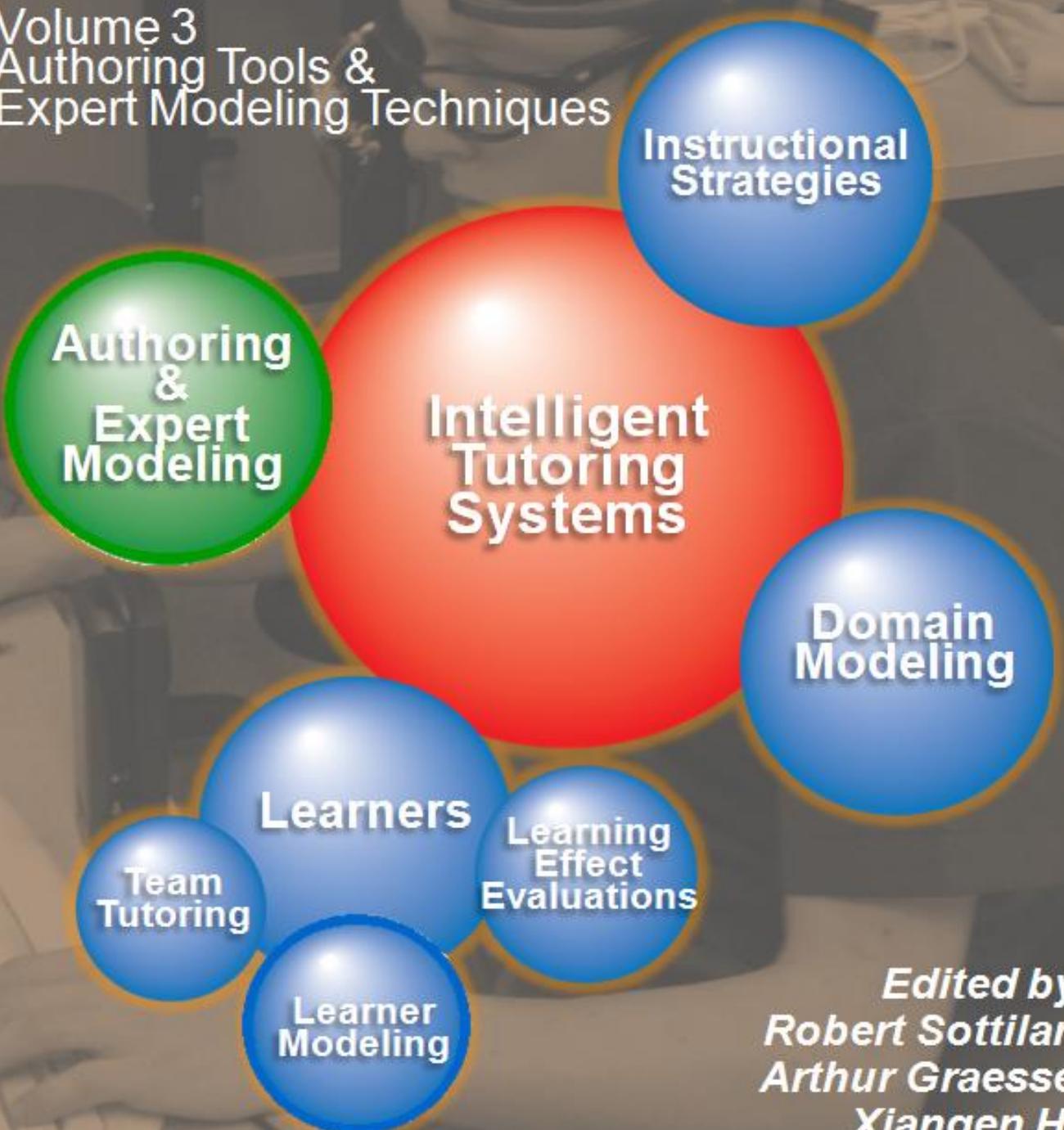


Design Recommendations for Intelligent Tutoring Systems

Volume 3
Authoring Tools &
Expert Modeling Techniques



Edited by:
Robert Sottilare
Arthur Graesser
Xiangen Hu
Keith Brawner

A Book in the Adaptive Tutoring Series

Design Recommendations for Intelligent Tutoring Systems

Volume 3
**Authoring Tools and Expert Modeling
Techniques**

Edited by:
Robert A. Sottilare
Arthur C. Graesser
Xiangen Hu
Keith Brawner

A Book in the Adaptive Tutoring Series

Copyright © 2015 by the U.S. Army Research Laboratory

**Copyright not claimed on material written by an employee of the U.S. Government.
All rights reserved.**

**No part of this book may be reproduced in any manner, print or electronic, without written
permission of the copyright holder.**

*The views expressed herein are those of the authors and do not necessarily reflect the views
of the U.S. Army Research Laboratory.*

Use of trade names or names of commercial sources is for information only and does not imply endorsement
by the U.S. Army Research Laboratory.

