (pp. 47-49). San Antonio, TX: AIED.

Graesser, A. C., Moreno, K., Marineau, J., Adcock, A., Olney, A., Person, N. (2003). AutoTutor improves deep learning of computer literacy: Is it the dialog or the talking head?

Artificial Intelligence in Education: Shaping the Future of Learning through Intelligent Technologies, *97*, 47-55.

Graesser, A. C., VanLehn, K., Rosé, C. P., Jordan, P. W., & Harter, D. (2001). Intelligent tutoring systems with conversational dialogue. *AI Magazine*, *22*(4), 39-53.

Graesser, A. C., Wiemer-Hastings, K., Wiemer-Hastings, P., Kreuz, R., & Tutoring Research Group. (1999). AutoTutor: A simulation of a human tutor. *Cognitive Systems Research*, *1*(1), 35-51.

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.

Heffernan, N. T., Turner, T. E., Lourenco, A. L., Macasek, M. A., Nuzzo-Jones, G., & Koedinger, K. R. (2006). The ASSISTment builder: Towards an analysis of cost effectiveness of ITS creation. In G. Sutcliffe & R. Goebel (Eds.), *FLAIRS Conference* (pp. 515-520). Florida: AAAI.

Hoppcetal, H. C. (2003). Can automated questioning help children's reading comprehension? In J. Lester, R. Vicari, & F. Paraguacu (Eds.), *Intelligent Tutoring Systems: 7th International Conference* (pp. 490-502). Berlin/Heidelberg: Springer.

Jackson, T., Mathews, E., Lin, K. I., Olney, A., & Graesser, A. (2003). Modeling student performance to enhance the pedagogy of AutoTutor. In P. Brusilovsky, A. Corbett, & F. de Rosis (Eds.), *User Modeling 2003* (pp. 368-372). Berlin/Heidelberg: Springer.

Joseph, E. (2005). Engagement tracing: Using response times to model student disengagement. In C. Looi, G. McCalla, B. Bredeweg, & J. Breuker (Eds.), *Artificial Intelligence in*

Education: Supporting Learning through Intelligent and Socially Informed Technology, *125*, 88-95. The Netherlands: IOS Press Amsterdam.

Kodaganallur, V., Weitz, R. R., & Rosenthal, D. (2005). A comparison of model-tracing and constraint-based intelligent tutoring paradigms. *International Journal of Artificial Intelligence in Education, 15*(2), 117-144.

Koedinger, K. R., & Anderson, J. R. (1998). Illustrating principled design: The early evolution of a cognitive tutor for algebra symbolization. *Interactive Learning Environments*, *5*(1), 161-179.

Koedinger, K. R., McLaughlin, E. A., & Heffernan, N. T. (2010). A quasi-experimental evaluation of an on-line formative assessment and tutoring system. *Journal of Educational Computing Research, 43*(4), 489-510.

Lallé, S., Mostow, J., Luengo, V., & Guin, N. (2013). Comparing student models in different formalisms by predicting their impact on help success. In H. Lane, K. Yacef, J. Mostow, & P. Pavlik (Eds.), *International Conference on Artificial Intelligence in Education* (pp. 161-170). Berlin/Heidelberg: Springer.

Martin, B., Koedinger, K. R., Mitrovic, A., & Mathan, S. (2005). On using learning curves to evaluate ITS. In C. Looi, G. McCalla, B. Bredeweg, & J. Breuker (Eds.), *Proceedings of the 12th International Conference on Artificial Intelligence in Education* (pp. 419-428). The Netherlands: IOS Press Amsterdam.

Melis, E., & Andres, E. (2005). Global feedback in ActiveMath. *The Journal of Computers in Mathematics and Science Teaching, 24*(2), 197-220.

Melis, E., Andres, E., Budenbender, J., Frischauf, A., Goduadze, G., Libbrecht, P., … & Ullrich, C. (2001). ActiveMath: A generic and adaptive web-based learning environment. *International Journal of Artificial Intelligence in Education (IJAIED), 12*, 385-407.

Melis, E., Goguadze, G., Libbrecht, P., & Ullrich, C. (2009). ActiveMath—A learning platform with semantic web features. In D. Dicheva, R. Mizoguchi, & J. Greer (Eds.), *The Future of Learning* (pp. 159-177). Amsterdam IOS Press.

Melis, E., Moormann, M., Ullrich, C., Goguadze, G., & Libbrecht, P. (2007). How ActiveMath supports moderate constructivist mathematics teaching. In E. Milkova & P. Prazak (Eds.), *8th International Conference on Technology in Mathematics Teaching* (pp. 1-4), Czech Republic: University of Hradec Kralove.

Mendicino, M., Razzaq, L., & Heffernan, N. T. (2009). A comparison of traditional homework to computer-supported homework. *Journal of Research on Technology in Education, 41*(3), 331-359.

Mitrovic, A. (1997). SQL-Tutor: a preliminary report. http://hdl.handle.net/10092/2994

Mitrovic, A. (1998). Experiences in implementing constraint-based modeling in SQL-Tutor. In B. Goettl, H. Halff, C. Redfield, & V. Shute (Eds.), *Intelligent Tutoring Systems* (pp. 414-423). Berlin/Heidelberg: Springer.

Mitrovic, A. (2010). Modeling domains and students with constraint-based modeling. In R. Nkambou, J. Bourdeau, & R. Mizoguchi (Eds.), *Advances in Intelligent Tutoring Systems* (pp. 63-80). Berlin/Heidelberg: Springer.

Mitrovic, A., Martin, B., & Mayo, M. (2002). Using evaluation to shape ITS design: Results and experiences with SQL-Tutor. *User Modeling and User-Adapted Interaction*, *12*(2), 243-279.

Mitrovic, A., & Ohlsson, S. (1999). Evaluation of a constraint-based tutor for a database language. Retrieved from http://hdl.handle.net/10092/327

Mostow, J., & Aist, G. (1999). Giving help and praise in a reading tutor with imperfect listening—Because automated speech recognition means never being able to say you're certain. *CALICO Journal, 16*(3), 407-424.

Mostow, J., Beck, J., Chalasani, R., Cuneo, A., & Jia, P. (2002). Viewing and analyzing multimodal human-computer tutorial dialogue: a database approach. In *Proceedings of the 4th IEEE International Conference on Multimodal Interfaces* (pp. 129-139). Pittsburg, PA: IEEE.

Mostow, J., & Duong, M. (2009). Oral reading prosody. In V. Dimitrova, R. Mizoguchi, B.D. Boulay, and A. Graesser (Eds.), *Proceeding of the 14th International Conference on Artificial Intelligence in Education (AIED2009)* (pp. 189-196). Brighton, UK. IAIED.

Nkambou, R., Mizoguchi, R., & Bourdeau, J. (Eds.). (2010). *Advances in intelligent tutoring systems*. Berlin/Heidelberg: Springer.

Nwana, H. S. (1990). Intelligent tutoring systems: An overview. *Artificial Intelligence Review, 4*(4), 251-277.

Nye, B. D. (2015). AIED is splitting up (into services) and the next generation will be all right. *AIED Workshops, 1432*(4), 62-71.

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.

Ohlsson, S. (1996). Learning from performance errors. *Psychological Review*, *103*(2), 241-262.

Pane, J. F., Griffin, B. A., McCaffrey, D. F., & Karam, R. (2014). Effectiveness of cognitive tutor algebra I at scale. *Educational Evaluation and Policy Analysis*, *36*(2), 127-144.

Person, N. K., Craig, C., Price, P., Hu, X., Gholson, B., & Graesser, A. C. (2000). Incorporating human-like conversational behaviors into AutoTutor. In *Proceedings of the Workshop on Achieving Human-like Behavior in the Interactive Animated Agents at the Agents 2000 Conference* (pp. 85-92). Barcelona, Spain: The ACM Press.

Person, N. K., Graesser, A. C., Kreuz, R. J., & Pomeroy, V. (2003). Simulating human tutor dialog moves in AutoTutor. *International Journal of Artificial Intelligence in Education (IJAIED)*, *12*, 23-39.

Rajan, S., Craig, S. D., Gholson, B., Person, N. K., Graesser, A. C., & Tutoring Research Group. (2001). AutoTutor: Incorporating back-channel feedback and other human-like conversational behaviors into an intelligent tutoring system. *International Journal of Speech Technology*, *4*(2), 117-126.

Razzaq, L. M., Feng, M., Nuzzo-Jones, G., Heffernan, N. T., Koedinger, K. R., Junker, B., … & Rasmussen, K. (2005). Blending assessment and instructional assisting. In C. Looi, G. McCalla, B. Bredeweg, & J. Breuker (Eds.), *AIED* (pp. 555-562). The Netherlands: IOS Press Amsterdam.

Razzaq, L., & Heffernan, N. (2006). Scaffolding vs. hints in the ASSISTment System. In M. Ikeda, K. Ashley, & T. Chan (Eds.) *Intelligent Tutoring Systems* (pp. 635-644). Berlin/Heidelberg: Springer.

Razzaq, L. M., & Heffernan, N. T. (2009). To tutor or not to tutor: That is the question. In V. Dimitrova, R. Mizoguchi, B. du Boulay, & A. Graesser (Eds.), *Proceedings of the Conference on Artificial Intelligence in Education* (pp. 457-464). Amsterdam: IOS Press.

Razzaq, L., Heffernan, N. T., & Lindeman, R. W. (2007). What level of tutor interaction is best? *Frontiers in Artificial Intelligence and Applications*, *158*, 222-229.

Ritter, S., Anderson, J. R., Koedinger, K. R., & Corbett, A. (2007). Cognitive Tutor: Applied research in mathematics education. *Psychonomic Bulletin & Review*, *14*(2), 249-255

Ringenberg, M., & VanLehn, K. (2006). Scaffolding problem solving with annotated, worked-out examples to promote deep learning. In M. Ikeda, K. Ashley, T. Chan (Eds.), *Intelligent Tutoring Systems* (pp. 625-634). Berlin/Heidelberg: Springer.

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

Schulze, K. G., Shelby, R. N., Treacy, D. J., Wintersgill, M. C., VanLehn, K., & Gertner, A. (2000). Andes: An intelligent tutor for classical physics. *Journal of Electronic Publishing, 6*(1). Retrieved from http://dx.doi.org/10.3998/3336451.0006.110

Self, J. (1990). Theoretical foundations for intelligent tutoring systems. *Journal of Artificial Intelligence in Education, 1*(4), 3-14.

Singh, R., Saleem, M., Pradhan, P., Heffernan, C., Heffernan, N., Razzaq, L., & Dailey, M. (2011). Improving K-12 homework with computers. In *Proceedings of the Artificial Intelligence in Education Conference* (pp. 328-336). New York, NY: Springer.

Tvarožek, J., Kravčík, M., & Bieliková, M. (2008). Towards computerized adaptive assessment based on structured tasks. In W. Nejdl, J. Kay, P. Pu, & E. Herder (Eds.), *International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems* (pp. 224-234). Berlin/Heidelberg: Springer.

Uhr, L. (1969). Teaching machine programs that generate problems as a function of interaction with students. In *Proceedings of the 1969 24th National Conference* (pp. 125-134). New York, NY: ACM.

Ullrich, C. (2003). *Pedagogical rules in ActiveMath and their pedagogical foundations.* Univ. Fachbereich Informatik.

Ullrich, C., & Libbrecht, P. (2008). Educational services in the ActiveMath learning environment. In D. Dicheva, R. Mizoguchi, & J. Greer (Eds.), *The Future of Learning* (pp. 211-236). Amsterdam: IOS Press.

VanLehn, K. (1996). Conceptual and meta learning during coached problem solving. In C. Frasson, G. Gauthier, & A. Lesgold (Eds.), *Intelligent Tutoring Systems* (pp. 29-47). Berlin/Heidelberg: Springer.

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

VanLehn, K., Lynch, C., Schulze, K., Shapiro, J. A., Shelby, R., Taylor, L., … & Wintersgill, M. (2005). The Andes physics tutoring system: Lessons learned. *International Journal of Artificial Intelligence in Education, 15*(3), 147-204.

VanLehn, K., van de Sande, B., Shelby, R., & Gershman, S. (2010). The Andes physics tutoring system: An experiment in freedom. In R. Nkambou, J. Bourdeau, & R. Mizoguchi (Eds.), *Advances in Intelligent Tutoring Systems* (pp. 421-443). Berlin/Heidelberg: Springer.

Wenger, E. (1987). *Artificial intelligence and tutoring systems: Computational and cognitive approaches to the communication of knowledge.* Los Altos, CA: Morgan Kaufmann.

Wixon, M., Arroyo, I., Muldner, K., Burleson, W., Rai, D., & Woolf, B. (2014). The opportunities and limitations of scaling up sensor-free affect detection. In J. Stamper, Z.

Pardos, M. Mavrikis, & B. McLaren (Eds.), *Proceedings of the 7th International Conference on Educational Data Mining 2014* (pp. 145-152). London, UK: IEDMS.

Woo, C. W., Evens, M. W., Freedman, R., Glass, M., Shim, L. S., Zhang, Y., ... & Michael, J. (2006). An intelligent tutoring system that generates a natural language dialogue using dynamic multi-level planning. *Artificial Intelligence in Medicine*, *38*(1), 25-46.

Woo, C. W., Evens, M., Michael, J., & Rovick, A. (1991). Dynamic instructional planning for an intelligent physiology tutoring system. In *Computer-Based Medical Systems, 1991. Proceedings of the Fourth Annual IEEE Symposium* (pp. 226-233). IEEE.

Woods, P., & Hartley, J. R. (1971). Some learning models for arithmetic tasks and their use in computer based learning. *British Journal of Educational Psychology, 41*(1), 38-48.

Zhang, X., Mostow, J., Duke, N., Trotochaud, C., Valeri, J., & Corbett, A. (2008). Mining free-form spoken responses to tutor prompts. In R. Baker, T. Barnes, & J. Beck (Eds.), *Educational Data Mining 2008* (pp. 234-241). Montreal, Canada: IEDMS.

Zhou, Y., Freedman, R., Glass, M., Michael, J. A., Rovick, A. A., & Evens, M. W. (1999). Delivering hints in a dialogue-based intelligent tutoring system. In *AAAI/IAAI 1999* (pp. 128-134). Orlando, FL: AAAI.

ITS CASE STUDIES

ActiveMath

ActiveMath is unique among intelligent tutoring systems (ITSs) in that it is a mathematics e-textbook that is dynamically created for students. It has been in development since 2000 and has reached several countries in Europe as well as the US. Information used for this section comes from 17 articles ranging from 2001 to 2010.

Domain Model

What content does this ITS cover? ActiveMath is an ITS for mathematics (Andres, Fleisher, & Liang, 2010; Melis et al., 2001) with a special focus on the mathematical competencies from PISA (e.g., problem solving), an international student assessment (Ullrich, 2005). In contrast to many ITS which are created for one language and region, ActiveMath has been localized into several languages and regions. ActiveMath has been used to teach fractions in German; differential calculus in German, English, and Spanish; operations research in Russian and English; methods of optimization in Russian; statistics and probability in calculus in German (Ullrich & Libbrecht, 2008); and computer science in Chinese (Andres et al., 2010).

As an e-textbook, ActiveMath allows students to choose the concepts and topics they wish to study (Ullrich, 2003). Based on their choices, ActiveMath will then dynamically create a textbook that covers the chosen topics and any necessary prerequisites. Interactive activities and problems are included with the e-textbook.

What level of understanding is the goal? ActiveMath started out with mastery learning as its goal (Melis et al., 2001; Ullrich, 2003). However, mastery for each knowledge component was computed for three levels of Bloom's Taxonomy: knowledge, comprehension, and application (Melis et al., 2001). For example, a student who read a portion of the textbook

would gain knowledge mastery for that knowledge component. If a student followed along with a worked-out example, they would gain comprehension mastery. Finally, a student who successfully solves exercises would gain application mastery. ActiveMath was updated to be able to adapt to different competency frameworks based on the educational system (Melis, Goguadze, Libbrecht, & Ullrich, 2009).

What is used to create the expert? The expert knowledge associated with ActiveMath is constructed by means of a domain ontology and pedagogical ontology (Melis et al., 2009). The domain ontology lists knowledge components and contains information like definitions and theorems. The pedagogical ontology contains information like relationships between knowledge components (e.g., prerequisite knowledge) and properties of learning outcomes like difficulty.

Tutoring Model

Strategies. The developers of ActiveMath claim a moderate constructivist approach to mathematics teaching (Goguadze & Melis, 2008; Melis, Moormann, Ullrich, & Libbrecht, 2007). They describe this approach as one that leaves space for students to make their own choices while providing enough guidance. Students choose their own learning goals, the context, and learning scenario. ActiveMath then reacts to their choices by creating a personalized e-textbook for that student, which the student can extend when they see fit. Interactive exercises are incorporated into the content so that the student can engage in problem solving as well (Melis et al., 2007). As such, the developers of ActiveMath consider learning to be "active, self-guided, constructive, situated, and a social process" (Melis et al., 2007).

Within the moderate constructivist approach, a few notable theories were also used in designing the tutor. The first was Polya's heuristics (Melis & Ullrich, 2003; Polya, 1957). These heuristics were used in ActiveMath to scaffold problem solving and learning. In tandem

with Polya's approach, ActiveMath implemented Merrill's Instructional Transaction Theory (Melis & Ullrich, 2003; Merrill, 2002;) as a shell for organizing their learnings scenarios. ActiveMath's approach to feedback is informed by a feedback framework known as the Interactive Tutoring Feedback model (Goguadze & Melis, 2008; Narciss, 2004). This framework conceptualizes feedback as a way of regulating the learning process to help learners acquire the knowledge and skills they need for mastery (Goguadze & Melis, 2008). As such, feedback typically has two ingredients: an evaluation of being correct or incorrect and additional information that helps the student complete the exercise (Goguadze & Melis, 2008). Examples of strategies that are used to provide feedback include decomposing the problem into sub-goals, presenting a simpler version, and getting rid of feedback altogether and allowing students only to make correct steps or failing them on that exercise (Goguadze & Melis, 2008).

Tactics. To reify ActiveMath's approach to tutoring, this section will include the specific tactics ActiveMath implements when working with students.

Feedback. ActiveMath provides three levels of feedback: flag feedback, local feedback, and global feedback. With flag feedback, answers are marked as right or wrong; no additional information is given to the student unless they ask for a hint (Melis et al., 2009). Local feedback (also known as step-based feedback) helps the student through their problem-solving steps. Information is given to the student as soon as they complete each step of the process and should direct the student towards the correct solution (Melis & Andre, 2005). Global feedback, on the other hand, is less concerned with the problem-solving steps and even the current problem. Instead, global feedback helps to scaffold the overall learning process and may be given independently of problem-solving steps. For example, ActiveMath accomplishes this by giving feedback to the student to let them know what they ought to do. If a student incorrectly

answered a problem and had skipped reading a section of their e-textbook, then ActiveMath would instruct them to review the skipped material.

Learning Scenarios. ActiveMath implements six learning scenarios that students can choose from to accomplish their learning goals. These scenarios are called: LearnNew, Rehearse, Overview, TrainCompetency, Workbook, and ExamSimulation. The purpose of each scenario is unique, and the pedagogical rules are based on the scenario. For example, LearnNew provides an in-depth course that thoroughly covers all the target material. In contrast, Overview broadly covers the chosen content (Ullrich, 2005). The rules for each scenario can be changed at the teacher's discretion. This allows for the localization of ActiveMath, not just with the language, but also with the teaching styles of different countries. For example, ActiveMath's developers highlight the difference between American and German teaching styles. A German teaching style might have definitions and theorems come before exercises, whereas an American style would present exercises first (Melis et al., 2001).

Erroneous Examples. A unique feature of ActiveMath is that erroneous examples were included as a pedagogical tactic (Melis, 2005). Erroneous examples can be classified as erroneous results or erroneous worked solutions. In erroneous worked solutions, all the problem steps are shown, and a student can follow the logic. With erroneous results, only the result is given. As students are working with erroneous examples, students can be asked to either correct errors that are already marked or be asked to find the errors and then correct them (Melis, 2005). In other situations, the problem may not be presented as erroneous from the start and the student must determine if the solution is flawed.

Hinting. Like other ITSs, ActiveMath employs hint sequencing which scaffolds hints as they are presented to the student (Goguadze & Melis, 2008). When a student requests a hint, the

first hint starts off being more general and increases in specificity as additional hints are requested. No information was found that describes the number of hint levels typical for each exercise. It seems that the nature of hints is dependent on teacher discretion. Hints are annotated into problem-solving steps and are dependent on the problem at hand (Goguadze & Tsigler, 2007).

Student Model

How is the student model created? ActiveMath seems to be using model tracing and knowledge tracing to create the student model. An example of model tracing in ActiveMath is the domain reasoners which follows students' problem-solving steps through branched decision paths. As a student follows one solution path, ActiveMath can determine where along the path a student is and then offer feedback based on its estimation of what the student is trying to accomplish (Goguadze, 2009). Using the same ontology, ActiveMath attaches competency values to each knowledge component. This process is called knowledge tracing. As a student interacts with ActiveMath, the techniques Item Response Theory and Transferable Belief Model (Faulhaber & Melis, 2008) are used to update the competencies based on the type of action the student performed (e.g., solving a problem vs. reading text) and other metadata (e.g., number of hints seen, whether the solution was correct or incorrect) (Melis et al., 2009).

What data is collected? To begin with, ActiveMath uses their registration page to collect information about the student in order to initialize the student model. The registration page collects information like which learning scenario the student wants to use as well as their goals for the session. Students are also asked to self-assess the amount of information they know regarding the concepts related to their learning goal (Melis et al., 2001). ActiveMath even customizes instruction based on the field and education level identified at registration (Ullrich,

2003). For example, an economics major at the university level studying statistics would benefit from different examples than a mathematics major studying the same subject.

As students interact with ActiveMath, additional information is collected to build the student model. Timestamps are used to denote when a student starts/finishes reading a page, starts/finishes an exercise, and when the problem was correctly or incorrectly solved (Melis & Andres, 2005). These data can provide information to the system such as the presentation order of material and exercises. For example, if a student read sections A and C of a concept, but skipped section B, ActiveMath would be able to detect that. Then, if a student incorrectly solves a problem that related to information in section B, ActiveMath can suggest to the student to go back and more thoroughly read the section that was skipped.

ActiveMath even attempts to collect data on and model students' focus. To model students' attention, ActiveMath implemented a "poor man's eyetracker" (Melis & Ullrich, 2002). More sophisticated eyetrackers use cameras to track where the eye is looking to track attention. ActiveMath's developers created a simpler version which they believe accomplishes the same thing. The underlying premise is that mouse location can determine where the attention of the student is. However, not all students use the mouse to track where their attention is.

So ActiveMath developed a method of softly forcing students to keep track with their mouse by making the text slightly unreadable (Melis & Ullrich, 2002). Three variants exist to implement the poor man's eyetracker. First, elements of the page that do not have the focus of the mouse are covered, meaning a black box covers the text. Second, a fade option exists which fades unfocused content to a color that is similar to the background color. Third, elements on the page that do not have focus are zoomed out, meaning that the font size is changed to something smaller. In order to restore the information to a readable state, the student must move their

mouse over that content. Once they do, a timer is started after a latent period to keep track of where a student's focus is. This information can then be used in the student model since the tutor can now determine which specific units the student paid attention to on a single page thus helping to inform the tutor how much to increase mastery levels by for the competencies of concepts.

Interface

ActiveMath is a web-based (Goguadze, 2009), dynamically created textbook that includes content and exercises for students along with explanations, visualizations, and interactive exercises (Andres et al., 2010). Student responses are entered via mouse and keyboard, sometimes typing in their answers and solution steps (Goguadze & Melis, 2008) and other times choose answers to multiple choice questions. No research was found that explores students' experience with ActiveMath.

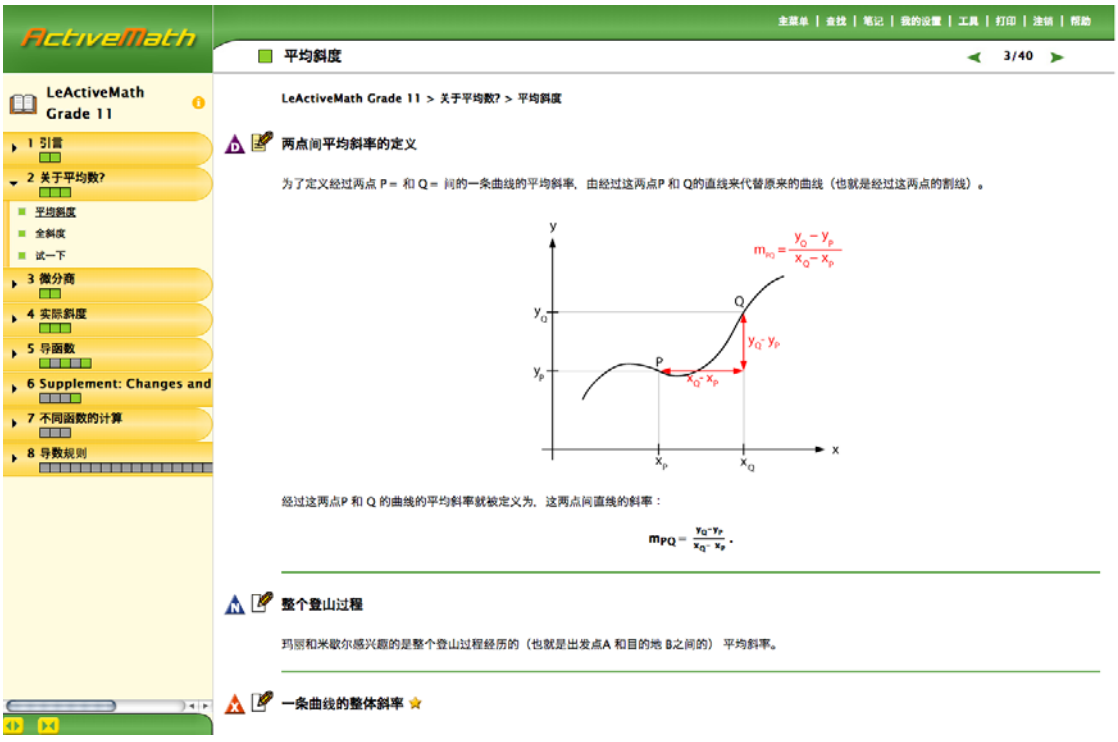

Figure 1. The interface of ActiveMath (Andres et al., 2010).

Learning Gains

No research was found that describes any learning gains that were measured when using ActiveMath.

References

Andres, E., Fleischer, R., & Liang, M. (2010). An adaptive theory of computation online course in ActiveMath. In *Computer Science and Education (ICCSE), 2010 5th International Conference on* (pp. 317-322). IEEE.

Faulhaber, A., & Melis, E. (2008). An efficient student model based on student performance and metadata. *ECAI, 178*, 276-280.

Goguadze, G. (2009). Semantic evaluation services for web-based exercises. In M. Spaniol, Q. Li, R. Klamma, & R. Lau (Eds.), *Advances in Web Based Learning–ICWL 2009* (pp. 172-181). Berlin/Heidelberg: Springer.

Goguadze, G., & Melis, E. (2008). Feedback in ActiveMath exercises. In *International Conference on Mathematics Education (ICME)* (pp. 1-7). ICME.

Goguadze, G., & Tsigler, J. (2007). Authoring interactive exercises in ActiveMath. In *Proceedings of the Mathematical User-Interfaces Workshop (MathUI-2007)* (pp. 1-10). Linz, Austria.

Melis, E. (2005). Design of Erroneous Examples for ACTIVEMATH. In *Proceedings of the 2005 Conference on Artificial Intelligence in Education* (pp. 451-458). Amsterdam, The Netherlands: IOS Press.

Melis, E., & Andres, E. (2005). Global feedback in ActiveMath. *The Journal of Computers in Mathematics and Science Teaching, 24*(2), 197-220.

Melis, E., & Ullrich, C. (2002). The poor man's eyetracker tool of ActiveMath. In *E-Learn: World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education* (pp. 2313-2316). Chesapeake, VA: AACE.

Melis, E., & Ullrich, C. (2003). How to teach it-Polya-inspired scenarios in ActiveMath. In U. Hoppe, F. Verdejo, & J. Kay (Eds.), *Proceedings of the International Conference on AI in Education AIED* (pp. 141-147). Sydney, Australia: IOS Press.

Melis, E., Andres, E., Budenbender, J., Frischauf, A., Goduadze, G., Libbrecht, P., … & Ullrich, C. (2001). ActiveMath: A generic and adaptive web-based learning environment. *International Journal of Artificial Intelligence in Education (IJAIED), 12*, 385-407.

Melis, E., Goguadze, G., Libbrecht, P., & Ullrich, C. (2009). ActiveMath–a learning platform with semantic web features. In D. Dicheva, R. Mizoguchi, & J. Greer (Eds.), *The Future of Learning* (pp. 159-177). Amsterdam IOS Press.

Melis, E., Moormann, M., Ullrich, C., Goguadze, G., & Libbrecht, P. (2007). How ActiveMath supports moderate constructivist mathematics teaching. In E. Milkova & P. Prazak (Eds.), *8th International Conference on Technology in Mathematics Teaching* (pp. 1-4)*,* Czech Republic: University of Hradec Kralove.

Merrill, M. D. (2002). First principles of instruction. *Educational Technology Research and Development*, *50*(3), 43-59.

Narciss, S. (2004). The impact of informative tutoring feedback and self-efficacy on motivation and achievement in concept learning. *Experimental Psychology*, *51*(3), 214-228.

Polya, G. (1957). *How to solve it: A new aspect of mathematical method*. Princeton, NJ: Princeton University Press.

Ullrich, C. (2003). *Pedagogical rules in ActiveMath and their pedagogical foundations.* Univ. Fachbereich Informatik.

Ullrich, C. (2005). Course generation based on HTN planning. In A. Jedlitschka & B. Brandherm (Eds.), *Proceedings of the 13th Annual Workshop of the SIG Adaptivity and User Modeling in Interactive Systems* (pp. 74-79). Saarbrucken, Germany.

Ullrich, C., & Libbrecht, P. (2008). Educational services in the ActiveMath learning environment. In D. Dicheva, R. Mizoguchi, & J. Greer (Eds.), *The Future of Learning* (pp. 211-236). Amsterdam: IOS Press.

ANDES

The developers of ANDES claim that ANDES was the first model-tracing tutor for the science domain. Created and used at the US Naval Academy, ANDES was developed to mimic paper-and-pencil homework albeit in a digital format so that intelligent tutoring could be given to the student (Schulze et al., 2000b). The developers stated that the goal of ANDES is "to demonstrate that intelligent tutoring can be decoupled from content reform and yet still improve learning" (VanLehn et al., 2005b, p. 2). Seventeen articles spanning from 1996-2010 were used for this review.

Domain Model

What content does this ITS cover? Andes is an ITS that covers classical Newtonian mechanics at the university level (VanLehn, 1996). Topics covered include static forces, translational and rotational kinematics, translational and rotational dynamics, energy, and linear and angular momentum (Schulze et al., 2000c).

What level of understanding is the goal? ANDES started off with mastery learning as its goal, stating that a "student knows a rule when she is able to apply it correctly whenever it is required to solve a problem in all possible contexts," (Conati, Gertner, VanLehn, & Druzdzel, 1997, p. 235). However, ANDES is not designed to teach physics all by itself. It still expects students to attend classes and read textbooks. ANDES is meant to be a homework aid that provides practice problems, which are scaffolded with tutoring (Conati, Gertner, & VanLehn, 2002).

Even though mastery was the initial goal of ANDES, as the system evolved and was regularly used, there was a discrepancy between what the instructors wanted and how ANDES operated (VanLehn, Van De Sande, Shelby, & Gershmann, 2010). The instructors could not

adapt their courses to allow students to progress through the course as they wanted and so the adaptive learning process that comes with mastery learning was not used. Additionally, instructors did not use the student model for grading. As such, the mastery learning model was done away with and ANDES provided tutoring on problems chosen by the teacher.

What is used to create the expert? The expert for ANDES was created by three physics professors who were the domain and pedagogical experts (Conati et al., 1997). Together, they created over 600 physics rules that governed ANDES' behavior (Schulze et al., 2000b). However, these rules do not teach things that experts do since there is no expert consensus on how to solve problems (VanLehn et al., 2005b). Instead, ANDES focuses on a conceptual understanding of physics since experts can reach consensus on the major principles needed to solve problems.

ANDES uses the rules to create a solution graph which produced all the acceptable solution paths for a problem since multiple solution paths can exist (Schulze et al., 2000b). Doing so required too much time and was too slow as the program was more regularly used. Now, ANDES just evaluates the equation by replacing variables with numbers so as to provide feedback and hints to students (Shapiro, 2005).

Tutoring Model

Strategies. ANDES uses constructivism as its underlying strategy for teaching students. Specifically, coached problem solving is used, which is a kind of cognitive apprenticeship that is limited to solving problems while providing adaptive scaffolding (Ringenberg & VanLehn, 2006; VanLehn, 1996). Cognitive skill acquisition is viewed as having three phases. First, students must acquire preliminary knowledge. Second, students learn to apply knowledge by applying existing knowledge and learning new knowledge. Third, students practice until they've

reached automaticity, in other words, a state where minimal cognitive effort is required to perform the task (VanLehn, 1996).

The developers of Andes identified the following fundamental principles, which guided the design and strategies of the system (Gertner & VanLehn, 2000):

1. Encouraging students to construct their own knowledge by providing hints that require them to reach the solution on their own.
2. Making the interface as much like paper and pencil to promote transfer.
3. Giving immediate feedback to maximize the opportunities for learning and minimize the amount of time pursuing incorrect solution paths.
4. Allowing students to perform and skip steps as they please.

Tactics. This section will talk about the tutoring tactics of ANDES, specifically its feedback, fading, hints, algebra solver, mini-lessons, and encouragement of good behaviors.

Feedback. When a student answers a question incorrectly, they are immediately given flag feedback which simply flags the problem step as correct or incorrect by marking it green or red respectively (VanLehn, 1996). If a student wants additional help, they click on a "What's Wrong?" button which will give a feedback message to the student. To determine the feedback given to the student, ANDES initially compared the student's solution state to the solution graph (Schulze et al., 2000c). When a match was found, feedback associated with that state in the solution graph could be given to the student. After discontinuing the practice of generating all possible solution states beforehand, ANDES switched to an evaluative model which substituted numbers for variables to see if the equation balances. If it does, then the solution state is marked as green, otherwise it is marked as red (VanLehn et al., 2010).

Fading. Two forms of scaffolding were present in ANDES (VanLehn, 1996). The first, which was already discussed, was the flag feedback mechanism which is eventually supposed to be faded. The second form relates to the user interface for qualitative analysis which, too, is faded with time. Here, students are, at first, required to enter qualitative analyses of the problems they are given until it is faded where they are no longer required to do so. This fading process is meant to parallel the final phase of a cognitive apprenticeship, or in this case, coached problem solving.

Hints. As one of the fundamental principles above stated, hints are meant to help the student to solve a problem on their own rather than giving them the answer (Conati et al., 1997). Hints are sequenced from general to specific until a bottom-out hint is given. Generally, there are three levels of hints (pointing hints, teaching hints, and bottom-out hints; VanLehn et al., 2005b). Pointing hints identify where the problem is, teaching hints provides remedial instruction, and bottom-out hints provide an answer. Experiments with this hint sequence have included replacing teaching hints with multimedia or natural language dialogue which reportedly improved learning in a lab setting (VanLehn et al., 2005b). Another experiment replaced the bottom-out hint with worked examples (Ringenberg & VanLehn, 2006).

Students can receive hints in three situations (VanLehn et al., 2005a). First, when the system determines that a student slipped and made an error due to lack of attention, an error message pops up. Second, when the mistake was not a slip, then the flag feedback occurs, and students can receive help via the "What's Wrong?" button. Last, when a student is stuck and does not know what to do next, they may click on the "Next Step" button. "What's Wrong?" help follows the pattern described in the paragraph above (pointing, teaching, and bottom-out hints). "Next Step" help differs since it is not in response to a student mistake. The system

attempts to recognize the student's plan via the solution graph and then suggested the next step. However, when they compared ANDES' suggested action path with experts' suggested paths, a discrepancy was found (VanLehn et al., 2010). In response, ANDES replicated the experts' plan and suggests to the students a particular problem-solving strategy. From there, ANDES guides the student through the steps of that strategy. Students maintain the freedom to abort the current path at any time and do their own thing.

Algebra Solver. With ANDES' focus being on providing a conceptual understanding of physics and less on the mathematical problem solving associated with it, the developers provide an algebra solver that removes some of the cognitive burden from the student (Schulze et al., 2000a). This frees students to focus on the concepts being taught rather than their algebraic skills.

Mini-lessons. When ANDES determines that a student does not understand a specific concept, a short, annotated lesson appears in a new window. This is regardless of whether a student asks for the help. A student cannot continue working on the current problem until they've finished the mini-lesson or dismiss the lesson by closing the window (Schulze et al., 2000b).

Encouraging good behaviors. As many ITSs have experienced, students engage in gaming behavior with ANDES as well. Gaming behavior is when students minimize their effort to maximize the outcome. Each tutoring system develops its own way of dealing with gaming behaviors. ANDES' solution was to display an overall score in the lower right corner (VanLehn et al., 2010). As students solved steps correctly and without help, they would gain points. However, if a student engages in gaming behavior or bad physics practices, then they would lose points.

Student Model

How is the student model created? In the beginning, ANDES used a probabilistic framework for its student model (Conati et al., 1997). This framework accomplished three goals, two of which have already been discussed: recognizing the student's current solution path and which goals and actions the student is trying to pursue via model tracing. The third goal provides long-term assessment of the domain knowledge via knowledge tracing. However, as was previously mentioned, ANDES eliminated most of its student model once it was realized that the professors using the system did not want it to be used for macro-adaptation (individualized sequencing of content; VanLehn et al., 2005b) and mastery learning wasn't used for grading criteria (VanLehn et al., 2010). Thus, it does not appear that knowledge tracing was used.

What issues are there regarding the development of the student model? While ANDES still had a student model, an issue associated with knowledge tracing was brought up, which was how to assign credit or blame to knowledge components and solution paths (Conati et al., 2002). That is, if there are multiple explanations for a student's solution, then which one receives the credit or the blame? Earlier solutions to this problem included restricting student input to one solution path, however the professors involved with the design of ANDES insisted that all possible solution paths be viable. ANDES' developers addressed this issue by using the estimate of what a student knows to influence the prediction of which student paths a student is likely to take. In other words, if a student is more likely to know the skills associated with solution path A over solution path B, yet both solution paths could explain the current solution state, the tutor can make the decision that path A is more likely since the student knows those

skills. This makes ANDES the first ITS that modeled knowledge tracing, plan recognition, and action prediction at the same time (Conati et al., 2002).

What data is collected? Since the developers of ANDES abandoned the probabilistic portions of its student model (e.g., likeliness of knowing a knowledge component), not much was reported in terms of the data ANDES collects. From what was reported, it seems that ANDES attempted to track attention in a modified version of ANDES that focused on self-explanation by masking parts of the interface (Conati et al., 2002).

As was mentioned previously, ANDES computes a score that is displayed at the bottom of the screen. This score is a weighted sum of data including the proportion of correct entries and the number of requests for bottom-out hints (VanLehn et al., 2005b; VanLehn et al., 2010). So, while it's apparent that student data still has uses within the ANDES system, not much is reported on the nature of the data collected.

Interface

ANDES's interface is referred to as the ANDES Workbench and is meant to replicate paper and pencil—thus becoming a substitute for traditional homework (VanLehn et al., 2005b). The upper right panel of the interface holds variables and their definitions which students are required to fill out. The bottom right panel acts as a sheet of paper with separate lines available for the student to type in their problem-solving steps line by line. The current problem is displayed on the left. The student has various commands at their disposal to do things like request for hints, solve algebraic expressions, use a Greek keyboard, and use buttons that assist in drawing vectors and free-body diagrams (Schulze et al., 2000a). While ANDES started as a desktop application (VanLehn, 1996), it eventually transitioned fully to the web (VanLehn et al., 2010).

Initial student reactions to ANDES showed that students were irritated at first for being forced to enter and define variables especially on easy problems. The developers of ANDES claim that students eventually adapted and recognized the utility of being forced to define variables. These student responses were anecdotal (Schulze et al., 2000c). With more experiments being done with ANDES, additional student reactions were collected. One study found that those students who used ANDES a lot like it while those who did not use ANDES as much were less partial to the system (Schulze et al., 2000b). The same study also suggested that a student should be given a difficult problem when first using the system to help them appreciate the functionality of ANDES. However, students reported that 40-50% of them like the system and nearly 70% said ANDES was more effective than other tools at the university (VanLehn et al., 2005b).

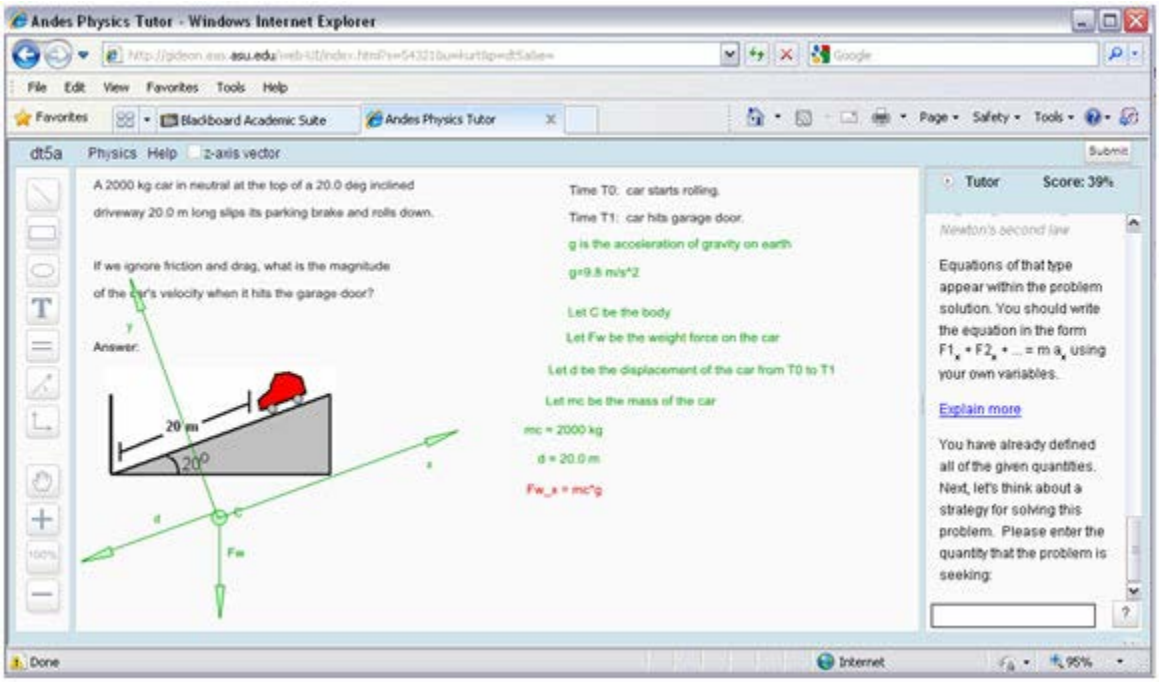

Figure 1. The interface of the web-based version of ANDES (VanLehn et al., 2010).

Learning Gains

ANDES conducted several studies which measured the learning gains produced by the system. Each study used students from multiple sections of a physics course and each student took the same final exam. However, each professor assigned different homework problems and midterms. Three professors in this study used ANDES instead of traditional homework. The other professors who took part in the study served as control groups. The homework problems assigned in the control group were similar to the problems assigned in ANDES while the midterms and final exam were the same for both treatment and control. No pretest was administered to determine if students had similar knowledge, but the groups were considered equal and randomly distributed based on GPA and college major (VanLehn et al., 2010).

Overall, the developers of ANDES reported having an average effect size of 0.61 (VanLehn et al., 2010). Effect sizes for the following years were also reported for the midterms: 1999: 0.21; 2000: 0.92; 2001: 0.52; 2002: 0.44; 2003: 0.6. ANDES did not cover much of the content of the final exam until 2003 when it covered 70% of the material. Based on the data from that year, the developers reported an effect size of 0.25 for all student. When broken down by major, engineering and science students showed no significant difference between treatment and control while other majors showed an effect size of 0.5 (VanLehn et al., 2010)

References

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

Conati, C., Gertner, A. S., VanLehn, K., & Druzdzel, M. J. (1997). On-line student modeling for coached problem solving using Bayesian networks. In A. Jameson, C. Paris, C. Tasso (Eds.), *User Modeling* (pp. 231-242). Vienna: Springer.

Gertner, A. S., & VanLehn, K. (2000). Andes: A coached problem solving environment for physics. In G. Gauthier, C. Frasson, & K. VanLehn (Eds.), *International conference on intelligent tutoring systems* (pp. 133-142). Berlin/Heidelberg: Springer.

Ringenberg, M., & VanLehn, K. (2006). Scaffolding problem solving with annotated, worked-out examples to promote deep learning. In M. Ikeda, K. Ashley, T. Chan (Eds.), *Intelligent Tutoring Systems* (pp. 625-634). Berlin/Heidelberg: Springer.