This publication is intended to provide accurate information regarding the subject matter addressed herein. The
information in this publication is subject to change at any time without notice. The U.S. Army Research
Laboratory, nor the authors of the publication, makes any guarantees or warranties concerning the information
contained herein.

Printed in the United States of America
First Printing, June 2015

*U.S. Army Research Laboratory
Human Research & Engineering Directorate
SFC Paul Ray Smith Simulation & Training Technology Center
Orlando, Florida*

International Standard Book Number: 978-0-9893923-7-2

We wish to acknowledge the editing and formatting contributions of Carol Johnson and Deeja Cruz, ARL

Dedicated to current and future scientists and developers of adaptive learning technologies

CONTENTS

Introduction	i
Section I: Perspectives of Authoring Tools and Methods	1
Chapter 1 Challenges to Enhancing Authoring Tools and Methods for Intelligent Tutoring Systems	3
Chapter 2 Theory-based Authoring Tool Design: Considering the Complexity of Tasks and Mental Models	9
Chapter 3 One-Size-Fits-Some: ITS Genres and What They (Should) Tell Us About Authoring Tools	31
Chapter 4 Generalizing the Genres for ITS: Authoring Considerations for Representative Learning Tasks	47
Section II: Authoring Model-Tracing Tutors	65
Chapter 5 A Historical Perspective on Authoring and ITS: Reviewing Some Lessons Learned	67
Chapter 6 Authoring Example-based Tutors for Procedural Tasks	71
Chapter 7 Supporting the WISE Design Process: Authoring Tools that Enable Insights into Technology-Enhanced Learning	95
Chapter 8 Authoring Tools for Ill-defined Domains in Intelligent Tutoring Systems: Flexibility and Stealth Assessment	109
Chapter 9 Design Considerations for Collaborative Authoring in Intelligent Tutoring Systems	123
Chapter 10 Authoring for the Product Lifecycle	137

Section III: Authoring Agent-Based Tutors **145**

Chapter 11	Authoring Agent-based Tutors	147
Chapter 12	Design Principles for Pedagogical Agent Authoring Tools	151
Chapter 13	Adaptive and Generative Agents for Training Content Development	161
Chapter 14	Authoring Conversation-based Assessment Scenarios	169
Chapter 15	Authoring Networked Learner Models in Complex Domains	179

Section IV: Authoring Dialogue-Based Tutors **193**

Chapter 16	Authoring Conversation-based Tutors	195
Chapter 17	ASAT: AutoTutor Script Authoring Tool	199
Chapter 18	Constructing Virtual Role-Play Simulations	211
Chapter 19	Emerging Trends in Automated Authoring	227
Chapter 20	Developing Conversational Multimedia Tutorial Dialogues	243

Section V: Increasing Interoperability and Reducing Workload and Skill Requirements for Authoring Tutors **255**

Chapter 21	Approaches to Reduce Workload and Skill Requirements in the Authoring of Intelligent Tutoring Systems	257
Chapter 22	Reflecting on Twelve Years of ITS Authoring Tools Research with CTAT	263
Chapter 23	Usability Considerations and Different User Roles in the Generalized Intelligent Framework for Tutoring	285
Chapter 24	Invisible Intelligent Authoring Tools	293
Chapter 25	Lowering the Technical Skill Requirements for Building Intelligent Tutors: A Review of Authoring Tools	303
Chapter 26	Authoring Instructional Management Logic in GIFT Using the Engine for Management of Adaptive Pedagogy (EMAP)	319

Chapter 27 Tiering, Layering and Bootstrapping for ITS Development	335
Chapter 28 Expanding Authoring Tools to Support Psychomotor Training Beyond the Desktop	347
Biographies	357
Index	375

INTRODUCTION

*Robert A. Sottilare¹, Arthur C. Graesser², Xiangen Hu²,
and Keith W. Brawner¹, Eds.*

*U.S. Army Research Laboratory - Human Research and Engineering Directorate¹
University of Memphis Institute for Intelligent Systems²*

This book is the third in a planned series of books that examine key topics (e.g., learner modeling, instructional strategies, authoring, domain modeling, impact on learning, and team tutoring) in intelligent tutoring system (ITS) design through the lens of the Generalized Intelligent Framework for Tutoring (GIFT) (Sotilare, Brawner, Goldberg & Holden, 2012; Sotilare, Brawner, Goldberg & Holden, 2013). GIFT is a modular, service-oriented architecture created to reduce the cost and skill required to author ITSs, manage instruction within ITSs, and evaluate the effect of ITS technologies on learning, performance, retention, and transfer.