Shapiro, J. A. (2005). An algebra subsystem for diagnosing students' input in a physics tutoring system. *International Journal of Artificial Intelligence in Education*, *15*(3), 205-228.

Schulze, K. G., Shelby, R. N., Treacy, D. J., Wintersgill, M. C., VanLehn, K., & Gertner, A. (2000a). Andes: A coached learning environment for classical Newtonian physics. In G. Gautheier, C. Frasson, & K. VanLehn (Eds.) *Intelligent Tutoring Systems: 5th International Conference, ITS 2000* (pp. 133-142). New York: Springer.

Schulze, K. G., Shelby, R. N., Treacy, D. J., Wintersgill, M. C., VanLehn, K., & Gertner, A. (2000b). Andes: An active learning, intelligent tutoring system for Newtonian physics. *THEMES in Education*, *1*(2), 115-136.

Schulze, K. G., Shelby, R. N., Treacy, D. J., Wintersgill, M. C., VanLehn, K., & Gertner, A. (2000c). Andes: An intelligent tutor for classical physics. *Journal of Electronic Publishing, 6*(1). Retrieved from http://dx.doi.org/10.3998/3336451.0006.110

VanLehn, K. (1996). Conceptual and meta learning during coached problem solving. In C. Frasson, G. Gauthier, & A. Lesgold (Eds.), *Intelligent Tutoring Systems* (pp. 29-47). Berlin/Heidelberg: Springer.

VanLehn, K., Lynch, C., Schulze, K., Shapiro, J. A., Shelby, R., Taylor, L., ... & Wintersgill, M. (2005a). *The Andes physics tutoring system: Five years of evaluations*. Annapolis, MD: Naval Academy.

VanLehn, K., Lynch, C., Schulze, K., Shapiro, J. A., Shelby, R., Taylor, L., … & Wintersgill, M. (2005b). The Andes physics tutoring system: Lessons learned. *International Journal of Artificial Intelligence in Education, 15*(3), 147-204.

VanLehn, K., van de Sande, B., Shelby, R., & Gershman, S. (2010). The Andes physics tutoring system: An experiment in freedom. In R. Nkambou, J. Bourdeau, & R. Mizoguchi (Eds.), *Advances in Intelligent Tutoring Systems* (pp. 421-443). Berlin/Heidelberg: Springer.

Aplusix

Aplusix is an ITS that was developed and used in various parts of Europe to teach Algebra (Chaachoua, Nicaud, Bronner, & Bouhineau, 2004). It was designed with two goals in mind (Hadjerrouit & Bronner, 2014). First, to create a computer-based environment where students can write and transform algebraic expressions much like they do on paper. Second, to provide appropriate feedback to students.

Domain Model

What content does this ITS cover? Aplusix is an ITS that covers several topics within Algebra (Nicaud, 1993). Topics include algebra and numerical calculation, expansion and simplification, factorization, solving equations, solving inequalities, solving systems of equations and inequalities (Dagami et al., 2011).

What level of understanding is the goal? Nothing was found in the articles relating to this question.

What is used to create the expert? Nothing was found in the articles relating to this question.

Tutoring Model

Strategies. The developers of Aplusix adopt a constructivist approach to their tutoring (Chaachoua et al., 2004). Relying on the Theory of Didactical Situations (TDS), they claim that "learning occurs by means of interaction between learner and a 'milieu'" (Hadjerrouit & Meier, 2010, p. 3433). The milieu is an educational environment with which students interact. As students perform actions in the milieu, the milieu generates feedback for the student which then causes the student to experience difficulties and contradictions. As students learn to adapt to the feedback, this can be considered evidence of learning. However, Aplusix developers are clear on

their emphasis that teachers are still responsible for the teaching and learning process. Aplusix is only meant to help students apply the rules and methods correctly, thus narrowing their constructivist approach to one of learning by doing (Nicaud, 1993).

Three major principles were reported that seem to guide not only tutoring decisions, but also the interface design as well (Nicaud, Bouhineau, Chaachoua, Huguet & Bronner, 2003). First, a student will show their work by entering the results of their calculation steps. Second, non-intrusive feedback is given to the student. Third, some commands will be available for the system to perform routine calculations so that the student can focus on higher-order concepts.

Tactics. The tutoring tactics of Aplusix are discussed next, in particular: feedback, hints, and modes of instruction.

Feedback. Flag feedback is implemented in Aplusix (Nicaud et al., 2003). Flag feedback is used to check the structure or syntax of the algebra as well as semantic equivalence. So, if a step is well-structured, the step's font is colored black. If the syntax is entered incorrectly, it is highlighted blue. Syntax errors that result in undefined being returned (like dividing by zero) are turned red. Equivalence between problem-solving steps is also flagged. If two steps are equivalent, black arrows are used to represent this. If the steps are not equivalent, then a red "X" is used. Finally, students are alerted when they have reached a solved state. However, students are free to end when they think they are done since there are multiple solution paths, and help functions do not place a student on a solution path (Rodrigo et al., 2008; Nicaud, 1993).

Hints. Following the pattern of other ITSs, Aplusix implements hint sequences that end in bottom-out hints (Rodrigo, Baker, et al., 2008). However, students can only see the bottom-out hint after submitting their answers and receiving feedback on the errors they've already

committed. These error messages do not indicate which part of the equation is wrong, but the student is left to figure that out (Rodrigo, Anglo, et al., 2008).

Modes of instruction. The pedagogical goals of Aplusix can be adapted based on the selected mode of instruction. Four modes exist within Aplusix (Hadjerrouit & Meier, 2010): training mode, test mode, self-correction, and observation. Each mode differs in the amount of help and scaffolding present. For example, training mode uses scaffolding (showing equivalence between steps) to help the students learn the material. However, test mode would be devoid of any hints or scaffolding in order to simulate a classroom test.

Student Model

How is the student model created? Aplusix, at one point, had 260 rules that captured both conceptions and misconceptions a student has (Nicaud, Chaachoua, & Bittar, 2006). Based on their definition of how the rules worked, it sounds similar to how model and knowledge tracing work. For example, as students transform equations, the rules check to see if a student properly applied the transformation (model tracing). Knowledge tracing works as conceptions or misconceptions are attributed to a student when several behaviors verify that the student has that conception or misconception. No sophisticated, probabilistic models were reported as of 2006 (Nicaud et al., 2006) to determine when the behaviors verify that a student has a conception or misconception.

In addition to this cognitive model of what a student knows, work has been done to create non-cognitive models that measure student affect (Anglo & Rodrigo, 2010). Using system interactions, semantic data, and naturalistic observations, the developers of Aplusix attempted to model boredom, confusion, and frustration. Unfortunately, the results were subpar and could barely detect student affect better than chance (Anglo & Rodrigo, 2010). Another study

attempted to measure engaged concentration, boredom, confusion, delight, surprise, frustration, and a neutral state, but produced poor models that only confirmed the importance of the average number of steps a student takes to solve a problem in determining student affect (Andallaza, Rodrigo, Lagud, Jimenez, & Sugay, 2012).

What data is collected? One article listed the variables collected in Aplusix's log file (Dagami et al., 2011). Content of the log files includes: school, run (the student's run or batch number), student number (id), set number (set number of current exercise), problem number within the set (item number within the set number), absolute problem number (the item number relative to all the problems answered by the student), date, time started, level (degree of difficulty), step number (the step number which is then number of the current step), duration (the number of seconds describing how long each step was done), action (the action performed by the student), error (the error committed by the student while solving problem), etape (the stage or phase of the solution), expression (the state of the math expression), etat (the current state of the solution), cursor (location of cursor), selection (selected values in the solution), equivalence (indicates whether the equation is correct or not), and resolution (indicates whether the problem has been solved or not).

Aplusix makes use of this data to inform interventions like it does with its embodied conversational agent (ECA) called Grimace who responds to students' affective states (Andallaza & Rodrigo, 2013). Grimace used two features (the number of steps and the duration) to determine a student's affective state. Using these features and threshold values for each, Grimace classified students into three categories (high-performing, average, and low-performing). The classification determined the affective state for a student. So those who were classified as high-performers could be considered engaged, while those who were average were

confused, and the low-performers were considered bored. Grimace would then give feedback to the students based on their affective state.

Interface

Aplusix is considered to be an "advanced 2D editor of algebraic expressions" (Nicaud et al., 2003, p. 1). A student uses both a physical and a virtual keyboard to enter in their mathematical equations into the system, but some tools are provided that solve simpler steps for students so they can focus on more advanced concepts (Chaachoua et al., 2004). A student is not restricted in making only correct transformations of an equation, but are free to do as they will (Hadjerrouit & Bronner, 2014).

Aplusix has two forms of pedagogical agents. First, there are domain-based agents named Chloe, Julien, and Olivia who provide feedback and hints to students about the problems being solved (Andallaza et al., 2012). The other agent, Grimace, was mentioned previously. Grimace detects and responds to student affect by using an appropriate expression and providing both text and spoken feedback to the student (Andallaza et al., 2012). Research on Grimace's effectiveness showed that responses were too quick and frequent which irritated some students (Andallaza et al., 2012). These findings prompted a newer version of Grimace that addressed the quick and frequent responses. The newer version of Grimace was better received as students expressed an overall, more positive trend on survey questions towards the newer version of Grimace when compared to a control group that used the old version of Grimace (Andallaza & Rodrigo, 2013).

Aplusix is unique in that it does not provide a web-based version of its tutor. It must be purchased and downloaded onto each computer (Hadjerrouit & Bronner, 2014). This paywall and lack of web-based support caused some teachers to be dissatisfied with the accessibility of

Aplusix since students could not work with the system at home (a mean score of 3.43 on a 5-point Likert scale with 1 being highest and 5 being the lowest; Hadjerrouit & Bronner, 2014). The same group of teachers were not overly impressed with the interface in terms of attractiveness, visual design, and appearance demonstrated by an average score of 2.86 for a survey question relating to those characteristics (Hadjerrouit & Bronner, 2014). These and other usability concerns have prevented Aplusix from being "fully successful in classroom [sic]" (Hadjerrouit & Bronner, 2014, p. 1).

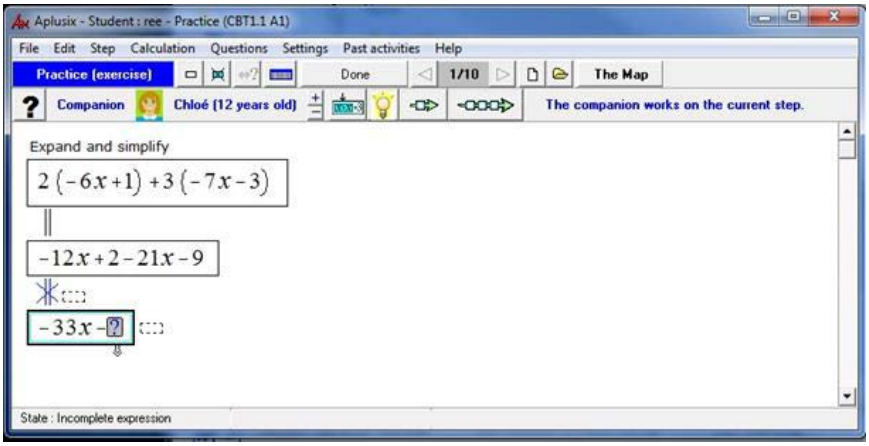

Figure 1. The user interface of Aplusix (Andallaza et al., 2012).

Learning Gains

No controlled experiments comparing use of Aplusix was found. Instead, gains were reported in terms of pre-/post-test scores within Aplusix (Chaachoua et al., 2004; Hadjerrouit & Meier, 2010). One study showed that Aplusix can increase learning on its own pre-/post-tests by 70% to 250% (Bouhineau, Nicaud, Chaachoua, Bittar, & Bronner, 2005). These data were collected over two years of use and four experiments. The experiments ranged from having two to three sessions with Aplusix with each session being 50 minutes. Test problems covered linear equations and inequations for all but one of the experiments which had test questions relating to

every type of problem available in Aplusix. A total of 2874 students were involved ranging from 13-17 years of age.

References

Andallaza, T. C. S., & Rodrigo, M. M. T. (2013). Development of an affect-sensitive agent for Aplusix. In H. Lane, K. Yacef, J. Mostow, & P. Pavlik (Eds.), *International Conference on Artificial Intelligence in Education* (pp. 575-578). Berlin/Heidelberg: Springer.

Andallaza, T. C. S., Rodrigo, M. M. T., Lagud, M. C. V., Jimenez, R. J. M., & Sugay, J. O. (2012). Modeling the affective states of students using an intelligent tutoring system for algebra. In *Proceedings of The Third International Workshop on Empathic Computing (IWEC 2012)* (pp. 12-21). Malaysia: IWEC.

Anglo, E. A., & Rodrigo, M. M. T. (2010). Can affect be detected from intelligent tutoring system interaction Data?–A preliminary study. In V. Aleven, J. Kay, & J. Mostow (Eds.), *International Conference on Intelligent Tutoring Systems* (pp. 260-262). Berlin/Heidelberg: Springer.

Bouhineau, D., Nicaud, J. F., Chaachoua, H., Bittar, M., & Bronner, A. (2005). Two years of use of the Aplusix system. In *8th IFIP World Conference on Computer in Education* (pp. 23-31). Cape Town, South Africa: IFIP.

Chaachoua, H., Nicaud, J. F., Bronner, A., & Bouhineau, D. (2004). Aplusix, a learning environment for algebra, actual use and benefits. In *ICME 10: 10th International Congress on Mathematical Education* (pp. 8-16). Copenhagen, Denmark: ICME.

Dagami, M. M. C., Guo, B., Pating, M. B., Guia, T. F. G., Leonor, W. B. S., & Rodrigo, M. M. T. (2011). Determining the precursors of boredom. *Philippine Computing Journal*, *6*(1), 43-66.

Hadjerrouit, S., & Bronner, A. (2014). An instrument for assessing the educational value of Aplusix (a+ x) for learning school algebra. In M. Searson & M. Ochoa (Eds.), *Society for*

Information Technology & Teacher Education International Conference (pp. 2241-2248). Chesapeake, VA: AACE.

Hadjerrouit, S., & Meier, A. (2010). an empirical evaluation of the software tool Aplusix for learning elementary algebra. In D. Gibson & B. Dodge (Eds.), *Society for Information Technology & Teacher Education International Conference* (pp. 3432-3439). Chesapeake, VA: AACE.

Nicaud, J. F. (1993). Building ITSs to be used: Lessons learned from the APLUSIX project. In P. Mendelsson, R. Lewis (Eds.), *Proceedings of Lessons from Learning* (pp. 181-198). Archamps, France: IFIP.

Nicaud, J. F., Bouhineau, D., Chaachoua, H., Huguet, T., & Bronner, A. (2003). A computer program for the learning of algebra: Description and first experiment. In *PEG 2003 Conference* (pp. 7-14). St. Petersburg, Russia: PEG.

Nicaud, J. F., Chaachoua, H., & Bittar, M. (2006). Automatic calculation of students' conceptions in elementary algebra from Aplusix log files. In M. Ikeda, K. Ashley, & T. Chan (Eds.), *Intelligent tutoring systems* (pp. 433-442). Berlin/Heidelberg: Springer.

Rodrigo, M. M. T., Anglo, E. A., Sugay, J. O., & Baker, R. (2008). Use of unsupervised clustering to characterize learner behaviors and affective states while using an intelligent tutoring system. In T. Chan et al., (Eds.), *International Conference on Computers in Education* (pp. 57-64). Taipei, Taiwan: ICCE.

Rodrigo, M. M. T., Baker, R. S., D'Mello, S., Gonzalez, M. C. T., Lagud, M. C., Lim, S. A., ... & Tep, S. (2008). Comparing learners' affect while using an intelligent tutoring system and a simulation problem solving game. In B. Woolf, E. Aimeur, R. Nkambou, & S. Lajoie (Eds.), *Intelligent Tutoring Systems* (pp. 40-49). Berlin/Heidelberg: Springer.

ASSISTments

ASSISTments is a domain-independent intelligent tutoring system (ITS), but the developers refer to it as a pseudo-tutor (a simplified ITS) since it doesn't have the full functionality and reasoning of a regular ITS (Tvarožek, Kravčík, & Bieliková, 2008). ASSISTments set out to create a platform that can easily be used for a wide range of content and experiments while allowing for enough flexibility that teachers accept it as their own (Heffernan & Heffernan, 2014). This allows the teachers to be in charge and has helped with its adoption. For this section, 22 articles were used from the years 2005-2015.

Domain Model

What content does this ITS cover? ASSISTments started off by providing tutoring assistance for mathematics (Razzaq et al., 2005). However, ASSISTments was designed to be domain-independent. This allowed others to use the ASSISTments system to create courses for physics, life science, earth science, English, and statistics (Gobert, Montaly, Toto, Sao Pedro, & Baker, 2010; Heffernan & Heffernan, 2014).

What level of understanding is the goal? As with other ITSs, ASSISTments strives for mastery learning and students gain mastery after solving three questions in a row correctly (Beck & Gong, 2013).

What is used to create the expert? Originally, the expert came from the Massachusetts Comprehensive Assessment System (MCAS) test items that were released to the public (Razzaq et al., 2005). Three math departmental heads were interviewed and asked questions like, "What kinds of questions would you ask the student?" and "What kind of hints would you give?" Based on their answers, the developers of Aplusix created the content for the test items. Later, teachers and the developers of ASSISTments annotated other examples to give hints and feedback and to

also identify the knowledge components associated with each problem (Feng & Heffernan, 2006). Once the knowledge components were identified, groups of knowledge components could be organized relationally and hierarchically. This creates what ASSISTments calls a transfer model (Feng & Heffernan, 2006). This transfer model acts as the ontology for the domain and does not require a production rule system like in other systems (Razzaq, Feng, et al., 2007). Yet, the transfer model still allows for useful student modeling techniques like knowledge tracing.

ASSISTments has a web-based item builder that both teachers and researchers can use to create items, hints, feedback, and identify knowledge components and map them to the transfer model (Junker, 2006). This process allows new courses to be created independent of the domain and for teachers to create transfer models that suit their preferences and classroom needs. Creating examples this way, which ASSISTments calls example tracing, reduces the time for content creation considerably (Mendicino, Razzaq, & Heffernan, 2009). Previous estimates placed the manhours required for one hour of instruction to be between 100 and 1000 hours. ASSISTments reduces that time to be between 10 and 30 man-hours for one hour of instruction.

Tutoring Model

Strategies. With tutoring being designed at an item level, global tutoring strategies seem to be inconsistent as they depend on the ideas and beliefs of the one creating each example. Further, each example is self-contained and independent of other problems. Other than a generic tutored problem-solving approach (Razzaq & Heffernan, 2009) to learning, no overall strategy was reported.

Tactics. The tutoring tactics of ASSISTments is discussed next, in particular: scaffolding, explaining, spaced practice, differential instruction, and hinting.

Scaffolding. Scaffolding for each problem only appears if the student answers the question incorrectly. After their first error, a student cannot try the problem further until they go through a sequence of scaffolding questions and answer one of them correctly (Razzaq et al., 2005). The scaffolding questions break down the problem into its constituent parts to get a clearer picture of what the student is struggling with (Koedinger, McLaughlin, & Heffernan, 2010).

Explain. On some occasions, messages are given to students that explain the problem rather than giving hints or other feedback (Razzaq, Feng, et al., 2007). This is used when it is assumed that the student already knows how to solve the problem.

Spaced Practice. Potentially unique to ASSISTments is the implementation of spaced practice (Heffernan & Heffernan, 2014). To promote learning and retention, once a skill is mastered, a student will be reassessed on a schedule of 7, 14, 28, and 60 days after initial mastery. If a student fails one of the reassessments, they will be given items to relearn the material and the spaced practice schedule will be reset.

Differential Instruction. Although the general progression of problems is linear (Mendicino et al., 2009), at times, the teacher can offer differential instruction (Heffernan & Heffernan, 2014). For example, if 67% of students get a problem set wrong, a different set of problems can be assigned to that group while the other 33% can receive a different set of problems.

Hinting. ASSISTments uses a hint sequence that follows this format: hint, hint, bottom-out hint (the answer; Razzaq et al., 2005). Students have the freedom to ask for hints at any time they are confused or are unable to answer (Koedinger et al., 2010).

Student Model

How is the student model created? There are no production rules in ASSISTments which means it can't solve its own problems nor follow student solution paths to provide feedback (i.e., model tracing). However, that is not needed considering ASSISTments is an example-tracing tutor and each example has all the information it needs contained within it whereas a model tracing tutor uses rules to dictate the domain.

As far as knowledge tracing is concerned, no dynamic version was used at first. Instead, a poor man's version was created. The student's percent correct across all previous problems combined with four Boolean values indicating the level of prior knowledge is compared to the average prior knowledge of all students in combination with the standard deviation of that average (Walanoski & Heffernan, 2006). This poor man's version can do as well as the MCAS to predict performance on future MCAS exams. Eventually, ASSISTments transitioned to a probabilistic model of inferring student knowledge (Pedro, Baker, Bowers, & Heffernan, 2013), but continued to claim that ASSISTments does not have a strong student model that does deep reasoning about the student or content (Heffernan & Heffernan, 2014).

What issues are there regarding the development of the student model? ASSISTments identified an issue with adaptive tutoring and mastery loops called wheel-spinning (Pedro et al., 2013). Wheel-spinning is defined as a student being stuck in a mastery loop – one where students spend too much time struggling to learn a topic without mastery and the system not moving on from the problem. To address this issue, ASSISTments locks out a student from solving a problem after 10 failed attempts on a skill. A student can return to the problem on a different day.

What data is collected? ASSISTments uses a logging unit that records student interaction with the system at a fine-grained level. ASSISTments also has other mechanisms that can abstract or coarsen the data (Junker, 2006).

Examples of fine-grained data collected include: total time spent in ASSISTments, time spent today in ASSISTments, number of problems seen, percent correct, timestamps, student ID, problem ID, student action type, student input and response, number of scaffolding questions, and number of hint requests (Feng & Heffernan, 2006); answer speed and accuracy (Junker, 2006); progress within a curriculum and its sections (Razzaq, Feng, et al., 2007); assistance needed (e.g., hints, scaffolding), number of attempts made, number of hints requested, response time, and number of opportunities to practice (Koedinger et al., 2010); proportion of correct actions, and number of first attempts on problems (a proxy for overall usage; Pedro et al., 2013).

These data can be used to help teachers with classroom interventions. The data are coarsened into a gradebook for each student and for the entire class thus providing valuable data. For example, student performance can be used to predict and report to teachers MCAS scores and performance levels (Feng & Heffernan, 2006). Data can also be used to alert the teacher to unusual behaviors like asking for more hints than other students, gaming, or student affective states (e.g., bored, engaged; Feng & Heffernan 2006; Pedro et al., 2013). The data are also used to create estimates of student knowledge (Pedro et al., 2013). This information helps teachers to know which items are difficult, which items students perform below the state average, and which items could use more focus in classroom instruction (Razzaq et al., 2005).

Interface

ASSISTments is a web-based tutor wherein students answer questions and receive scaffolding, hints, and feedback (Heffernan & Heffernan, 2014). ASSISTments comes with

several tools in the form of widgets (Razzaq, Feng, et al., 2007). Some low-level widgets include text labels, text fields, and images whereas some high-level widgets include multimedia, spell-checkers, and algebra parsing text fields.

ASSISTments provides resources for multiple stakeholders in the learning process. It tutors students, provides feedback to teachers to improve classroom decisions (Koedinger et al., 2010, and alerts parents by allowing teachers to share student data with them (Heffernan & Heffernan, 2014).

User experience studies of ASSISTments seem to focus on the teacher experience with the system. In 2005 (Razzaq et al., 2005), teachers thought highly of the system since it included MCAS items and scaffolded assistance. A few years later in 2007 (Razzaq, Feng, et al., 2007), the authors reported that some students found the forced scaffolding to be frustrating. However, not much else was reported in terms of user experience in the articles reviewed.

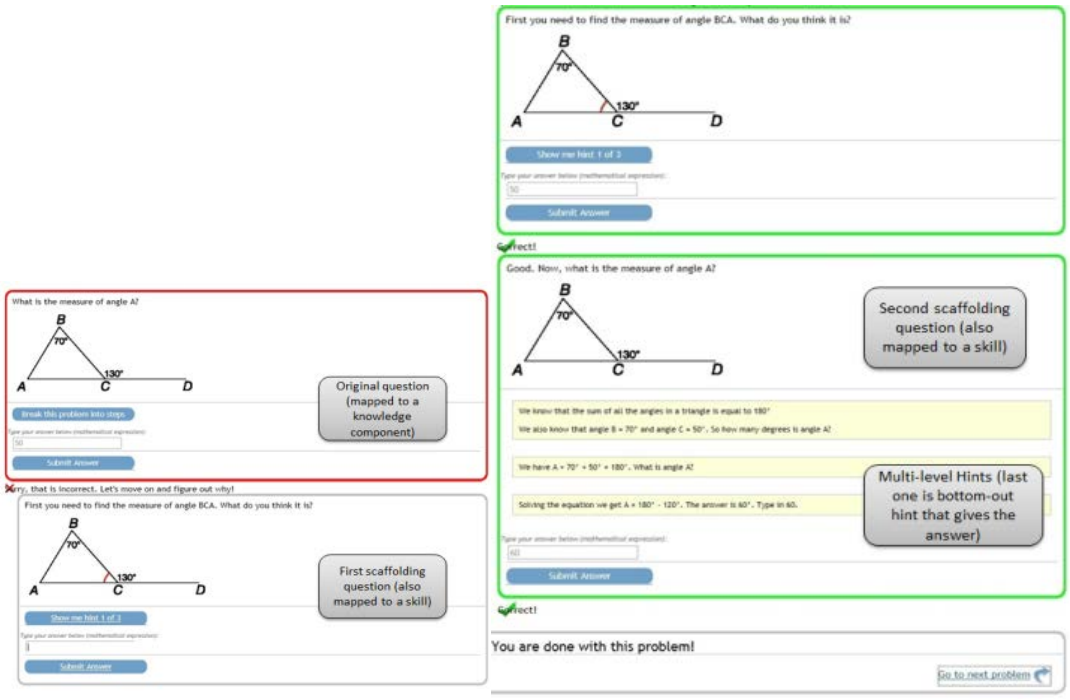

Figure 1. A problem in ASSISTments including scaffolding and hints (Pedra et al., 2013).

Learning Gains

ASSISTments was originally designed to predict MCAS scores and so learning gains were measured in its ability to predict these scores. At one point, student performance on ASSISTments had an r value of 0.75 in predicting MCAS scores two years later (Razzaq et al., 2005). Learning gains were later measured as effect sizes and studies reported effect sizes ranging from .23 to .61 (Heffernan & Heffernan, 2014; Koedinger et al., 2010; Mendicino et al., 2009; Singh et al., 2011;). Their research found that the scaffolding and tutoring was more beneficial for some groups over others (Koedinger et al., 2010; Razzaq & Heffernan, 2009; Razzaq, Heffernan, & Lindeman, 2007). For example, students with less knowledge benefited more from the scaffolding and hints than those students who had more knowledge and benefitted from a delayed feedback setting (Razzaq, Heffernan, et al., 2007). Focusing on the same categories of students, another study found that high performers do better learning from worked examples while low-performing students benefitted more from the tutored problem solving (Razzaq & Heffernan, 2009). Another study found that those with IEPs benefitted more from ASSISTments' tutoring than did their peers, resulting in an effect size of .50 (Koedinger et al., 2010).

References

Beck, J. E., & Gong, Y. (2013). Wheel-spinning: Students who fail to master a skill. In H. Lane, K. Yacef, J. Mostow, P. Pavlik (Eds.), *Artificial Intelligence in Education* (pp. 431-440). Berlin/Heidelberg: Springer.

Feng, M., & Heffernan, N. T. (2006). Informing teachers live about student learning: Reporting in the ASSISTment system. *Technology Instruction Cognition and Learning, 3*(1/2), 63-77.

Gobert, J. D., Montalvo, O., Toto, E., Sao Pedro, M., & Baker, R. S. J. D. (2010). The science ASSISTments project: Scaffolding scientific inquiry skills. In V. Aleven, J. Kay, & J. Mostow (Eds.), *Intelligent Tutoring Systems* (pp. 445-445). Berlin/Heidelberg: Springer.

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.

Junker, B. W. (2006). Using on-line tutoring records to predict end-of-year exam scores: experience with the ASSISTments project and MCAS 8th grade mathematics. In R. Lissitz (Ed.), *Assessing and Modeling Cognitive Development in School: Intellectual Growth and Standard Setting* (pp. 1-34). Maple Grove, MN: JAM Press.

Koedinger, K. R., McLaughlin, E. A., & Heffernan, N. T. (2010). A quasi-experimental evaluation of an on-line formative assessment and tutoring system. *Journal of Educational Computing Research, 43*(4), 489-510.

Mendicino, M., Razzaq, L., & Heffernan, N. T. (2009). A comparison of traditional homework to computer-supported homework. *Journal of Research on Technology in Education, 41*(3), 331-359.

Pedro, M. O., Baker, R., Bowers, A., & Heffernan, N. (2013). Predicting college enrollment from student interaction with an intelligent tutoring system in middle school. In *Educational Data Mining 2013* (pp. 177-184). Memphis, TN: IEDMS.

Razzaq, L. M., & Heffernan, N. T. (2009). To tutor or not to tutor: That is the question. In V. Dimitrova, R. Mizoguchi, B. du Boulay, & A. Graesser (Eds.), *Proceedings of the Conference on Artificial Intelligence in Education* (pp. 457-464). Amsterdam: IOS Press.

Razzaq, L. M., Feng, M., Nuzzo-Jones, G., Heffernan, N. T., Koedinger, K. R., Junker, B., … & Rasmussen, K. (2005). Blending assessment and instructional assisting. In C. Looi, G. McCalla, B. Bredeweg, & J. Breuker (Eds.), *AIED* (pp. 555-562). The Netherlands: IOS Press Amsterdam.

Razzaq, L., Feng, M., Heffernan, N., Koedinger, K., Junker, B., Nuzzo-Jones, G., ... & Walonoski, J. (2007). A web-based authoring tool for intelligent tutors: Blending assessment and instructional assistance. In N. Nedjah, L. Mourelle, M. Borges, & N. Almeida (Eds.), *Intelligent Educational Machines*, *44,* 23-49. Berlin/Heidelberg: Springer.

Razzaq, L., Heffernan, N. T., & Lindeman, R. W. (2007). What level of tutor interaction is best? *Frontiers in Artificial Intelligence and Applications*, *158*, 222-229.

Singh, R., Saleem, M., Pradhan, P., Heffernan, C., Heffernan, N., Razzaq, L., & Dailey, M. (2011). Improving K-12 homework with computers. In *Proceedings of the Artificial Intelligence in Education Conference* (pp. 328-336). New York, NY: Springer.

Tvarožek, J., Kravčík, M., & Bieliková, M. (2008). Towards computerized adaptive assessment based on structured tasks. In W. Nejdl, J. Kay, P. Pu, & E. Herder (Eds.), *International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems* (pp. 224-234). Berlin/Heidelberg: Springer.

Walonoski, J., & Heffernan, N. (2006). Detection and analysis of off-task gaming behavior in intelligent tutoring systems. In M. Ikeda, K. Ashley, & T. Chan (Eds.), *Intelligent Tutoring Systems* (pp. 382-391). Berlin/Heidelberg: Springer.

AutoTutor

AutoTutor is an ITS that covers the subjects of computer literacy, conceptual Newtonian physics, and critical thinking by using mixed-initiative dialogue. This means that AutoTutor will provide a question or prompt that the student then answers or asks follow-up questions. If the answer given is too short, incomplete, or wrong, AutoTutor helps the student construct the right answer by asking additional questions to draw out information or will correct the student (Graesser, Chipman, Haynes, & Olney, 2005). This review was conducted using 20 articles from the years 1999-2014.

Domain Model

What content does this ITS cover? AutoTutor was originally designed to cover topics relating to computer literacy (i.e., fundamentals of hardware, operating systems, and the internet; Graesser, Wiemer-Hastings, Wiemer-Hastings, Kreuz, & Tutoring Research Group, 1999). The system was later expanded to cover Newtonian physics and critical thinking skills (D'Mello et al., 2008; Graesser, Hu, et al., 2001).

What level of understanding is the goal? Shifting from the traditional goal of mastery learning, AutoTutor instead focuses on deep reasoning through its dialogue-based tutoring (Graesser, Hu, et al., 2001; D'Mello & Graesser, 2012). This puts more focus on higher levels of learning rather than focusing on the basics and skill-and-drill approaches.

What is used to create the expert? The expert is organized into a curriculum script which is a loosely ordered but well-defined set of skills, concepts, example problems, and question-answer units (Graesser et al., 1999). This script organizes the topics and content of tutorial dialogue by including things like example problems, figures and diagrams, questions for the tutor to ask, hints, libraries for good and bad responses, buggy knowledge, misconceptions,

and errors. While it is not possible to anticipate all possible student answers as a dialogue-based system, AutoTutor can recognize good and bad student responses, which it can then add to the curriculum script, thus growing its knowledge base. These curriculum scripts are what allow AutoTutor to expand its content offerings as the questions, summary, tutor prompts and answers, etc., can be written independently of the system.

Tutoring Model

Strategies. One of the defining features of AutoTutor is its ability to engage in natural dialogue with a student with the aim of getting a student to talk and explore what they know (Graesser et al., 1999)—thus more fully replicating a one-on-one tutoring session. Based on about 100 hours of naturalistic tutoring sessions (Graesser et al., 1999), AutoTutor's design is based on explanation-based constructivist theories and collaborative constructivist activities that occur in human tutoring (Graesser et al., 2003) as well as the ability of a human tutor to adapt to a student's knowledge (Graesser et al., 2005).

AutoTutor demonstrates its explanation-based constructivist theory in the way it models student knowledge and communicates with the student. That is, AutoTutor only believes a student knows something after the student has stated it (Jackson, Mathews, Link, Olney, & Graesser 2003). In this sense, a student must do a sufficient job in convincing the tutor (AutoTutor) that they know enough about a subject. Only after such an explanation will AutoTutor give the student credit for that knowledge.

The collaboration component is very evident in most tutor-student interactions. AutoTutor will present a question or problem to the student and asks them to state the answer. Generally, students only provide parts of the ideal solution and several dialogue turns are required for the student's answer to be fully developed. For challenging questions, it takes about

100 dialogue turns for the student to supply a sufficient answer (D'Mello & Graesser, 2012). In this process, called expectations and misconceptions dialogue (EMT dialogue), AutoTutor compares the student's response to an ideal answer. If a student's response only satisfies certain portions of the ideal answer, then AutoTutor will ask the student for more information or provide it if needed. Thus, the two work together to help the student's knowledge become more like the expert's. Throughout this, AutoTutor is capable of dynamically applying and testing different pedagogical principles in choosing which expectation it coaches a student through (Graesser et al., 2005). In some cases, AutoTutor may opt for an expectation that lies in a student's zone of proximal development (estimated by the system), while in other cases, it chooses to maximize the coherence of the conversation by choosing an expectation that is most similar to what was recently discussed.

Other tutoring systems that have implemented a dialogue approach restrict student input to some degree (Freedman, 1997), usually by allowing only answers from the student and not questions. AutoTutor goes beyond those restrictions and encourages students to ask questions of the system (Rajan et al., 2001). This mixed-initiative dialogue is part of AutoTutor's adaptation to what a student knows. So, if a student requests more information about a topic or seems confused, AutoTutor can respond to what the student knows and change what is being presented to meet the needs of the student (Graesser et al., 2005).

With mixed-initiative dialogue being part of the AutoTutor system, this introduces the need for tutoring strategies for the dialogue (e.g., Socratic dialogue) that expert tutors implement, not just for how the system operates. Surprisingly, AutoTutor does not implement sophisticated tutoring strategies, but instead uses the tutoring strategies of unskilled tutors (Graesser, Hu, et al., 2001). The reasoning behind this is based on a previous meta-analysis in which tutor experience

did not significantly predict learning gains (Cohen, Kulik, & Kulik, 1982; Graesser et al., 1999). This finding led the authors of AutoTutor to hypothesize that, perhaps, it is not the sophisticated tutoring techniques or expertise of the tutor that matters, but rather the conversational properties of tutorial dialogue that are responsible for learning (Graesser et al., 2003).

Going beyond cognitive concerns, AutoTutor's strategies also include the affective and metacognitive needs of a student (D'Mello et al., 2008). For example, consider what happens when a student is frustrated. AutoTutor relies on attribution theory to address the student's frustration in an empathic and empowering way by helping the student maintain positive self-efficacy and blaming the system. In other instances, AutoTutor may push the student by offering encouragement when they are confused, giving the student affirmations that they are capable of solving the problem.

Tactics. The tutoring tactics of AutoTutor are described next, more specifically its dialogue framework, dialogue moves, conversational behaviors, feedback, and use of simulation.

Dialogue Framework. Following the basic structure human tutors use in their interactions, AutoTutor follows a five-step framework that structures the basics of dialogue with a student (Graesser et al., 1999). First, AutoTutor will ask a question. Second, the student answers the question. Third, the tutor gives short, immediate feedback on the quality of the student's response. Fourth, the tutor and student collaboratively work together to improve the quality of the student's answer. Fifth, the tutor assesses the student's understanding with follow-up questions. Eventually, the fifth step was removed from AutoTutor's framework since it produced unreliable student responses (Person, Graesser, Kreuz, & Pomeroy, 2003).

Dialogue Moves. AutoTutor began with 12 dialogue moves that it would employ in its conversations with students (Rajan et al., 2001). Throughout development, AutoTutor refined its

dialogue moves and more recently described nine (D'Mello & Graesser, 2012). Those nine are discussed here. The first move is called Main Question wherein AutoTutor initiates the dialogue with a question. After this, the moves come in no particular order. Second, is a Pump where AutoTutor is asked for more information regarding the topic. Hint is third and is a leading question or statement that directs the student's thinking. Fourth, Prompt, leads a student to supply a word that is missing from an important idea. Fifth, Short Feedback, signals the quality of the student response in terms that are positive, neutral, or negative. Sixth, Correction, corrects a statement made by the learner. Seventh, Assertion, states a main idea within the problem or the answer to the problem. Eighth, Answer, is a response to a learner's question regarding a definition of a concept. Ninth, Summary, summarizes the full answer to the student at the end of the dialogue.

Conversational Behaviors. In addition to the aforementioned verbal behaviors, AutoTutor expresses other conversational cues via a talking head that acts as the system's pedagogical agent (Rajan et al., 2001). Behaviors for this agent include head movements, facial expressions, intonation variation, gaze behaviors, and blinking patterns. These behaviors are used in conjunction with feedback so that a human-like, empathic response is also given. So, if a student receives positive feedback, the agent will smile and say the praise with enthusiasm.

Feedback. There are three levels of feedback given in AutoTutor (Graesser, VanLehn, Rose, Jordan, & Harter, 2001). The first level of feedback is called backchannel feedback. This feedback can be verbal or nonverbal and is meant to facilitate the conversation but not add any content (e.g., head nods, saying "Uh-huh"; Rajan et al., 2001). The next level of feedback is evaluative pedagogical feedback. Here, responses are given along a spectrum of negative to

neutral to positive feedback letting the student know of the quality of their work. Lastly, there is corrective feedback that repairs buggy knowledge and misconceptions.

Simulation. In one study, a simulation environment was designed that presented physics problems to students (Graesser, Jackson, Kim, & Olney, 2006). Within these 3D micro-worlds, students could see the question play out. Then, to help them answer the tutor's questions, they could manipulate the variables of the micro-world to see how changing one thing would affect the other parameters.

Student Model

How is the student model created? AutoTutor's developers make the argument that real tutors have a hard time gauging student understanding and that there is a lot of misunderstanding that happens. As such, AutoTutor can manage well enough with a shallow understanding of the student's knowledge as long as the selection of dialogue moves is strategic (Graesser et al., 1999).

That being said, AutoTutor performs knowledge tracing by using the student's responses compared to an ideal answer (Rajan et al., 2001). The degree to which a student knows something is reflected in the quality of the answer, which is measured by using a technique called latent semantic analysis (LSA). AutoTutor can compare a student's response to an ideal answer and the degree to which they match shows the quality.

For example, let's suppose a problem has seven expectations that need to be met in the ideal answer. A student can only supply four of them before the tutor supplies the remaining expectations. However, the student encountered a similar problem before and could only provide two of the expectations the first time. AutoTutor is able to create a local and global model of the student's knowledge with this information (D'Mello & Graesser, 2012). The local

model deals with the current student response and gives the tutor information in terms of how the student is currently doing. This information will affect the dialogue moves and feedback it provides. The global model uses the current response and all previous student responses for the current problem, which provides a model of the student knowledge for the current problem.

What data is collected? The student's responses are the main source of data for AutoTutor. As was mentioned, these responses are compared to ideal and bad answers to determine the level of semantic similarity. In this way, AutoTutor can perform knowledge tracing. AutoTutor tracks the following features of student responses: the quality of a student's current response, how much of the current topic is covered so far (in terms of expectations met), and the student's overall ability level for the topic (global competence; Person et al., 2003). Other features of student response that are recorded include the verbosity of the student, reaction and response times, the length of an answer, and other parameters about the conceptual quality (Sidney et al., 2005).

After identifying six academic emotions (frustration, boredom, flow, confusion, eureka, and neutral), AutoTutor began to collect information to model students' affect states (Craig, Graesser, Sullins, & Gholson, 2004). AutoTutor uses non-intrusive sensing devices in the form of a facial camera that runs facial expression recognition and tracks the pupils, a pressure-sensitive chair for posture readings, and log files for conversation clues (D'Mello et al., 2008; Sidney et al., 2005). The facial camera was able to achieve an accuracy of 68% when classifying facial action units to determine student affect. The pressure-sensitive chair was able to classify interest by analyzing posture sequences with an accuracy of 82.3% for known subjects and 76.5% on new subjects. The log files are used to classify responses into five categories: meta-

communicative, metacognitive, shallow comprehension, assertions reflecting deep comprehension, and an "other" category (Sidney et al., 2005).

Interface

AutoTutor's interface varies a bit for each version, but most of them have the following five windows: a talking head that serves as a pedagogical agent, a textbox for typed student input, a textbox displaying the current question, and a graphics box that displays pictures or animations relevant to the current question, and a dialogue history (D'Mello & Graesser, 2012). Once AutoTutor presents the question, the mixed-initiative dialogue begins. While a student's input is usually typed, AutoTutor responds with facial expressions, gestures, and a synthesized voice which has appropriate intonation (Person et al., 2003). Not all student input is typed, though, as a version of AutoTutor has been developed which can handle some spoken input from the student (D'Mello & Graesser, 2012).

While nothing was found describing how students felt when using the system, one study did attest to the naturalness of AutoTutor's generated dialogue moves (Graesser et al., 2003). A bystander Turing test was conducted where a third party was asked to judge whether a transcript of a tutor-student interaction was conducted by a computer or human tutor. Bystanders were unable to discriminate whether a move was generated by a computer or a tutor.

AutoTutor has also been used to study the effects that multimedia has on learning (Graesser et al., 2003) answering the question, "To what extent does having a talking head or using synthesized speech over text affect learning?" They found that by removing the agent or presenting text only (no voice or pedagogical agent), there was a non-significant decrease of 0.13 sigma in learning gains. What really mattered was being able to present the right content. These results along with another study dispute the redundancy effect for multimedia learning which

says that duplicating modes of communication (e.g., having speech and text) hurts learning. The other study (Graesser, Jeon, & Dufty, 2008) found a non-significant, positive impact on learning with a sigma of 0.34.

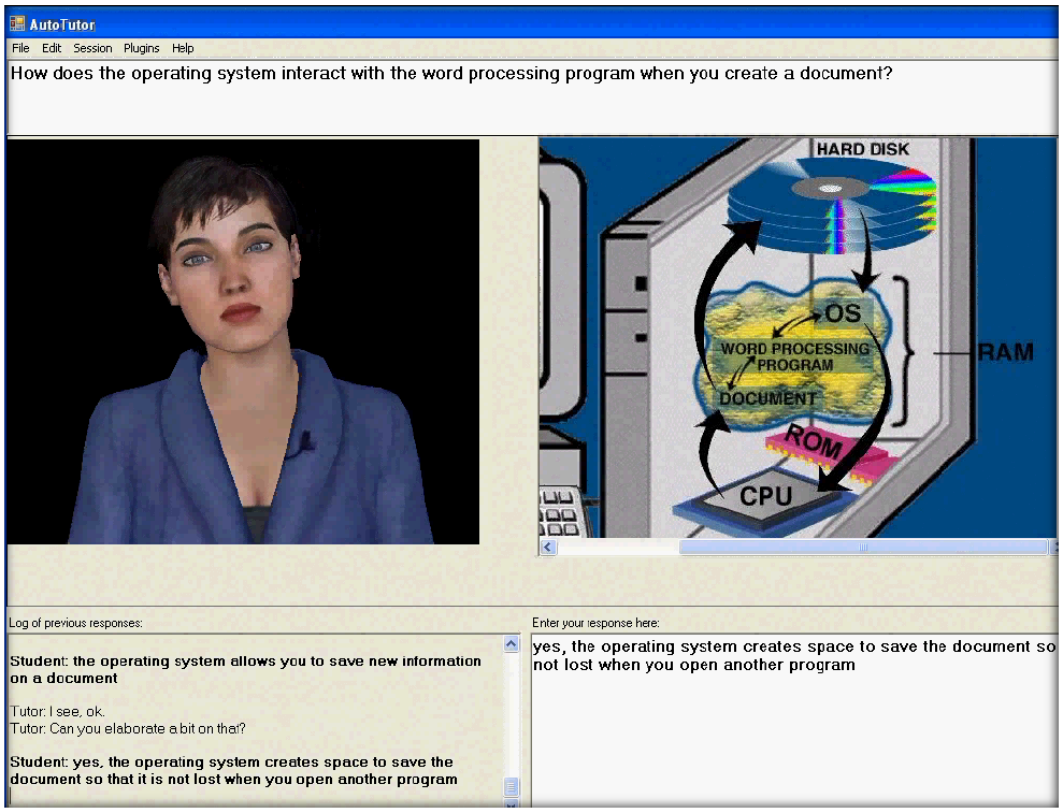

Figure 1. The interface of AutoTutor (D'Mello & Graesser, 2012).