The first two books in this series, Learner Modeling (ISBN 978-0-9893923-2-7) and Instructional Management (ISBN 978-0-9893923-0-3), are freely available at www.GIFTutoring.org and on Google Play.

This introduction begins with a description of tutoring functions, provides a glimpse of authoring best practices, and examines the motivation for standards in the design, authoring, instruction, and evaluation of ITS tools and methods. We introduce GIFT design principles discuss how readers might use this book as a design tool. We begin by examining the major components of ITSs.

Components and Functions of Intelligent Tutoring Systems

It is generally accepted that an ITS has four major components (Elson-Cook, 1993; Nkambou, Mizoguchi & Bourdeau, 2010; Graesser, Conley & Olney, 2012; Psotka & Mutter, 2008; Sleeman & Brown, 1982; VanLehn, 2006; Woolf, 2009): the domain model, the student model, the tutoring model, and the user-interface model. GIFT similarly adopts this four-part distinction, but with slightly different corresponding labels (domain module, learner module, pedagogical module, and tutor-user interface) and the addition of the sensor module, which can be viewed as an expansion of the user interface.

- (1) The **domain model** contains the set of skills, knowledge, and strategies/tactics of the topic being tutored. It normally contains the ideal expert knowledge and also the bugs, mal-rules, and misconceptions that students periodically exhibit.
- (2) The **learner model** consists of the cognitive, affective, motivational, and other psychological states that evolve during the course of learning. Since learner performance is primarily tracked in the domain model, the learner model is often viewed as an overlay (subset) of the domain model, which changes over the course of tutoring. For example, “knowledge tracing” tracks the learner’s progress from problem to problem and builds a profile of strengths and weaknesses relative to the domain model (Anderson, Corbett, Koedinger & Pelletier, 1995). An ITS may also consider psychological states outside of the domain model that need to be considered as parameters to guide tutoring.
- (3) The **tutor model** (also known as the pedagogical model or the instructional model) takes the domain and learner models as input and selects tutoring strategies, steps, and actions on what the tutor should do next in the exchange. In mixed-initiative systems, the learners may also take actions, ask questions, or request help (Aleven, McLaren, Roll & Koedinger, 2006; Rus & Graesser, 2009), but the ITS always needs to be ready to decide “what to do next” at any point and this is determined by a tutoring model that captures the researchers’ pedagogical theories.
- (4) The **user interface** interprets the learner’s contributions through various input media (speech, typing, clicking) and produces output in different media (text, diagrams, animations, agents). In addition to the conventional human-computer interface features, some recent systems have incorporated natural language interaction (Graesser et al., 2012; Johnson & Valente, 2008),

speech recognition (D'Mello, Graesser & King, 2010; Litman, 2013), and the sensing of learner emotions (Baker, D'Mello, Rodrigo & Graesser, 2010; D'Mello & Graesser, 2010; Goldberg, Sottilare, Brawner, Holden, 2011).

The designers of a tutor model must make decisions on each of the various major components in order to create an enhanced learning experience through well-grounded pedagogical strategies (optimal plans for action by the tutor) that are selected based on learner states and traits and that are delivered to the learner as instructional tactics (optimal actions by the tutor). Next, tactics are chosen based on the previously selected strategies and instructional context (the conditions of the training at the time of the instructional decision. This is part of the learning effect model (Sottilare, 2012; Fletcher & Sottilare, 2013; Sottilare, 2013; Sottilare, Ragusa, Hoffman & Goldberg, 2013), which has been updated and described below in more detail in section titled “Motivations for Intelligent Tutoring System Standards” in this introductory chapter.

Principles of Learning and Instructional Techniques, Strategies, and Tactics

Instructional techniques, strategies, and tactics play a central role in the design of GIFT. Instructional techniques represent instructional best practices and principles from the literature, many of which have yet to be implemented within GIFT at the writing of this volume. Examples of instructional techniques include, but are not limited to, error-sensitive feedback, mastery learning, adaptive spacing and repetition, and fading worked examples. Others are represented in the next section of this introduction. It is anticipated that techniques within GIFT will be implemented as software-based agents where the agent will monitor learner progress and instructional context to determine if best practices (agent policies) have been adhered to or violated. Over time, the agent will learn to enforce agent policies in a manner that optimizes learning and performance.

Some of the best instructional practices (techniques) have yet to be implemented in GIFT, but many instructional strategies and tactics have been implemented. Instructional strategies (plans for action by the tutor) are selected based on changes to the learner's state (cognitive, affective, physical). If a sufficient change in any learner's state occurs, this triggers GIFT to select a generic strategy (e.g., provide feedback). The instructional context along with the instructional strategy then triggers the specific selection of an instructional tactic (an action to be taken by the tutor). If the strategy is “provide feedback,” then the tactic might be to “provide feedback on the error committed during the presentation of instructional concept ‘B’ in the chat window during the next turn.” Tactics detail what is to be done, why, when, and how.