Learning Gains

Several studies have shown significant learning gains for students using AutoTutor. Over 1000 students have taken part in these studies and there have almost always been significant effect sizes ranging from 0.2 to 1.5 sigma with an average of 0.8 for computer literacy and physics (D'Mello & Graesser, 2012). These results were compared to a do-nothing control or reading from a textbook on the same topics for the same amount of time. The developers of AutoTutor concluded that when there is an intermediate gap of knowledge between the student

and AutoTutor, then the system is most effective. If the student does not know enough or knows too much, AutoTutor's effectiveness is diminished (D'Mello & Graesser, 2012).

References

Bloom, B. S. (1984). The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring. *Educational Researcher*, *13*(6), 4-16.

Cohen, P. A., Kulik, J. A., & Kulik, C. C. (1982). Educational outcomes of tutoring: A meta-analysis of findings. *American Educational Research Journal,19*(2)*,* 237-248

Craig, S., Graesser, A., Sullins, J., & Gholson, B. (2004). Affect and learning: An exploratory look into the role of affect in learning with AutoTutor. *Journal of Educational Media, 29*(3), 241-250. doi:10.1080/1358165042000283101

D'Mello, S., Jackson, T., Craig, S., Morgan, B., Chipman, P., White, H., ... & Graesser, A. (2008). AutoTutor detects and responds to learners affective and cognitive states. In *Workshop on Emotional and Cognitive Issues at the International Conference on Intelligent Tutoring Systems* (pp. 306-308). Montreal, Canada.

D'Mello, S., & Graesser, A. (2012). AutoTutor and affective AutoTutor: Learning by talking with cognitively and emotionally intelligent computers that talk back. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, *2*(4), 23-63.

Graesser, A. C., Chipman, P., Haynes, B. C., & Olney, A. (2005). AutoTutor: An intelligent tutoring system with mixed-initiative dialogue. *IEEE Transactions on Education*, *48*(4), 612-618.

Graesser, A. C., Hu, X., Susarla, S., Harter, D., Person, N. K., Louwerse, M., & Olde, B. (2001). AutoTutor: An intelligent tutor and conversational tutoring scaffold. In *10th ICAI in Education* (pp. 47-49). San Antonio, TX: AIED.

Graesser, A. C., Jackson, G. T., Kim, H. J. J., & Olney, A. (2006). AutoTutor 3-D simulations: Analyzing users' actions and learning trends. In *Proceedings of the Cognitive Science Society* (pp. 1557-1562). Mahwah, NJ: Erlbaum.

Graesser, A. C., Jeon, M., & Dufty, D. (2008). Agent technologies designed to facilitate interactive knowledge construction. *Discourse Processes, 45*(4–5), 298–322.

Graesser, A. C., Moreno, K., Marineau, J., Adcock, A., Olney, A., Person, N. (2003). AutoTutor improves deep learning of computer literacy: Is it the dialog or the talking head? *Artificial Intelligence in Education: Shaping the Future of Learning through Intelligent Technologies*, *97*, 47-55.

Graesser, A. C., VanLehn, K., Rosé, C. P., Jordan, P. W., & Harter, D. (2001). Intelligent tutoring systems with conversational dialogue. *AI Magazine*, *22*(4), 39-53.

Graesser, A. C., Wiemer-Hastings, K., Wiemer-Hastings, P., Kreuz, R., & Tutoring Research Group. (1999). AutoTutor: A simulation of a human tutor. *Cognitive Systems Research*, *1*(1), 35-51.

Jackson, T., Mathews, E., Lin, K. I., Olney, A., & Graesser, A. (2003). Modeling student performance to enhance the pedagogy of AutoTutor. In P. Brusilovsky, A. Corbett, & F. de Rosis (Eds.), *User Modeling 2003* (pp. 368-372). Berlin/Heidelberg: Springer.

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.

Person, N. K., Craig, C., Price, P., Hu, X., Gholson, B., & Graesser, A. C. (2000). Incorporating human-like conversational behaviors into AutoTutor. In *Proceedings of the Workshop on*

Achieving Human-like Behavior in the Interactive Animated Agents at the Agents 2000 Conference (pp. 85-92). Barcelona, Spain: The ACM Press.

Person, N. K., Graesser, A. C., Kreuz, R. J., & Pomeroy, V. (2003). Simulating human tutor dialog moves in AutoTutor. *International Journal of Artificial Intelligence in Education (IJAIED)*, *12*, 23-39.

Rajan, S., Craig, S. D., Gholson, B., Person, N. K., Graesser, A. C., & Tutoring Research Group. (2001). AutoTutor: Incorporating back-channel feedback and other human-like conversational behaviors into an intelligent tutoring system. *International Journal of Speech Technology*, *4*(2), 117-126.

Sidney, K. D., Craig, S. D., Gholson, B., Franklin, S., Picard, R., & Graesser, A. C. (2005). Integrating affect sensors in an intelligent tutoring system. In *Affective Interactions: The Computer in the Affective Loop Workshop* (pp. 7-13). New York, NY: ACM Press.

CIRCSIM-Tutor

CIRCSIM-Tutor is a natural-language dialogue tutor designed for medical students. It focuses on teaching causal relationships as students fill out a prediction table to determine the effects of changes in the cardiovascular system. Once a student has made their predictions, the tutor engages in dialogue with the student to coach them through their difficulties. Sixteen articles from the years 1991-2010 were used as the basis for this review.

Domain Model

What content does this ITS cover? CIRCSIM-Tutor covers the baroreceptor reflex of the cardiovascular system for first year medical students (Evens et al., 1997; Shim, Evens, Michael, & Rovick, 1991). In doing so, it teaches qualitative reasoning in the form of causal relationships between variables that affect the baroreceptor reflex.

What level of understanding is the goal? CIRCSIM-Tutor distinguishes between two types of knowledge: declarative and procedural (Hume, 1992). As students are tutored, CIRCSIM-Tutor teaches declarative knowledge of the cardiovascular system and procedural knowledge involving the rules and heuristics that need to be applied in problem-solving situations (Woo et al., 2006).

What is used to create the expert? Two experts were used to create a knowledge base that is organized around variables and causal relations (Evens et al., 1997; Shim et al., 1991). Production rules are written that allow the tutor to have a problem solver that can trace solution paths (model tracing) and solve the problems it presents to students (Woo et al., 2006).

Tutoring Model

Strategies. CIRCSIM-Tutor uses natural language dialogue to tutor students (Zhou & Evens, 1999). The strategies used for CIRCSIM-Tutor are based on over 75 hours of human-to-

human tutoring transcripts (Freedman & Evens, 1996). CIRCSIM-Tutor begins with a coach-like approach when students initially fill out a prediction table (predicting the cause and effect of variables) that is used to determine student errors and areas for tutoring. Once it moves on from the prediction table entry and addresses student errors, CIRCSIM-Tutor uses Socratic teaching for interactive tutoring (Woo et al., 2006). CIRCSIM-Tutor can have more of an interactive approach since it does not teach new material, but rather tutors material the students should have already studied (Freedman, 1997). However, CIRCSIM-Tutor does not handle mixed-initiative dialogue very well and actively constrains student responses by doing things like asking short-answer questions or ending every dialogue move with a request of some sort (Freedman, 1997; Zhou & Evens, 1999).

Tactics. The following section will describe the tutoring tactics of CIRCSIM-Tutor. Specifically, CIRCSIM-Tutor's hints, dialogue moves, and responses to student initiative will be addressed.

Hints. CIRCSIM-Tutor first provides hints when a student shows signs of difficulty such as making an error, asking a question, or being confused (Hume, Michael, Rovick, & Evens, 1996). The tutor will try as many as two hints before providing an explanation and moving on from the topic. The system is sensitive enough to switch hinting tactics when the student model determines that the student does not have sufficient background knowledge to respond to the hints or if the student rarely responds well to hints (Hume et al., 1996). CIRCSIM-Tutor keeps a tutor history to avoid giving the same hints or returning to causal relationships that were already tutored (Zhou et al., 1999).

The hints that are given were originally very general and did not consider what the student had said previously (Zhou et al., 1999). Hints started to become tailored to student

answers by tailoring them when students give partial hints (Zhou et al., 1999). CIRCSIM-Tutor uses the following hints in ranked order: (a) hints specifically related to the student's answers, (b) hints involving equations, (c) hints involving evocative synonyms, (d) hints involving a related anatomical object, (e) hints giving intermediate logical step, and a sixth category that covers everything else.

Dialogue Moves. After student knowledge is tested at first with a prediction table, CIRCSIM-Tutor identifies misconceptions and then engages in a natural language tutoring dialogue (Woo et al., 1991). Examples of the types of dialogue moves CIRCSIM-Tutor can make include direct questions, hints, explanations, and reminders (Woo et al., 1991). Every dialogue move has the same basic structure: an optional response to the student's previous statement, new material, and then a question or imperative for the student (Freedman & Evens, 1996). As was stated earlier, CIRCSIM-Tutor ends each dialogue move with a question or imperative to limit the size and complexity of expected student responses which reduces the chances of misunderstanding and student frustration (Freedman, 1997). When CIRCSIM-Tutor does receive student input that it does not recognize, it describes the kind of input that is expected so that the student can respond appropriately (Evens et al., 1997).

Responses to Student Initiative. Although student input is constrained, CIRCSIM-Tutor does have ways of dealing with unexpected input like asking a question (Evens et al., 1997). For example, if a student requests more information, CIRCSIM-Tutor can respond to the statement and then return to the lesson plan, it can replace the current plan with the student's request, it can delay answering the request by putting it elsewhere in the agenda, it can acknowledge the student input without responding, or it can ignore what the student said.

Student Model

How is the student model created? Since CIRCSIM-Tutor addresses both declarative and procedural knowledge, it needed a way to model both forms. The overlay model and bug library model were popular at the time to deal with declarative and procedural knowledge modeling respectively. CIRCSIM-Tutor decided to combine both for their student model (Shim et al., 1991; Woo et al., 2006). The overlay model assumes that student knowledge is a subset of expert knowledge and any deviance from the model showed discrepancies in a student's declarative knowledge (Hume, 1992). A bug library assumes that student errors reflects misconceptions about how to solve a problem and thus is informative in identifying gaps in procedural knowledge (Hume, 1992). By integrating the two, CIRCSIM-Tutor could now keep track of students' ability to make predictions and their performance on questions as well as correcting any misconceptions students have (Woo et al., 2006).

What data is collected? Initial student data is collected through a prediction table in which the student predicts the causal relationships between various parameters involving the baroreceptor reflex (Hume, 1992). This allows the tutoring dialogue to proceed with some information about what a student knows so that it can make a tutoring plan to address student deficiencies. After that, the basic piece of data collected from each student is their written responses in the dialogue.

CIRCSIM-Tutor has an input understander which extracts what it needs from student responses so that it can understand what the student is trying to say while being as permissive as possible in what a student can input (Glass, 2001). Student answers are then classified into eight distinct categories (Zhou et al., 1999): correct, partially correct answer, near miss answer, "I

don't know" answer, "grain of truth" answer, misconception, other incorrect answers, and mixed answers (having a combination of other classifications).

CIRCSIM-Tutor maintains a performance model that estimates student competence (Zhou & Evens, 1999). These estimates happen on four levels: globally (an overall measure of student performance), procedure-level (a measure for each problem), stage assessment (a measurement for each physiological stage in a problem), and local (a measurement for each variable). The information provided by the performance model coupled with what a student knows allows CIRCSIM-Tutor to adapt the difficulty level.

For each concept within CIRCSIM-Tutor, a student reply history is kept (Zhou & Evens, 1999). This history stores information on the part of the causal chain that was covered along with the student's answers in the dialogue. Each response is also put into one of the categories mentioned above.

A student solution record is kept which records the number of errors a student made while solving a problem (Zhou & Evens, 1999). A detailed description of the error is kept. This record is useful in being a rough estimate of the student's ability at qualitative reasoning since the solution path cannot be inferred from a student's predictions.

A final component of the student model is the tutoring history model wherein the plan history and discourse history are kept (Zhou & Evens, 1999). In other words, this is a record of what the system has done without any direct information on the student. This information informs CIRCSIM-Tutor on what it needs to do next.

Interface

CIRCSIM-Tutor has four windows in its application (Michael, Rovick, Glass, Zhou, & Evens, 2003). The first window is the tutoring window which maintains the dialogue history and

allows the student to type their response. A procedure description window explains the current scenario. The prediction table window is where a student inputs their predictions for the effect of the scenario on different parameters in the causal relationship map. The final window is a notes window where students can leave notes for themselves.

A survey was conducted to see what students thought of the system (Woo et al., 2006). The survey was a 5-point Likert scale with 1 being "Definitely YES" and 5 being "Definitely NO". Based on the responses, students said the system was easy to use (mean – 1.7), that they learned a lot (mean – 1.9), and that they would recommend it to others (mean – 1.9). Mean responses to other questions in the survey were all positive and indicated a pleasant experience with the system.

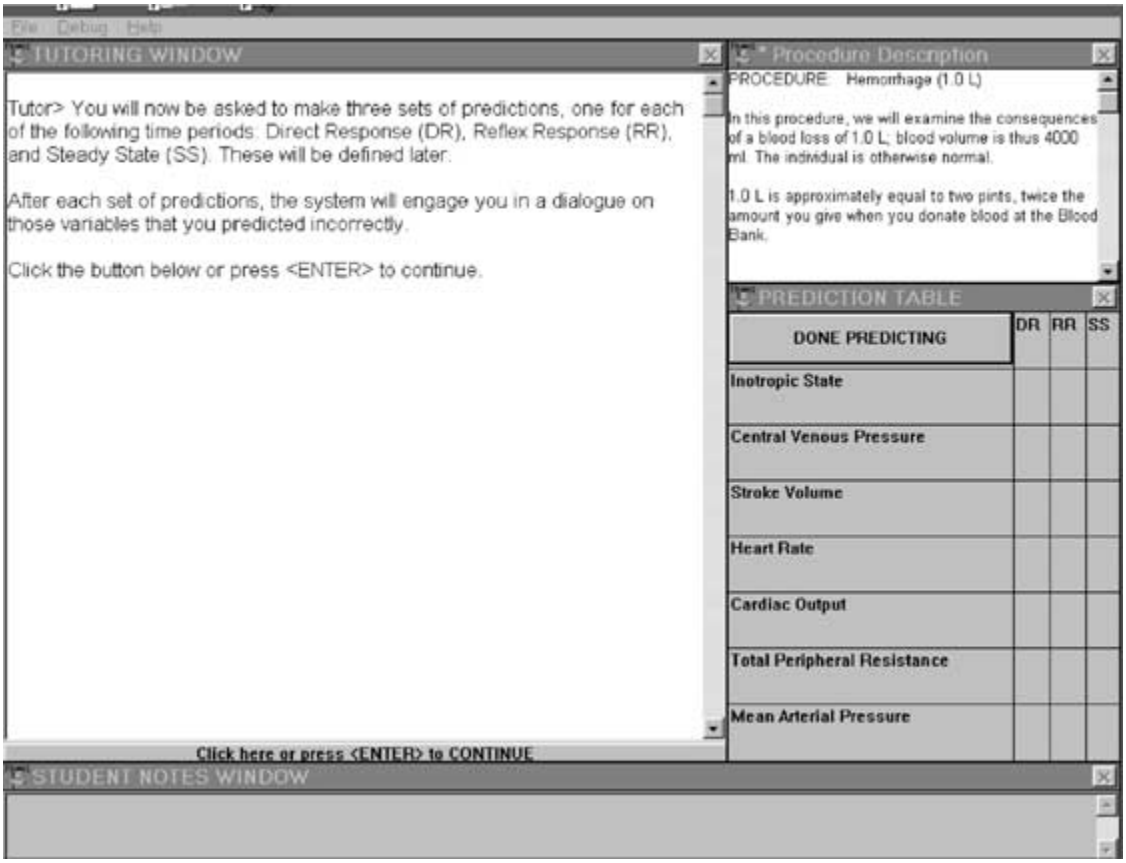

Figure 1. The user interface of CIRCSIM-Tutor (Michael et al., 2003).

Learning Gains

One experiment was reported that measured the effect size of working with CIRCSIM-Tutor (Woo et al., 2006). A control group where students read from a textbook was compared to students who used CIRCSIM-Tutor. The control group outperformed CIRCSIM-Tutor in memorizing relationship information with a significant effect size of 1.27 compared to CIRCSIM-Tutor's significant 0.65 effect size for the same portion of the test. However, CIRCSIM-Tutor outperformed the control group in terms of problem-solving and making correct predictions with a significant effect size of 1.24 for CIRCSIM-Tutor and 0.48 for the control. This was the only study found that reported learning gains.

References

Evens, M. W., Chang, R. C., Lee, Y. H., Shim, L. S., Woo, C. W., Zhang, Y., ... & Rovick, A. A. (1997). CIRCSIM-Tutor: An intelligent tutoring system using natural language dialogue. In *Proceedings of the fifth conference on Applied natural language processing: Descriptions of system demonstrations and videos* (pp. 13-14). Oxford, OH: Association for Computational Linguistics.

Freedman, R. (1997). Degrees of mixed-initiative interaction in an intelligent tutoring system. In *Working Notes of AAAI97 Spring Symposium on Mixed-Initiative Interaction* (pp. 44-49). Stanford, CA: AAAI.

Freedman, R., & Evens, M. W. (1996). Generating and revising hierarchical multi-turn text plans in an ITS. In C. Frasson, G. Gauthier, A. Lesgold (Eds.), *International Conference on Intelligent Tutoring Systems* (pp. 632-640). Berlin/Heidelberg: Springer.

Glass, M. (2001). Processing language input in the CIRCSIM-Tutor intelligent tutoring system. In J. Moore et al., (Eds.), *Artificial Intelligence in Education* (pp. 210-221). San Antonio, TX: IOS Press.

Hume, G. D. (1992). A dynamic student model in a cardiovascular intelligent tutoring system. In *Computer-Based Medical Systems, 1992. Proceedings, Fifth Annual IEEE Symposium* (pp. 370-377). Durham, NC: IEEE.

Hume, G., Michael, J., Rovick, A., & Evens, M. (1996). Hinting as a tactic in one-on-one tutoring. *The Journal of the Learning Sciences*, *5*(1), 23-47.

Michael, J., Rovick, A., Glass, M., Zhou, Y., & Evens, M. (2003). Learning from a computer tutor with natural language capabilities. *Interactive Learning Environments*, *11*(3), 233-262.

Shim, L., Evens, M. W., Michael, J. A., & Rovick, A. A. (1991). Effective cognitive modeling in an intelligent tutoring system for cardiovascular physiology. In *Proceedings of the Fourth Annual IEEE Symposium* (pp. 338-345). Baltimore, MD: IEEE.

Woo, C. W., Evens, M. W., Freedman, R., Glass, M., Shim, L. S., Zhang, Y., ... & Michael, J. (2006). An intelligent tutoring system that generates a natural language dialogue using dynamic multi-level planning. *Artificial Intelligence in Medicine*, *38*(1), 25-46.

Woo, C. W., Evens, M., Michael, J., & Rovick, A. (1991). Dynamic instructional planning for an intelligent physiology tutoring system. In *Computer-Based Medical Systems, 1991. Proceedings of the Fourth Annual IEEE Symposium* (pp. 226-233). IEEE.

Zhou, Y., & Evens, M. W. (1999). A practical student model in an intelligent tutoring system. In. *Proceeding of the 11th IEEE International Conference* (pp. 13-18). Chicago, IL: IEEE.

Zhou, Y., Freedman, R., Glass, M., Michael, J. A., Rovick, A. A., & Evens, M. W. (1999). Delivering hints in a dialogue-based intelligent tutoring system. In *AAAI/IAAI 1999* (pp. 128-134). Orlando, FL: AAAI.

Cognitive Tutor

Cognitive Tutor, developed at Carnegie Mellon University, has arguably the most success at bridging the gap between research and application being used by more than 500,000 students every year (Koedinger et al., 2010). Cognitive Tutor was created to provide a full course experience for teacher and student alike as it tutors students in subjects like algebra, geometry, and genetics. This review was conducted using 19 articles from the years 1998-2014.

Domain Model

What content does this ITS cover? Cognitive Tutor is one of the more prolific ITSs in that it has been used in several domains and is used by hundreds of thousands of students (Koedinger et al., 2010). Cognitive Tutor originally was designed to teach first-year, high school algebra (Koedinger & Anderson, 1998), but has been expanded to include algebra II (Corbett, McLaughlin, Scarpinatto, & Hadley, 2000), programming, geometry (Corbett, McLaughlin, & Scarpinatto, 2000), scatterplots (Baker, Corbett, Koedinger, & Wagner, 2004), metacognitive help-seeking behaviors (Aleven, McLaren, & Koedinger, 2004), and genetics (Corbett, Kauffman, MacLaren, Wagner, & Jones, 2010).

What level of understanding is the goal? Cognitive Tutor uses a modified version of mastery learning to fit within the time constraints of a classroom (Corbett, McLaughlin, & Scarpinatto, 2000; Pane, Griffin, McCaffrey, & Karam, 2014). A maximum number of problems is set for each section. If a student masters every component in the section, then they are considered to have graduated. However, if a student fails to reach mastery, but has reached the maximum number of problems, then they are promoted to the next section. This way, students can continue to progress with the class without falling too far behind the rest of the students.

What is used to create the expert? Cognitive Tutor originally started with a math teacher and textbook to define the expert (Koedinger & Anderson, 1998). The model for each tutor is derived from a cognitive task analysis for domain knowledge and reasoning strategies that students need to complete the course (Corbett et al., 2010). The model is then codified into production rules that represent the skills and strategies (Aleven & Koedinger, 2002). Production rules are IF-THEN statements that take declarative knowledge as a condition for the IF and the THEN portion says what to do (procedural knowledge).

Tutoring Model

Strategies. Cognitive Tutor relies on the ACT-R Theory which conceptualizes knowledge as being either declarative or procedural (Koedinger & Anderson, 1998). Declarative knowledge includes such things as facts and concepts and is encoded through activities like reading (Corbett, McLaughlin, & Scarpinatto, 2000). Procedural knowledge is performance knowledge that can only be measured by doing and requires applying declarative knowledge. As such, Cognitive Tutor focuses on teaching procedural knowledge by doing (Koedinger & Anderson, 1998). Thus, students engage in a cognitive apprenticeship with Cognitive Tutor wherein students become experts by practicing the declarative and procedural knowledge encoded in the system (Koedinger & Corbett, 2006; Ritter, Anderson, Koedinger, & Corbett, 2007).

In helping students learn by doing, Cognitive Tutor developed three design goals: (a) support students in applying algebra to real-world problems (contextualizing), (b) support reasoning among multiple representations (tables, graphs, symbolic expressions, and natural language), and (c) employ modern computational tools.

Tactics. The following section will talk about the tutoring tactics of Cognitive Tutor including its hints, feedback, anti-gaming measures, use of self-explanation, and a stand-alone Help Tutor.

Hints. Students can request help at any time (Koedinger & Anderson, 2008), however if a student gets more than two errors on a step, help is volunteered to the student (Aleven & Koedinger, 2001). The first hint is vague to maximize the opportunities a student must reason it out or to encourage the student to seek a teacher for help (Corbett, McLaughlin, & Scarpinatto, 2000). If a student requests additional hints, the hints become more specific until a suggested action is presented. There are usually five to eight levels of hints provided for each problem (Aleven & Koedinger, 2001). While students do not have to follow a specific solution path when solving a problem, if they ask for help, the hints will guide students along a recommended path (Corbett, McLaughlin, Scarpinatto, & Hadley, 2000).

When a student asks for help on what to do next, Cognitive Tutor has three general levels of advice. First, it will remind or advise the student of an appropriate goal. Second, it will provide general advice on how to solve the goal. Last, it will provide a concrete suggestion on how to solve the goal given the current context (Corbett, McLaughlin, & Scarpinatto, 2000).

Feedback. An initial form of feedback given to students is when they submit answers. Cognitive Tutor will either accept a step as correct or it will flag them for errors (Corbett, McLaughlin, & Scarpinatto, 2000). This process is automatic, and the student does not need to request it (Aleven & Koedinger, 2001). After that, some additional feedback can be given. For example, if a student performs a common error, an error message is displayed explaining the error to the student (Aleven & Koedinger, 2002).

Cognitive Tutors also provide metacognitive feedback to students. This type of feedback is given based on the student's learning behaviors rather than the accuracy of their responses (Roll, Aleven, McLaren, & Koedinger, 2011). The content of the feedback message explains what the desired learning behavior looks like. For example, if a student is avoiding asking for help and the system detects this, it can provide feedback that lets the student know they should be asking for help on this problem.

Anti-gaming Measures. The developers of Cognitive Tutor provided a lot of research on anti-gaming measures within ITSs. For example, a two second delay was implemented between each hint level to discourage students from clicking the hint button repeatedly until they reach a bottom-out hint (Aleven & Koedinger, 2001).

Another way Cognitive Tutor addresses gaming behavior is through a pedagogical agent called Scooter the Tutor (Baker et al., 2006). Scooter is an animated dog that observes students as they interact with the system. If Scooter detects gaming behavior, he looks increasingly unhappy. The implementation of Scooter had three goals (Baker et al., 2006). First, there should be some representation of how much a student is gaming. Second, Scooter's emotional response was meant to invoke social norms so as to discourage the behavior. Third, to allow the students another opportunity to learn material that is being gamed by providing supplemental material.

The supplemental material given by Scooter has three levels of multiple-choice questions (Baker et al., 2006). Students have one chance to answer each question correctly; if they are correct they are taken back to the main question. The first two questions test the student's understanding of one of the concepts or the role the gamed step plays in solving the overall problem. If a student gets to the third level, the question is still related but is very easy to avoid the student being stuck in an indefinite loop. If the student answers that last question incorrectly,

Scooter assumes the student is trying to game him, gets angry, and asks the student to try and get the questions right on the first try. The problem is then flagged to receive supplementary help in the future.

In one study with Cognitive Tutor, the prevalence of gaming behavior was recorded (Baker et al., 2004). The study involved observing student behavior as they interacted with the system while observers noted the first behavior exhibited in fixed time intervals. Of the behaviors recorded, it was found that students were on-task 82% of the time. Of the 18% off-task behaviors, 3% was due to gaming. This may seem like a relatively small percentage, but it should be known that 24% of the students were observed gaming at least once. Those who gamed had lower than average academic achievement and the least prior knowledge of the material. Even though efforts are made to help these students engage in productive behaviors, the authors of Cognitive Tutor found that as new anti-gaming measures are implemented, students find new ways to game (Baker et al., 2006).

Self-explanation. The Geometry Cognitive Tutor implemented self-explanation as a required step of problem solving in order to promote learning (Aleven, Popescu, & Koedinger, 2001). Students must justify their answers by stating the geometric definition or theorem that explains their step. If it is the wrong justification, the system provides feedback. The definitions and theorems are selected from a drop-down menu (Aleven & Koedinger, 2002). They found that while self-explanation does not increase the rate of knowledge acquisition, it does change the nature of the knowledge acquired (Aleven & Koedinger, 2002).

Help Tutor. As was mentioned earlier, Cognitive Tutor provides metacognitive feedback to students for help-seeking behaviors. This feedback was formed into a separate, add-on Tutor called the Help Tutor (Roll et al., 2011). The Help Tutor had 80 production rules that could

classify student behavior as either desired or undesired help-seeking behaviors. When a behavior is classified as undesired, immediate feedback is given such as "Take your time." The feedback is meant to be domain-independent so that it can be used with any tutor. When feedback is given to the student, it is distinguished by font and color from regular hints and feedback.

Student Model

How is the student model created? Cognitive Tutors uses various techniques to perform model and knowledge tracing (Koedinger & Anderson, 1998). Model tracing follows the student's individual solution path along the cognitive model for the domain. This lets the tutor know which path the student has taken and the student's current solution state so as to provide contextualized feedback to the student (Corbett, McLaughlin, & Scarpinatto, 2000). Knowledge tracing also follows the solution path to map the strengths and weaknesses of the student's acquisition of production rules and to provide tailored instruction (Koedinger & Corbett, 2006).

For knowledge tracing, each production rule is in one of two states: learned or unlearned (Corbett, McLaughlin, & Scarpinatto, 2000). A production rule can be learned through classroom activities or by applying the rule in a problem. A rule cannot be unlearned. The state of the rule is modeled probabilistically as a student does not always apply rules even if they know it (called a slip parameter) or they can guess and arrive at an answer correctly (called a guess parameter). For a student to have mastered a rule, the probability that it is in the learned state must be greater than or equal to 0.95. The probabilities of each production rule are updated at each opportunity it is used. The estimates of student knowledge are displayed to the student in what is referred to as an open student model (Aleven & Koedinger, 2002). The knowledge is displayed as a skillometer.

What data is collected? Not much was found that described the fine-grained data that Cognitive Tutor collects to inform its student model. However, some of the research explained their process of measuring student affect.

Six emotional states were identified: boredom, confusion, engaged concentration (flow), frustration, and "?" which represents everything else (Baker et al., 2012). In a naturalistic observation from the same study, the frequencies of each behavior were determined by an observer. Boredom was recorded 5.9% of the time, engaged concentration 84.5%, frustration 0.9%, confusion 1.8%, and the remaining behaviors were classified as "?". The time frames in which these behaviors were recorded were synced with the log files of the students to see if there were behavioral patterns that indicated each affect state.

Interface

The design decision was made that Cognitive Tutor would be a complete course that included textbooks, the software, supplementary materials, and teacher training (Ritter, Anderson, Koedinger, & Corbett, 2007). This decision allowed Cognitive Tutor to become a regular part of the classroom experience. It is designed so that two days a week, class time is spent on Cognitive Tutor, while the other three days are student-centered activities in class that involve group work and problem solving (Pane et al., 2014).

The interface that is described next refers to the Algebra Cognitive Tutor. Cognitive Tutor has several tools that it uses to help students (Koedinger & Corbett, 2006). For example, Algebra Cognitive Tutor uses a worksheet (much like one found in a spreadsheet) where students can solve problems step-by-step, a grapher to graph problems, and a solver which can calculate mathematical operations. These tools adapt as the student works with the program. As students

demonstrate their ability to perform lower-level skills, these tools automate those skills so that the student can focus on acquiring higher-level skills and concepts.

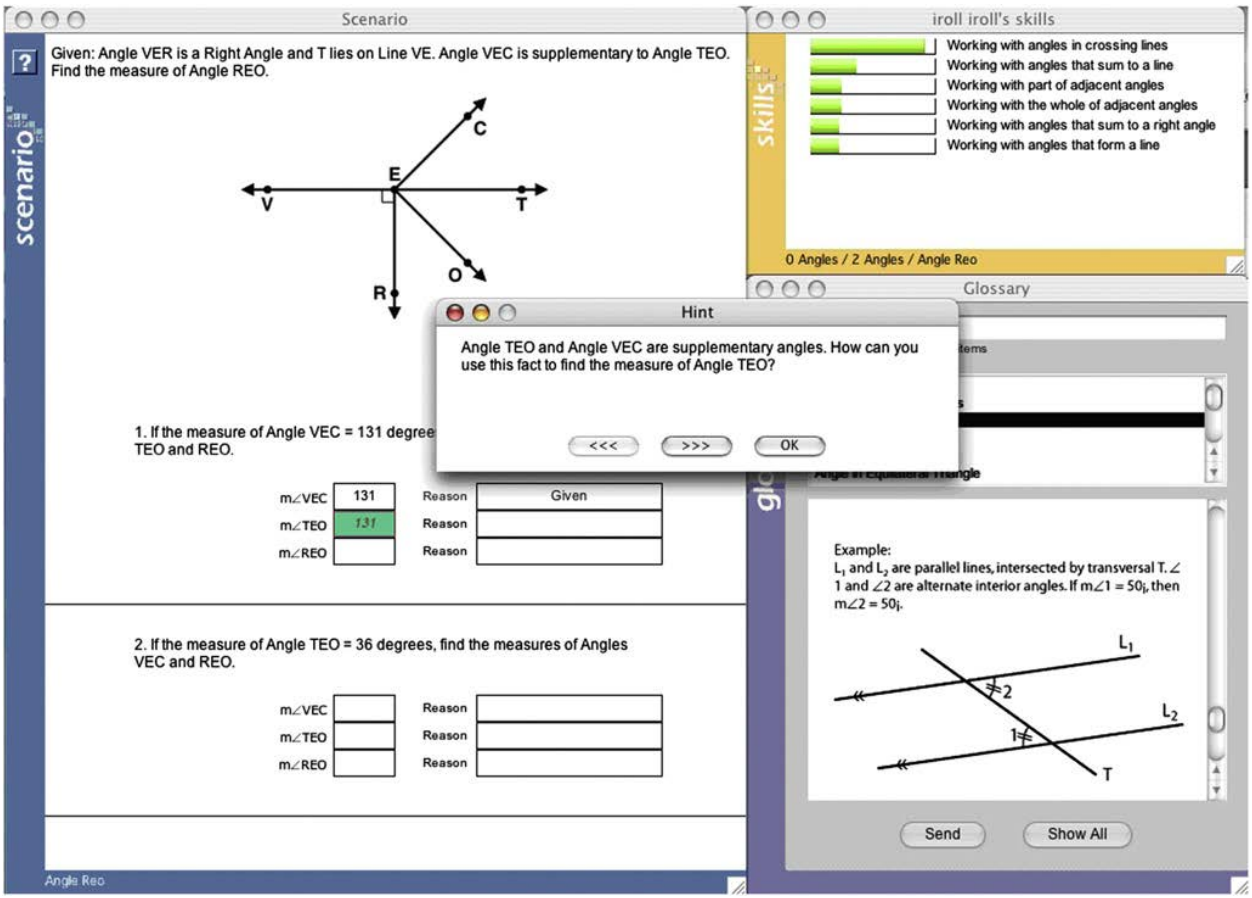

Figure 1. The interface of Algebra Cognitive Tutor (Rolle et al., 2011).

Learning Gains

Cognitive Tutor has demonstrated large learning gains within various domains. One study reported that the Programming and Geometry Tutors performed 1 standard deviation better than a control group and the Algebra I Tutor had learning gains ranging from .7 to 1.2 sigma (Ritter et al., 2007). Another study (Baker et al., 2006) showed that Cognitive Tutor performed 1 standard deviation better than classroom instruction and 1.75 standard deviations better than self-study.

In addition to increased learning, Cognitive Tutor has had some indication that it can also reduce the amount of time spent learning. One study (Corbett, McLaughlin, & Scarpinatto, 2000) found that students using Cognitive Tutor completed problem-solving activities in as little as one-third the time needed by students working in a conventional problem-solving environment.

References

Aleven, V. A., & Koedinger, K. R. (2002). An effective metacognitive strategy: Learning by doing and explaining with a computer-based Cognitive Tutor. *Cognitive science*, *26*(2), 147-179.

Aleven, V., & Koedinger, K. R. (2001). Investigations into help seeking and learning with a cognitive tutor. In R. Luckin (Ed.), *Papers of the AIED-2001 workshop on help provision and help seeking in interactive learning environments* (pp. 47-58), San Antonio, Texas: St. Mary's University.

Aleven, V., McLaren, B., Roll, I., & Koedinger, K. (2004). Toward tutoring help seeking. In J. Lester, R. Vicari, & F. Paraguacu (Eds.), *Intelligent Tutoring Systems* (pp. 227-239). Berlin/Heidelberg: Springer.

Aleven, V., Popescu, O., & Koedinger, K. R. (2001). Towards tutorial dialog to support self-explanation: Adding natural language understanding to a cognitive tutor. In J. Moore, C. Redfield, & W. Johnson (Eds.), *Proceedings of Artificial Intelligence in Education* (pp. 246-255). Amsterdam: IOS Press.

Baker, R. S., Corbett, A. T., Koedinger, K. R., Evenson, S., Roll, I., Wagner, A. Z., ... & Beck, J. E. (2006). Adapting to when students game an intelligent tutoring system. In Ikeda et al. (Eds.), *International Conference on Intelligent Tutoring Systems* (pp. 392-401). Berlin/Heidelberg: Springer.

Baker, R. S., Corbett, A. T., Koedinger, K. R., & Wagner, A. Z. (2004). Off-task behavior in the cognitive tutor classroom: When students game the system. In *Proceedings of the SIGCHI Conference on Human factors in Computing Systems* (pp. 383-390). New York, NY: ACM.

Baker, R. S., Gowda, S., Wixon, M., Kalka, J., Wagner, A., Salvi, A., ... & Rossi, L. (2012). Sensor-free automated detection of affect in a Cognitive Tutor for Algebra. In K. Yacef et al., (Eds.), *Educational Data Mining 2012* (pp. 126-133). Chania, Greece: International Educational Data Mining Society.

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*(2), 81-108.

Corbett, A., McLaughlin, M., Scarpinatto, K. C., & Hadley, W. (2000). Analyzing and generating mathematical models: An algebra II cognitive tutor design study. In G. Gautheier, C. Frasson, & K. VanLehn (Eds.), *International Conference on Intelligent Tutoring Systems* (pp. 314-323). Berlin/Heidelberg: Springer.

Koedinger, K. R., & Anderson, J. R. (1998). Illustrating principled design: The early evolution of a cognitive tutor for algebra symbolization. *Interactive Learning Environments*, *5*(1), 161-179.

Koedinger, K. R., & Corbett, A. (2006). Cognitive tutors: Technology bringing learning sciences to the classroom. In R. Sawyer (Ed.), *The Cambridge Handbook of: The Learning Science* (pp. 61-77). New York: Cambridge University Press.

Pane, J. F., Griffin, B. A., McCaffrey, D. F., & Karam, R. (2014). Effectiveness of cognitive tutor algebra I at scale. *Educational Evaluation and Policy Analysis*, *36*(2), 127-144.

Ritter, S., Anderson, J. R., Koedinger, K. R., & Corbett, A. (2007). Cognitive Tutor: Applied research in mathematics education. *Psychonomic Bulletin & Review*, *14*(2), 249-255

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

Reading Tutor

Reading Tutor helps children learn to read using expert reading interventions as students read sentences from a story aloud. By recording the student's audio, Reading Tutor can determine if the student is reading at a fluent level and use this information to provide tailored feedback and interventions. This was not always the case since Reading Tutor did not start off with a strong student model, but adjustments were made so that Reading Tutor could take advantage of the data it was collecting on students. A total of 23 articles were used for this review spanning the years of 1999-2013.

Domain Model

What content does this ITS cover? The Reading Tutor covers the domain of reading for first through fourth grades (Beck, Jia, & Mostow, 2003; Mostow & Aist, 1999).

What level of understanding is the goal? The Reading Tutor aims for oral reading fluency which means to read a text with speed, accuracy, and proper expression (Mostow & Duong, 2009). Reading Tutor measures oral reading fluency by using the number of words read correctly per minute and if the Reading Tutor can recognize the word as being read correctly without help or hesitation (Lalle, Mostow, Luengo, & Guin, 2013).

What is used to create the expert? The material for the Reading Tutor was adapted from a children's news magazine called *Weekly Reader* as well as other sources (Mostow & Aist, 1999). With that as source material, the oral part of the domain was created by using acoustic models that are trained on adult female speech (Mostow & Aist, 1999). The student's acoustic model is then compared to the adult's model as a standard of expert behavior. Since each sentence is known beforehand, there is no need for a general language model (Heiner, Beck, & Mostow, 2004).

Tutoring Model

Strategies. In a child's early years, the ability to identify words can create a bottleneck to fluent reading and comprehension. As such, Reading Tutor's strategies are based off the interventions used by expert reading teachers (Mostow & Aist, 1999; Zhang et al., 2008). For example, experts suggested the following steps (the Beck Battery): (a) give the context of the target vocabulary word in the story, (b) repeat the word to create a phonological representation, (c) explain the meaning of the word, (d) provide examples of the word in other contexts outside of the story, (e) allow children to practice the word in new contexts, and (f) repeat the word to reinforce the phonological representation (Heiner, Beck, & Mostow, 2006).

Tactics. The feedback, reading interventions, and use of cloze questions in Reading Tutor are discussed next.

Feedback. The Reading Tutor provides feedback to students in several ways. First, it provides flag feedback after hearing words by turning them green if they are read correctly (Mostow & Aist, 1999). Second, it can provide preemptive feedback with the mindset that preventing reading mistakes is better than fixing reading mistakes (Beck, Jia, Sison, & Mostow, 2003). So, if Reading Tutor determines that a student will have difficulty with a word (based off the student's previous performance and the length of the word), it will provide assistance by reading the word aloud before the student has a chance to. Third, if a student does not self-correct and moves on to the next word, Reading Tutor will interrupt and go back to the missed word (Mostow & Aist, 1999). Similarly, if a student takes too long (silent for more than seven seconds), Reading Tutor will intervene with assistance (Mostow & Aist, 1999). Last, Reading Tutor randomly offers praise at the end of a sentence if it was read correctly or if there was

improvement on the current performance compared to a previous one. However, praise is always offered at the end of reading a story (Mostow & Aist, 1999).

Reading Interventions. The reading interventions that the Reading Tutor implements are, in decreasing order of frequency (Heiner et al., 2004): saying the word, giving the word in context, autophonics (pronouncing the grapheme), sounding out the word using a video of a child's mouth sounding out phonemes, recuing, onset rime (which says the first phoneme, pauses, then says the rest of the phonemes), starts like, rhymes with, syllabify (breaking the word down into syllables and pronouncing each), showing a picture of the word, and making a sound effect related to the word.

The Reading Tutor aims to be a patient listener, giving plenty of space for the student to work through a sentence (Mostow & Aist, 1999). It will wait two seconds before providing back-channel feedback (e.g., "uh-huh"), four seconds before giving a hint, and seven seconds before prompting the student what to do.

Cloze Questions. To gauge reading comprehension, cloze questions were inserted into the stories (Mostow, 2004). These questions are generated by replacing a word in the question with a blank to fill in. The options the student can pick from fill in that missing word. There are four types of cloze questions: sight, easy, hard, and defined (Joseph, 2005). Sight questions are for very common words, while hard questions were for rarer words. Defined questions have words that need explanation for students. Performance on these questions was used to help gauge student engagement as part of the student model since it correlated well with performance on standardized tests of reading comprehension (Zhang, Mostow, & Beck, 2007).

Student Model

How is the student model created? Reading Tutor did not start with a student model when it was first made. Instead, it made a student model retroactively by collecting data first and then using the data to create a student model (Beck, Jia, & Mostow, 2003). However, their student model "does not have the strong reasoning about the user that distinguishes a classic intelligent tutoring system" (Beck, Jia, Sison, et al., 2003, p. 1). Rather than opting for a fine-grained analysis (e.g., knowing that *ph* makes an f sound in the word phone), the student model instead focuses on coarse-level assessment of student knowledge by relying on student input like latency and help request (Beck, Jia, & Mostow, 2003).

With the introduction of cloze questions, Reading Tutor was able to model student engagement in a process called engagement tracing (Joseph, 2005). Using student data on cloze questions, they found that students who spend about 4-7 seconds on a question had a higher likelihood of answering questions correctly (i.e., they are engaged) than those who spent a shorter amount of time (i.e., they are not engaged). Those who spent longer than 7 seconds saw a slight decrease in performance, but it could not be ascertained why that was nor could it be considered a sign of disengagement. The model could predict aggregate student performance very well with an r^2 of 0.954, however it did not account for individual differences like students who are very fast readers. Efforts were made to rectify that problem and the model correlated at 0.25 with learning gains on tests with a model reliability of 0.95. This model of student affect is unique in that it uses student data that is already normally collected by an ITS and does not require additional sensors.

Eventually, Reading Tutor was able to implement a form of knowledge tracing to be able to predict the likelihood that a student knows a particular word and to determine when students have mastered a word (Chang, Beck, Mostow, & Corbett, 2006).

What issues are there regarding the development of the student model? Reading Tutor's developers had trouble in creating a student model since reading is considered an ill-defined domain and there is no typed input that could be used (Beck, Jia, & Mostow, 2003). As such, the logging of student actions in the beginning was incomplete (e.g., the type of hint given was not recorded; Beck, Jia, Sison, et al., 2003).

What data is collected? The main form of student data in the Reading Tutor is the student's spoken responses. As such, many features of the student's speech are recorded. For example: the latency between words is measured, false starts, sounding out, and near misses are documented (Mostow & Aist, 1999) as well as student miscues like repetitions, insertions, or omissions (Mostow, Beck, Winter, & Wang, 2002). Eventually, the student's prosody was measured and compared to adult prosodic contours as a way of comparing latency, duration, mean pitch, or mean intensity (Mostow & Duong, 2009). The student's historical performance on a word is recorded as well (Mostow & Aist, 1999). Using these data, a student's fluency can be inferred from the student's accuracy and latency. Other system events are logged that timestamp when a student reads a story, sentence, word, or requests help (also measures how much help requested and the type of help; Beck, Jia, & Mostow, 2003). At a coarse level, the latency and help request data can be used to assess student knowledge (Beck, Jia, & Mostow, 2003).

Interface

Reading Tutor displays a sentence or fragment at a time that a student is supposed to read and is part of a story (Mostow & Aist, 1999). Students speak to the system and their responses are recorded and analyzed using automatic speech recognition software (Mostow & Aist, 1999). To aid the students, a simple, animated persona (a face) that "actively watches and patiently listens" (Mostow & Aist, 1999). It provides help to the student and highlights words as it speaks.

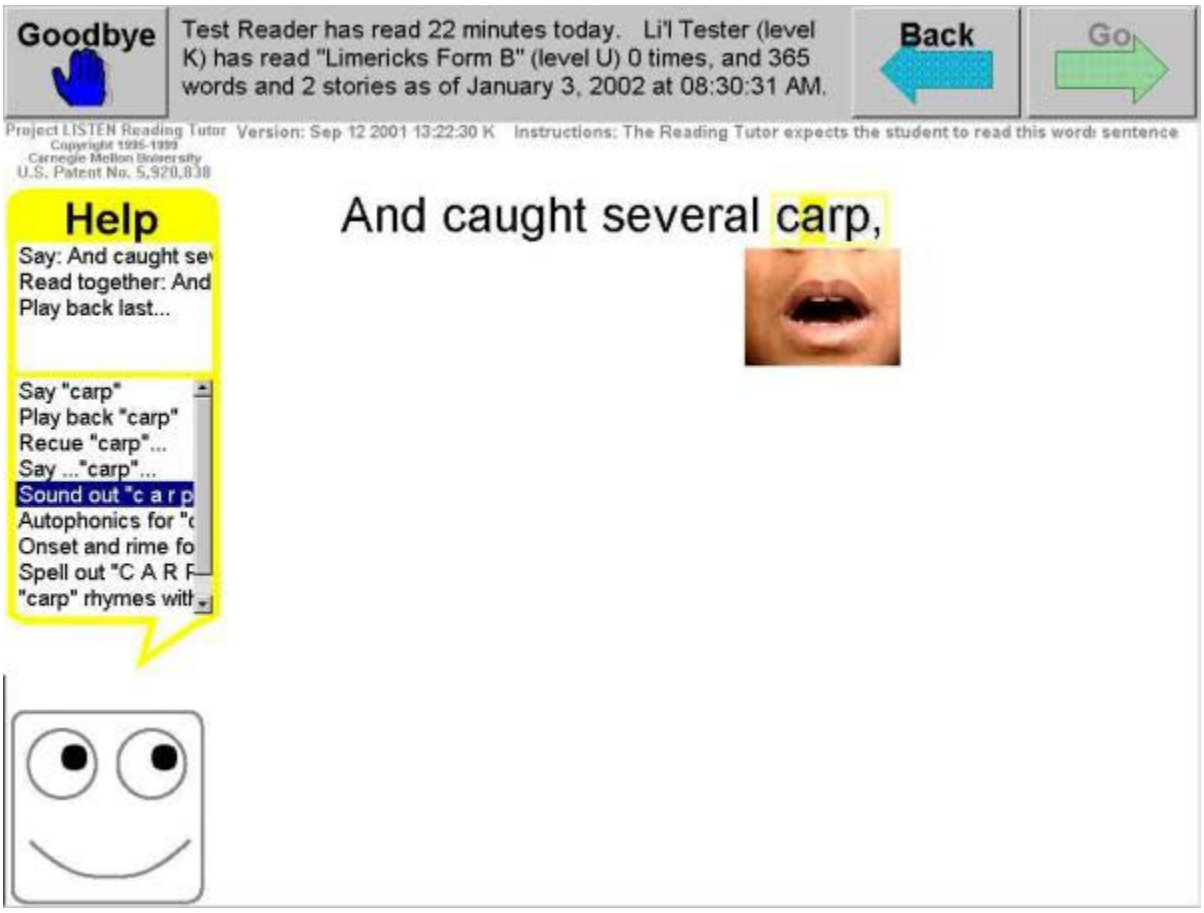

Figure 1. The interface of Reading Tutor (Aist, Kort, Reilly, Mostow, & Picard, 2002).

Learning Gains

No learning gains have been reported as a measure of Reading Tutor's effectiveness.

References

Aist, G., Kort, B., Reilly, R., Mostow, J., & Picard, R. (2002). Experimentally augmenting an intelligent tutoring system with human-supplied capabilities: adding human-provided emotional scaffolding to an automated reading tutor that listens. In *Multimodal Interfaces, 2002. Proceedings. Fourth IEEE International Conference on* (pp. 483-490). Pittsburgh, PA: IEEE.

Banerjee, S., Mostow, J., Beck, J., & Tam, W. (2003). Improving language models by learning from speech recognition errors in a reading tutor that listens. In *Proceedings of the Second International Conference on Applied Artificial Intelligence* (pp. 187-193). Kolhapur, India: AAI.

Beck, J., Jia, P., & Mostow, J. (2003). Assessing student proficiency in a Reading Tutor that listens. In P. Brusilovski, A. Corbett, & F. de Rosis (Eds.), *International Conference on User Modeling* (pp. 323-327). Berlin/Heidelberg: Springer

Beck, J. E., Jia, P., Sison, J., & Mostow, J. (2003). Predicting student help-request behavior in an intelligent tutor for reading. In P. Brusilovsky, A. Corbett, & F. De Rosis (Eds.), *International Conference on User Modeling* (pp. 303-312). Berlin/Heidelberg: Springer.

Chang, K. M., Beck, J., Mostow, J., & Corbett, A. (2006). A Bayes net toolkit for student modeling in intelligent tutoring systems. In M. Ikeda, K. Ashley, T. Chan (Eds.), *Intelligent Tutoring Systems* (pp. 104-113). Berlin/Heidelberg: Springer.

Heiner, C., Beck, J., & Mostow, J. (2004). Improving the help selection policy in a Reading Tutor that listens. In *InSTIL/ICALL Symposium 2004*. Pittsburgh, PA: CMU.

Heiner, C., Beck, J., & Mostow, J. (2006). Automated vocabulary instruction in a reading tutor. In M. Ikeda, K. Ashley, & T. Chan (Eds.), *Intelligent Tutoring Systems* (pp. 741-743). Berlin/Heidelberg: Springer.

Joseph, E. (2005). Engagement tracing: Using response times to model student disengagement. In C. Looi, G. McCalla, B. Bredeweg, & J. Breuker (Eds.), *Artificial Intelligence in Education: Supporting Learning through Intelligent and Socially Informed Technology*, *125*, 88-95. The Netherlands: IOS Press Amsterdam.

Lallé, S., Mostow, J., Luengo, V., & Guin, N. (2013). Comparing student models in different formalisms by predicting their impact on help success. In H. Lane, K. Yacef, J. Mostow, & P. Pavlik (Eds.), *International Conference on Artificial Intelligence in Education* (pp. 161-170). Berlin/Heidelberg: Springer.

Mostow, J. (2004). Some useful design tactics for mining ITS data. In *Proceedings of the ITS2004 Workshop on Analyzing Student-Tutor Interaction Logs to Improve Educational Outcomes* (pp. 20-28). Pittsburgh, PA: CMU.

Mostow, J., & Aist, G. (1999). Giving help and praise in a reading tutor with imperfect listening—Because automated speech recognition means never being able to say you're certain. *CALICO Journal, 16*(3), 407-424.

Mostow, J., Beck, J., Winter, S. V., & Wang, S. (2002). Predicting oral reading miscues. Retrieved from http://corescholar.libraries.wright.edu/knoesis/276

Mostow, J., & Duong, M. (2009). Oral reading prosody. In V. Dimitrova, R. Mizoguchi, B.D. Boulay, and A. Graesser (Eds.), *Proceeding of the 14th International Conference on Artificial Intelligence in Education (AIED2009)* (pp. 189-196). Brighton, UK. IAIED.

Zhang, X., Mostow, J., & Beck, J. E. (2007). Can a computer listen for fluctuations in reading comprehension? *Frontiers in Artificial Intelligence and Applications*, *158*, 495-503.

Zhang, X., Mostow, J., Duke, N., Trotochaud, C., Valeri, J., & Corbett, A. (2008). Mining free-form spoken responses to tutor prompts. In R. Baker, T. Barnes, & J. Beck (Eds.), *Educational Data Mining 2008* (pp. 234-241). Montreal, Canada: IEDMS.

SQL-Tutor

SQL-Tutor, as the name suggests, tutors the SQL programming language. While tutoring a programming language alone does not make SQL-Tutor stand out from similar systems, it is unique in its approach and underlying structure being the first of its kind to implement constraint-based modeling. Nineteen articles from the years of 1996-2016 provided the basis for this review.

Domain Model

What content does this ITS cover? SQL-Tutor is an ITS for the SQL programming language for upper-level undergraduate students (Mitrovic, 1997; Zhou, Wang, & Ng, 1996). More specifically, it covers the SELECT statement and querying databases (Mitrovic & Ohlsson, 2016).

What level of understanding is the goal? SQL-Tutor assumes that students were exposed to concepts of database management prior to interacting with the tutor. As such, it is meant to be a practice environment that complements but does not substitute the classroom (Mitrovic, 1997) and aims for mastery of the topics (Mitrovic, Martin, & Mayo, 2002).

What is used to create the expert? The expert for SQL-Tutor is based on over 600 IF-THEN statements called constraints (Mitrovic & Martin, 2003) which were created by the author (Mitrovic & Ohlsson, 1999). These constraints are similar in structure to production rules (sharing an IF-THEN structure), but they differ in purpose and nature. Production rules state that IF x is true, THEN do this thing. Constraints say that IF condition X is true, THEN condition Y better also be true. These IF-THEN statements can be written as an ordered pair as well (C_r, C_s) where C_r is a relevance condition and C_s is a satisfaction condition (Mitrovic, 1997). Constraints are written such that they represent correct domain knowledge which is the same

across all student populations and are general enough that they can be tested against any problem (Mitrovic & Ohlsson, 1999). Ideal answers are written for problems so that the tutor can trace a correct constraint path, but constraints can recognize correct student solutions even if they are not ideal so long as no constraints are violated. If a constraint is violated, that means the student has a misconception or buggy knowledge. Constraints are limited in that they cannot diagnose what caused the error, but instead focuses on teaching correct domain knowledge since feedback can be given for violated domain principles (Mitrovic, 2010). The model, called Constraint-based Modeling (CBM) is based on Ohlsson's theory on learning from performance errors (Ohlsson, 1996) and SQL-Tutor serves as the first constraint-based tutor. More explanation of CBM is given in the Student Model section below.

Tutoring Model

Strategies. SQL-Tutor's developers claim SQL-Tutor is an "integration of ideas from several theories of learning" (Mitrovic, 1997) including ACT-R (Anderson, 1993), impasse-driven learning (VanLehn, 1988), general architecture of intelligence SOAR (Rosenbloom, Laird, Newell, & McCarl, 1991), and Ohlsson's theory on learning from performance errors (1996). Learning has been conceptualized to happen in three phases (Mitrovic, 1997). First, a relatively short phase happens wherein students acquire new knowledge in declarative form (also known as conceptual learning). The second phase is longer and involves practicing and applying existing knowledge and acquiring new knowledge. This has been called procedural knowledge by Anderson (Anderson, 1993). VanLehn (1988) says the process of acquiring new knowledge happens because student reach an impasse and must use the resources available to them to solve the problem, while Ohlsson (1996) says that students can also learn from their errors.

SQL-Tutor also uses the zone of proximal development in determining the level of difficulty for the next problem (Mitrovic, Martin, & Mayo, 2002). SQL-Tutor predicts the number of errors a student will make then chooses a problem where a student will not make too few or too many mistakes.

In summary, SQL-Tutor is a guided discovery environment (Mitrovic, 1997). The tutor and student work cooperatively to achieve learning goals. The student can have the choice of being in control of problem and topic selection while the tutor will create a learning graph that the student can traverse. The student can also choose to let the system be in control of topic selection and determining what the student needs to learn (Zhou et al., 1996).

Tactics. SQL-Tutor's use of feedback will be discussed as a tutoring tactic below.

Feedback. Feedback generated by the system originally had five levels (Mitrovic, 1997; Mitrovic & Ohlsson, 1999): (a) positive or negative feedback to let the student know if the solution is correct or not, (b) an error flag, (c) hints (gives information about the type of error), (d) a partial solution (gives the correct content of a clause in the SELECT statement), and (e) a complete solution (gives a full, correct solution). Later, a sixth level of feedback was added called "all errors" where every error made was reported to the student instead of addressing one at a time like is typical for the system (Mitrovic & Martin, 2000; Mitrovic & Ohlsson, 1999).

As students are working on problems, feedback is offered after a student submits their answer making the feedback answer-based instead of step-based (Mitrovic & Ohlsson, 1999). SQL-Tutor will never offer more than the hint level of feedback, but students can request partial/complete solutions by clicking a button (Mitrovic & Ohlsson, 1999).

Feedback is generated for a student by determining which constraints apply to the current problem. Then, using the constraints to build an answer as close as possible to the student's

attempt, SQL-Tutor can provide feedback by comparing the constraint paths to see which are being violated by the student's answer.

The content for feedback was created intuitively at first with no supporting theory behind it. So, using Ohlsson's theory (1996), the feedback was changed to make sure it told the student what the error was, what constitutes the error, and reiterate the domain principles that were violated (Mitrovic, 2010).

Student Model

How is the student model created? Since there can be no correct answer that violates domain principles, constraint-based modeling holds that the diagnostic information of student errors is not hidden in the sequence of student's actions but in the problem state a student arrives at (Mitrovic, 1997). This allows CBM to be more permissive than model tracing since it adopts an innocent-until-proven-guilty mindset. If no constraints are violated, then the answer is assumed to be correct whereas model tracing would usually mark an unrecognized answer as wrong (Mitrovic et al., 2007).

Constraints come in two forms in SQL-Tutor (Mitrovic & Ohlsson, 2016). First, there are syntax constraints which checks for the syntactical correctness of the student's solution. Second, there are semantic constraints which check to see if the solution is correct for the current problem by comparing it to the ideal solution and the constraints that it uses. Both types of constraints are problem-independent which makes it easier to add new problems.

The student model itself is generated by propagating the student answer through the relevance (C_r) and satisfaction (C_s) networks of the constraints (Mitrovic, 1997; Martin, 1999). In other words, the student solution goes through all the C_r portions of the constraints to see which constraints apply to the current solution. Then, it works through the C_s portion of each

constraint to see if anything is violated. A list of all satisfied and violated constraints is then generated.

The model uses these two lists to probabilistically determine if the constraint is mastered (Mitrovic et al., 2002). There are three heuristics for updating the probabilities. First, if a constraint is satisfied, then the probability it is mastered increases by 10% of 1 minus its current mastered probability (1-P(Masteredc = Yes)). Second, if the constraint is violated and no feedback is given, then the probability it is mastered decreases by 20%. Third, if the constraint is violated but feedback is given, then the probability of it being mastered increases by 20% of (1-P(Masteredc = Yes)).

What issues are there regarding the development of the student model? Creating a student model is a complex process and many modeling approaches insist on cognitive validity and completeness (Mitrovic, 1998). The authors of SQL-Tutor argue that the student does not need to be cognitively valid and complete and that you can get away with a model that is good enough citing that human teachers do not have valid and complete cognitive models of their student, yet they still do well at their job (Mitrovic, 1998).

What data is collected? Each time a constraint is applicable to a solution, properties of that constraint are incremented (a used counter, a violated counter, and a relevant counter) (Martin, 1999; Mitrovic, 1999). This information creates a long-term history of each constraint, but a complete history cannot always be used since students are supposed to improve with time (Mitrovic, 2010). So instead, a window of time is used to determine the estimate of student knowledge.

In addition to performance on constraints, SQL-Tutor collects information about the student, such as reports of their self-efficacy. Students are asked during their first session

whether they are novices, intermediate, or experienced (Mitrovic et al., 2002). As the student uses the tutor, their level is updated. For example, if a student solves two or more problems that are at or above their current level, they will be promoted to the next level. This information is used in determining the level of difficulty for the next problem.

Difficulty of problems is also dynamically recorded based off student performance (Hull & du Boulay, 2015). Problems receive a difficulty level ranging from 1-9 (easy to hard). With each student attempt on a problem the difficulty level of all problems is reevaluated based on the student's performance.

Interface

SQL-Tutor's web-based interface is designed to promote problem solving through a learning-by-doing format (Mitrovic, 1997). It does so by reducing the memory load of students by displaying the database's schema and the current problem, providing the basic structure of the query, and providing explanations of the elements of SQL (Mitrovic, 1997). These parts of the interface are located in three separate windows that are always visible.

Another window was created that showed a skillometer to each student. The skillometer opens up the student model to show the student the system's estimates of what they know (Mitrovic & Martin, 2002). The constraints were categorized into the six clauses of the SELECT query since displaying each constraint is not feasible (Mitrovic & Martin, 2003). The mastery of each clause is based on aggregate information for each constraint.

SQL-Tutor integrated a pedagogical agent called Smart Egg into their tutoring (Mitrovic & Suraweera, 2000). Smart Egg is a cartoon-like character that explains the system's functions, provides feedback, and lets the student know of additional ways they can receive help. Students appeared to have liked Smart Egg, as the approval rating of the entire system had a mean of 4.5

(out of 5) compared to a control setting without Smart Egg of 3.83 (Mitrovic & Suraweera, 2000). Another evaluation of the system (without Smart Egg) showed that 50% of students agreed that they enjoyed working with SQL-Tutor (Mitrovic & Ohlsson, 1999).

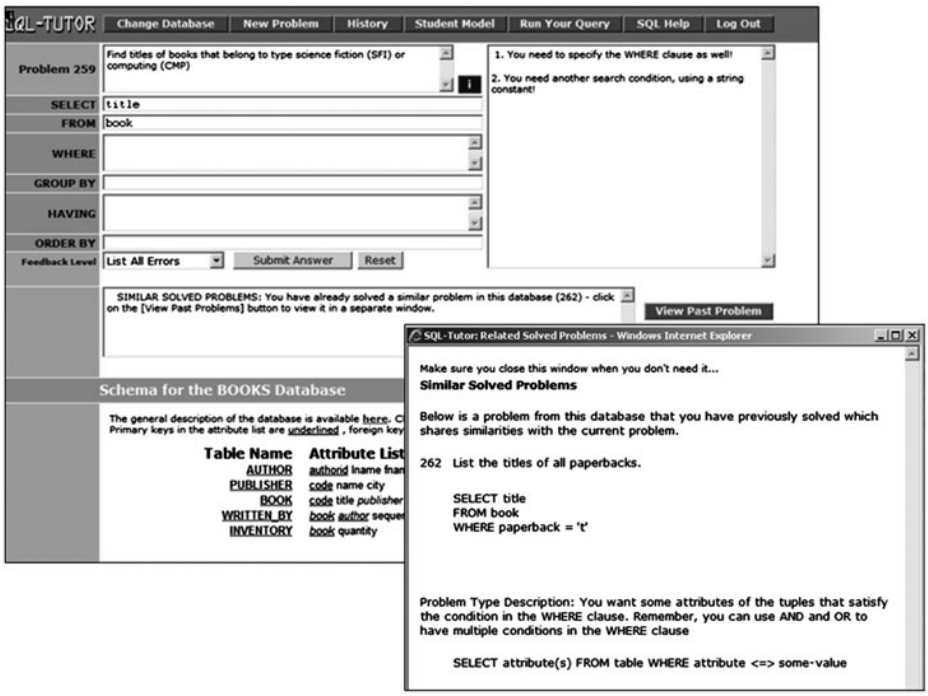

Figure 2. The SQL-Tutor interface (Hull & du Boulay, 2015).

Learning Gains

SQL-Tutor suggests that the use of effect sizes is not the most appropriate measure of learning gains since training histories are different for each ITS (Mitrovic & Ohlsson, 1999). Instead, they recommend looking at and comparing learning curves to see how quickly knowledge is learned. For example, by plotting the probability that a student will violate a constraint against the number of times the constraint is relevant for a problem, you can get a learning curve that demonstrates the power law of practice (Martin et al., 2005). The argument is made that those systems that provide a better power law fit may be superior. Using learning curves have their own disadvantages as well such as the difficulty of problems causing steeper learning curves. To compare one model to another, it would need the exact same problem sets.

They suggest that some way of normalizing the learning curves is needed (Martin et al., 2005). Even with the emphasis on learning curves, SQL-Tutor has reported an effect size. One study reported an effect size of 0.66 sigma (Mitrovic et al., 2002). Unfortunately, this was a non-controlled study where 20 students self-selected to participate to use SQL-Tutor for two hours before they took the final exam.

References

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

Hull, A., & du Boulay, B. (2015). Motivational and metacognitive feedback in SQL-Tutor. *Computer Science Education*, *25*(2), 238-256.

Martin, B. (1999). Constraint-based modelling: Representing student knowledge. *New Zealand Journal of Computing*, *7*(2), 30-38.

Martin, B., Koedinger, K. R., Mitrovic, A., & Mathan, S. (2005). On using learning curves to evaluate ITS. In C. Looi, G. McCalla, B. Bredeweg, & J. Breuker (Eds.), *Proceedings of the 12th International Conference on Artificial Intelligence in Education* (pp. 419-428). The Netherlands: IOS Press Amsterdam.

Mitrovic, A. (1997). SQL-Tutor: a preliminary report. http://hdl.handle.net/10092/2994

Mitrovic, A. (1998). Experiences in implementing constraint-based modeling in SQL-Tutor. In B. Goettl, H. Halff, C. Redfield, & V. Shute (Eds.), *Intelligent Tutoring Systems* (pp. 414-423). Berlin/Heidelberg: Springer.

Mitrovic, A. (2010). Modeling domains and students with constraint-based modeling. In R. Nkambou, J. Bourdeau, & R. Mizoguchi (Eds.), *Advances in Intelligent Tutoring Systems* (pp. 63-80). Berlin/Heidelberg: Springer.

Mitrovic, A., & Martin, B. (2000). Evaluating the effectiveness of feedback in SQL-Tutor. In *Proceedings International Worksho*p on Advanced Learning Technologies (pp. 143-144). New Zealand: IEEE.

Mitrovic, A., & Martin, B. (2002). Evaluating the effects of open student models on learning. *AH*, *2002*, 296-305.

Mitrovic, A., & Martin, B. (2003). Scaffolding and fading problem selection in SQL-Tutor. In *Proceedings of the 11th International Conference on Artificial Intelligence in Education* (pp. 479-481). Sydney, Australia: IAIED.

Mitrovic, A., Martin, B., & Mayo, M. (2002). Using evaluation to shape ITS design: Results and experiences with SQL-Tutor. *User Modeling and User-Adapted Interaction*, *12*(2), 243-279.

Mitrovic, A., Martin, B., & Suraweera, P. (2007). Intelligent tutors for all: Constraint-based modeling methodology, systems and authoring. *IEEE Intelligent Systems 22*(4), 38-45.

Mitrovic, A., & Ohlsson, S. (1999). Evaluation of a constraint-based tutor for a database language. Retrieved from http://hdl.handle.net/10092/327

Mitrovic, A., & Ohlsson, S. (2016). Implementing CBM: SQL-tutor after fifteen years. *International Journal of Artificial Intelligence in Education*, *26*(1), 150-159.

Mitrovic, A., & Suraweera, P. (2000). Evaluating an animated pedagogical agent. In G. Gauthier, C. Frasson, K. VanLehn (Eds.), *Intelligent tutoring systems* (pp. 73-82). Berlin/Heidelberg: Springer.

Ohlsson, S. (1996). Learning from performance errors. *Psychological Review*, *103*(2), 241-262.

Rosenbloom, P. S., Laird, J. E., Newell, A., & McCarl, R. (1991). A preliminary analysis of the Soar architecture as a basis for general intelligence. *Artificial Intelligence*, *47*(1-3), 289-325.

VanLehn, K. (1988). Toward a theory of impasse-driven learning. In H. Mandl & A. Lesgold (Eds.), *Learning issues for intelligent tutoring systems* (pp. 19-41). New York: NY: Springer.

Zhou, G., Wang, J. L., & Ng, P. A. (1996). Curriculum knowledge representation and manipulation in knowledge-based tutoring systems. *IEEE Transactions on Knowledge and Data Engineering*, *8*(5), 679-689.

Wayang Outpost

Wayang Outpost makes full use of multimedia as it engages students in a game-like environment where students demonstrate math skills as they attempt to solve problems relating to their adventure. The developers of Wayang Outpost also address issues like gender and ethnicity and the role they play in a student's self-efficacy towards mathematics. A total of 27 articles from 2003-2014 were used as the basis for review.

Domain Model

What content does this ITS cover? Wayang Outpost was designed to tutor geometry specifically for the SAT (Arroyo, Beal, et al., 2003). It was eventually extended to include statistics, and algebra (Arroyo, Mehranian, & Woolf, 2010).

What level of understanding is the goal? Wayang Outpost not only teaches students to a level of mastery (Arroyo, Woolf, & Shanabrook, 2012; Wixon et al., 2014), but also to a level of fluency (Arroyo, Royer, & Woolf, 2011). Fluency is defined here as "the speed with which students either retrieve or calculate answers to basic math" (Arroyo et al., 2012). In other words, not only are students expected to be able to demonstrate the skills of the domain, but they also must do so quickly. While there is a focus on speed, it is not intended to be a regression to the skill and drill methods of teaching. The intent is to make the skills more readily available to students as they move onto more complex problem-solving activities.

What is used to create the expert? The expert is based on SAT released problems which are then embedded into an educational game environment where students must solve the problems.

Tutoring Model

Strategies. Wayang Outpost is based on a cognitive apprenticeship between the tutor and the student (Wixon et al., 2014). There are several phases to this cognitive apprenticeship which are apparent in the game environment: modeling, practice with coaching, scaffolding, and reflection. The cognitive apprenticeship also utilizes the zone of proximal development to ensure students are receiving enough of a challenge by making sure that each problem has a small number of mistakes and limiting the amount of time spent on a problem (Arroyo, Beal, et al., 2003; Wixon et al., 2014).

Wayang Outpost is unique in that its tutoring strategies relies on educational gaming (Tai, Arroyo, & Woolf, 2013) and multimedia learning theory (Wixon et al., 2014). Wayang Outpost uses in-game characters to act as role models for students (especially for girls) and video-game-like adventures as a motivational strategy for students (Arroyo, Beal, Murray, Walles, & Woolf, 2004; Arroyo, Walles, Beal, & Woolf, 2003).

Wayang Outpost addresses the issue of transfer to make sure that students are able to apply what they learn in several contexts (Arroyo, Beal, et al., 2003). Problems are organized into short-term transfer and long-term transfer problems. Short-term transfer problems include variations of the SAT problem (with different numbers and minor changes in figures) and are multiple choice while long-term transfers involve more of the gaming component. Long-term transfer problems are couched in animated adventures where students solve real-world math problems which are part of a storyline for the game. The problems that are solved in the adventures rely on the same skills that are in the short-term transfer sets, but are more sophisticated in that they are not multiple choice. How well students do on the adventure

problems is a sign of how well the skills transferred to a new context (Arroyo, Walles, Beal, & Woolf, 2003).

Another unique feature of Wayang Outpost is its focus on gender and minority differences (Tai, Arroyo, & Woolf, 2013). Recognizing that girls and other minorities tend to have lower self-efficacy regarding math skill (Arroyo, Woolf, et al., 2010), Wayang Outpost addresses this issue in one way by making the scientists in the game based off actual female scientists to make the game female-friendly. Additional pedagogical support (i.e., pedagogical agents) are also sensitive to gender and minority differences by having agents that are gendered and can be of different racial backgrounds (Tai et al., 2013). Wayang Outpost also downplays the role of performance in favor of learning by promoting effort and perseverance and suggesting to the student that intelligence is malleable (Arroyo, Woolf, et al., 2010). Their research suggests that the needs of boys and girls could be different, and that the student model should consider gender in its approach for interventions (Arroyo, Woolf, Cooper, Burleson, & Muldner, 2011).

Tactics. The tutoring tactics of Wayang Outpost are now discussed. Specifically, the hinting, adaptiveness, use of multimedia, and feedback are discussed.

Hints. Wayang Outpost implements two distinct types of hints with one being based on an analytical approach and the other based on a spatial estimates approach (Arroyo, Beal, et al., 2003). The analytical approach focuses on computation and numeric help. The spatial estimates approach helps to teach the student heuristics and estimation techniques. The type of hint given is chosen based on estimates of how effective that hint would be for a student. This can be ascertained by a spatial ability test taken before interacting with the tutor (Arroyo, Woolf, & Beal, 2006).

Hints go from general to more specific and can be skipped (Arroyo et al., 2004). When a hint is skipped, the next hint will provide a summary of previous hint while giving a full explanation of the current hint. Each new hint is delivered with multimedia to simulate what a human tutor might do (e.g., drawing, highlighting); Arroyo, Beal, et al., 2003).

Adaptive. There is an adaptive module within Wayang Outpost that allows the sequencing of problems (Arroyo et al., 2006). Initially, experts determined the difficulty of each problem and the module would select based off that (Arroyo Mehranian, et al., 2010). However, the expert-defined difficulty of a problem did not necessarily line up with the perceived difficulty students had of the problem. To reconcile the two, Wayang Outpost started using data (number of attempts to solve a problem, amount of time spent on a problem, and the amount of help required or requested) to infer the student's perceived difficulty (Arroyo Mehranian, et al., 2010). Later, students were even asked if they wanted to see easier or harder problems (Tai et al., 2013).

Multimedia. While using multimedia to present hints, feedback, and problems may be useful in motivating students it comes with some costs. Problems are much more sophisticated in that animations and sound are used and become costlier because of that. This limits the number of problems Wayang Outpost can present since presenting the same problems with different numbers can be repetitive for students (Arroyo et al., 2004).

Feedback. The first round of feedback that students can get is flag feedback where a question is marked red if it is wrong and green if it is correct (Arroyo, Woolf, et al., 2010; Arroyo, Woolf, et al., 2011). Additional feedback is given to the student via the pedagogical agent present (Arroyo Mehranian, et al., 2010). These pedagogical agents are considered learning companions in that they are meant to be learning with the student and not be giving

instruction, so the nature of their feedback is different from what one might expect. Based on the student's effort model, the learning companions will give affective and metacognitive feedback. For example, the learning companions had about 50 different messages that emphasize the malleability of intelligence and the importance of effort and perseverance (Arroyo, Woolf, et al., 2010). The learning companions also displayed emotions in their response to effort. If a student is deemed to not have exerted much effort, the learning companion will seem unimpressed (Tai et al., 2013). Messages from the learning companions could also be metacognitive in nature by helping the students with problem-solving strategies (Arroyo, Woolf, et al., 2010).

Student Model

How is the student model created? Wayang Outpost started with 70 distinct problems which was thought to be too few to require a sophisticated algorithm that adapted problems based on skill mastery level (Arroyo et al., 2004). Instead, focus was spent on optimizing problem sequencing so that students performed slightly worse than the average behavior expected for a problem (called the desired problem difficulty). This level of difficulty was calculated from the log files of previous users based on the number of hints seen and the mistakes that were made (Arroyo et al., 2004). A skill's mastery levels were still inferred based on student performance though by using a variation of item response theory to perform knowledge tracing (Ferguson, Arroyo, Mahadevan, Woolf, & Barto, 2006; Johns & Woolf, 2006).

However, Wayang Outpost continued to be developed to focus on the student's effort in solving problems. In 2010 (Arroyo Mehranian, et al., 2010), an effort-based model was created that was able to accomplish more than knowledge tracing could on its own. Based on the number of hints requested, the number of attempts to solve a problem, and the time it took to

solve a problem, Wayang Outpost can discern between behaviors that relate to student engagement in addition to behaviors related to skill mastery (Arroyo Mehranian, et al., 2010). For example, this model could detect when a student demonstrated mastery without much effort. In response, the tutor can increase the desired problem difficulty and maybe show the student's learning progress. Another example could be that the student is avoiding hints but trying hard. In that case, Wayang Outpost would respond by reducing the problem difficulty and maybe offering hints on the next incorrect answer. This model further demonstrates Wayang Outpost's focus on effort over performance.

What issues are there regarding the development of the student model? Having based their tutor on released SAT items, some difficulties emerged in creating a student model. First, there is no clear idea of what the problem difficulty will be on future versions of the SAT nor which skills will be implemented. Second, the skills that are used are not used in more than a few problems which makes estimating student knowledge difficult (Arroyo et al., 2004). These issues may have been encouragement for shifting from a traditional knowledge tracing model to one that focuses more on effort.

Another issue common with mastery learning is the idea of being stuck in a mastery loop—that is, unable to attain mastery. Wayang Outpost addresses this issue by exiting the mastery loop under three conditions (Arroyo Mehranian, et al., 2010): when the topic mastery is reached, when the tutor has persistent difficulty in finding a problem of appropriate difficulty level, or when the student reaching the maximum amount of time or problems allowed to spend on a topic.

What data is collected? Wayang Outpost starts with a pretest to determine the initial student model by finding out which skills are already mastered (Ferguson et al., 2006). The

pretest also serves as a posttest comparison to be able to measure learning gains. Wayang Outpost also collects information on spatial reasoning and math fact retrieval skills prior to the student using the system (Arroyo et al., 2006).

Once the student begins tutoring, every interaction between the student and the tutor is logged (Arroyo & Woolf, 2005). Performance variables such as time spent on a problem, the number of problems seen, the speed of a response, and the number of hints viewed are all part of the log. Non-performance variables like affect are also recorded like when students are surveyed about how they are currently feeling while using the system (Arroyo, Woolf, Royer, & Tai, 2009).

All these data are collected and classified to detect student attitudes and beliefs so that the tutor can respond appropriately (Arroyo et al., 2004). Take gaming behavior as an example. Using student data, Wayang Outpost can identify two gaming behaviors that are most prevalent and respond to them: (a) being unmotivated and exhausting hints so as to get a bottom-out hint or (b) being unmotivated and quickly guessing for a correct answer (Arroyo et al., 2006). Other student behavior patterns can be determined from the data as well such as when a student is guessing, solving problems independently and accurately or using multimedia help to learn (Beal & Cohen, 2008). Being able to classify student behaviors this way allows the tutor to chain behaviors together in order to predict future behaviors (Beal & Cohen, 2008).

Student affect is also determined from more than just a survey. Wayang Outpost used four sensors to determine affect (Arroyo, Cooper, et al., 2009): facial recognition software, a wireless bracelet to measure skin conductivity, a pressure mouse, and pressure sensitive seat cushions and back pad. These sensors have prediction accuracies of 78-87% and can be used in

tailoring tutoring tactics to student performance and emotional states (Cooper, Muldner, Arroyo, Woolf, & Burleson, 2010).

Interface

Wayang Outpost is set at a research station in the Borneo rainforest where students play a game to address environmental issues surrounding orangutans. Wayang Outpost is web-based and uses multimedia to direct attention, animate parts of the solution, emphasize concepts, and motivate students (Arroyo, Beal, et al., 2003).

A pedagogical agent has always been a part of the tutoring system. The agent started off as an Indonesian shadow puppet who presented the problems to students and provided step-by-step help (Arroyo, Beal, et al., 2003). The shadow puppet was eventually replaced by a learning companion named Jake or Jane and could be a boy or girl and of an ethnic background (Arroyo, Woolf, et al., 2009). These two learning companions were exactly the same in the help they offered but differed in the voice that was used. Jake and Jane were empathic agents that reflected the emotions of the students since Jake and Jane are solving the problems along with the students (Arroyo, Woolf, et al., 2009). They react only when a student has answered a problem correctly or incorrectly and will either provide metacognitive or affective feedback or direct the student to use the help button for further assistance from the system (Cooper et al., 2009). While a learning companion was present, students reported significantly less frustration and more interest for the task at hand (Arroyo, Woolf, et al., 2010).

Overall Wayang Outpost provides relevant information that explores the different cognitive and affective needs of boys and girls. For example, females reported having a better learning experience with a learning companion present whereas males reported a better experience without one present (Arroyo, Woolf, et al., 2010). Other behavioral differences were

noted such as females would game less and spend more time on problems when a learning companion was present (Arroyo, Woolf, et al., 2010).

While a virtual learning companion seems to have its uses, Wayang Outpost recognizes the importance of collaboration with other students in the classroom. As the authors of the system observed students using Wayang Outpost, they noticed that students would frequently work together to solve problems but that was made difficult when students were working on separate problems. To facilitate this collaboration, a "Go To' button was added to the interface so that students could work on the same problem at the same time (Arroyo, Woolf, & Shanabrook, 2012).

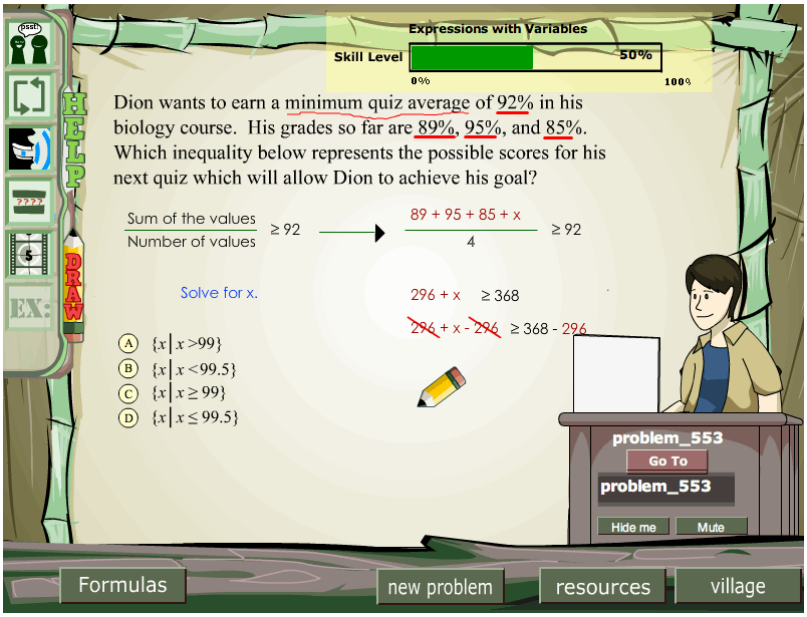

Figure 1. The user interface of Wayang Outpost (Arroyo et al., 2012).

Learning Gains

While the pre- and posttests that students take can act as measures of learning gains, not much was found that reported learning gains of the system. One study showed a 27% increase from pre- to posttest (Arroyo et al., 2006). Another study showed that scores improved by an average of 10% after three hours of instruction and 20% after four to five hours of instruction

with the tutor (Arroyo et al., 2012). One more study compared performance on state standardized exams where students who used Wayang Outpost scored a mean of 92% versus a control who averaged 76% (Wixon et al., 2014).

References

Arroyo, I., Beal, C., Bergman, A., Lindenmuth, M., Marshall, D., & Woolf, B. P. (2003). Intelligent Tutoring for high-stakes achievement tests. *Artificial Intelligence in Education: Shaping the Future of Learning Through Intelligent Technologies*, *97*, 365-368.

Arroyo, I., Beal, C., Murray, T., Walles, R., & Woolf, B. P. (2004). Web-based intelligent multimedia tutoring for high stakes achievement tests. In J. C. Lester, R. M. Vicari, & F. Paraguacu (Eds.), *Intelligent tutoring systems* (pp. 468-477). Berlin/Heidelberg: Springer.

Arroyo, I., Cooper, D. G., Burleson, W., Woolf, B. P., Muldner, K., & Christopherson, R. (2009). Emotion sensors go to school. In *Proceedings of the 2009 Conference on Artificial Intelligence in Education* (pp. 17-24). Amsterdam: IOS Press.

Arroyo, I., Mehranian, H., & Woolf, B. P. (2010). Effort-based tutoring: An empirical approach to intelligent tutoring. In R. Baker, A. Merceron, P. Pavlik (Eds.), *Educational Data Mining 2010* (pp. 1-10). Pittsburgh, PA: Carnegie Mellon University

Arroyo, I., Royer, J. M., & Woolf, B. P. (2011). Using an intelligent tutor and math fluency training to improve math performance. *International Journal of Artificial Intelligence in Education*, *21*(1-2), 135-152.

Arroyo, I., Walles, R., Beal, C. R., & Woolf, B. P. (2003). Tutoring for SAT-math with Wayang Outpost. In *Advanced Technologies for Mathematics Education Workshop* (pp. 40-46). Amherst, MA: University of Massachusetts.

Arroyo, I., & Woolf, B. P. (2005). Inferring learning and attitudes from a Bayesian network of log file data. In *Proceedings of the 2005 Conference on Artificial Intelligence in Education* (pp. 33-40). Amsterdam: IOS Press.

Arroyo, I., Woolf, B. P., & Beal, C. R. (2006). Addressing cognitive differences and gender during problem solving. *International Journal of Technology, Instruction, Cognition and Learning*, *4*, 31-63.

Arroyo, I., Woolf, B. P., Cooper, D. G., Burleson, W., & Muldner, K. (201). The impact of animated pedagogical agents on girls' and boys' emotions, attitudes, behaviors and learning. In I. Aedo et al., (Eds.), *Proceedings of Advanced Learning Technologies (ICALT), 2011 11th IEEE International Conference* (pp. 506-510), Athens, Georgia: IEEE.

Arroyo, I., Woolf, B. P., Royer, J. M., & Tai, M. (2009). Affective Gendered Learning Companions. *AIED*, *200,* 41-48.

Arroyo, I., Woolf, B. P., Royer, J. M., Tai, M., Muldner, K., Burleson, W., & Cooper, D. (2010). *Gender matters: The impact of animated agents on students' affect, behavior and learning.* Amherst: UMASS.

Arroyo, I., Woolf, B. P., & Shanabrook, D. (2012). Casual collaborations while learning mathematics with an intelligent tutoring system. In *Collaborative Learning Environments Workshop at ITS* (pp. 1-10). Crete, Greece: ISEE.

Beal, C. R., & Cohen, P. R. (2008). Temporal data mining for educational applications. In T. Ho & Z. Zhou (Eds.), *Pacific Rim International Conference on Artificial Intelligence* (pp. 66-77). Berlin/Heidelberg: Springer.

Cooper, D. G., Arroyo, I., Woolf, B. P., Muldner, K., Burleson, W., & Christopherson, R. (2009). Sensors model student self concept in the classroom. In G. J. Houben, G. McCalla, F. Pianesi, & M. Zancanaro (Eds.), *User Modeling, Adaptation, and Personalization* (pp. 30-41). Berlin/Heidelberg: Springer.

Cooper, D. G., Muldner, K., Arroyo, I., Woolf, B. P., & Burleson, W. (2010). Ranking feature sets for emotion models used in classroom based intelligent tutoring systems. In P. De Bra, A. Kobsa, & D. Chin (Eds.), *International Conference on User Modeling, Adaptation, and Personalization* (pp. 135-146). Berlin/Heidelberg: Springer.

Ferguson, K., Arroyo, I., Mahadevan, S., Woolf, B., & Barto, A. (2006). Improving intelligent tutoring systems: Using expectation maximization to learn student skill levels. In M. Ikeda, K. Ashley, & T. Chan (Eds.), *Intelligent Tutoring Systems* (pp. 453-462). Berlin/Heidelberg: Springer.

Johns, J., & Woolf, B. (2006). A dynamic mixture model to detect student motivation and proficiency. In *Proceedings of the National Conference on Artificial Intelligence* (pp. 2-8). Menlo Park, CA: AAAI Press

Tai, M., Arroyo, I., & Woolf, B. P. (2013). Teammate relationships improve help-seeking behavior in an intelligent tutoring system. In H. Lane, J. Mostow, & P. Pavlik (Eds.), *International Conference on Artificial Intelligence in Education* (pp. 239-248). Berlin/Heidelberg: Springer.

Wixon, M., Arroyo, I., Muldner, K., Burleson, W., Rai, D., & Woolf, B. (2014). The opportunities and limitations of scaling up sensor-free affect detection. In J. Stamper, Z. Pardos, M. Mavrikis, & B. McLaren (Eds.), *Proceedings of the 7th International Conference on Educational Data Mining 2014* (pp. 145-152). London, UK: IEDMS.