An adaptive, intelligent learning environment needs to select the right instructional strategies at the right time, based on its model of the learner in specific conditions and the learning process in general. Such selections should be taken to maximize deep learning and motivation while minimizing training time and costs. Authoring Tools was the theme of the third advisory board meeting of the collaboration between (1) the Human Research and Engineering Directorate (HRED) of the U.S. Army Research Laboratory (ARL) and (2) the Advanced Distributed Learning Center for Intelligent Tutoring Systems Research & Development (ADL CITSRD) in the Institute for Intelligent Systems (IIS) at the University of Memphis. The purpose of this volume is to provide a succinct illustration of some commonly used authoring tools and associated principles of authoring tool design.

The following are examples of successful authoring tools:

- The Authoring Software Platform for Intelligent Resources in Education (ASPIRE) (Mitrovic, et al., 2009), created by the Intelligent Computer Tutoring Group at the University of Canterbury in

New Zealand, employs domain experts to create constraint-based tutors through the generation of domain model supplemental information from interactions with the system. Such information is then processed by an expert user who has familiarity with the constraint language.

- The AutoTutor Authoring Tools were created by the University of Memphis IIS. These tools allow a user to configure AutoTutor conversational scripts via a desktop or web-based interface, and have made recent efforts to simplify the authoring process to a level which the student can have input. The AutoTutor Script Authoring Tool (ASAT) is compatible with the GIFT authoring suite and can be shared as sharable knowledge objects (SKOs) (Nye, Hu, Graesser, and Zhiqiang, 2014)).
- The Cognitive Tutor Authoring Tools (CTAT), developed by Carnegie Mellon University, are one of the longest running and most successful toolsets. CTAT allows authors to link tutoring knowledge to a graphical user interface (GUI) with little programming effort and demonstrate model solutions rapidly. Recently efforts have taken steps to automate authoring through a process of demonstration by an expert with a project called SimStudent (Matsuda, Cohen, and Koedinger, 2015), resulting in an expert model.
- The GIFT Authoring Tools, created by ARL and increasingly by the GIFT user community, are open source. GIFT was created to realize the US Army Learning Model (ALM) self-regulated learning capability and to reduce the time/cost/skill needed to author ITSs. Currently, the GIFT authoring tools consist of a series of developer-oriented, XML-based editing tools (e.g., Course Authoring Tool (CAT), Survey Authoring System, Domain Knowledge File Authoring Tool (DAT), and Pedagogy Configuration Authoring Tool (PCAT)), which are being integrated with a single simplified web-based authoring tool known as the GIFT Authoring Tool (GAT). These tools have been used to create a variety of tutors in a variety of domains of instruction (e.g., casualty care, cryptography, solving logic puzzles, and construction equipment use). The design goal for the GAT is to provide ITS authoring capabilities, which can be used by domain experts with little or no knowledge or skill in either computer programming or instructional system design to produce highly effective and efficient ITSs (Sotilare, 2013).
- The Situated Pedagogical (SitPed) authoring tool, created by the University of Southern California, focuses heavily on preview-based authoring, where a non-technical author can simulate the experience of a student while simultaneously demonstrating actions and statements to the tutor. This model blends the authoring components of an expert model, pedagogical action, and virtual human creation in order to gain efficiency.

There are a number of barriers to making authoring tools usable by the general public. The main barriers are:

- Specialized skills (e.g., computer programming, understanding of instructional design) are required to master existing authoring tools.
- Time and cost to author ITSs using existing authoring tools is high due to the complexity of ITSs and deficiencies in the usability of current authoring tools.
- Time required to retrieve and organize authoring content is high.
- Standards for ITS authoring are non-existent, yielding extremely low interoperability between authoring toolsets.

Members of the third advisory board were selected because their research fills many of these gaps and provides more sophisticated authoring strategies for GIFT. More specifically, researchers on the board have made major advances for model-tracing, agent-based, and/or dialogue-based ITSs in three thematic subcategories: (1) simplified user interfaces, (2) methods for curation of data (retrieval, storage, and organization), and (3) development of authoring job aids. Research in these subcategories is destined to move the horizon of authoring tools from the laboratory to the classroom through the creation of easy to use systems built on standardized design principles. Our goal was to elicit input from members of this advisory board and the authors of this book to shape ITSs authoring standards.

Motivations for Intelligent Tutoring System Standards

An emphasis on self-regulated learning has highlighted a requirement for point-of-need training in environments where human tutors are either unavailable or impractical. ITSs have been shown to be as effective as expert human tutors (VanLehn, 2011) in one-to-one tutoring in well-defined domains (e.g., mathematics or physics) and significantly better than traditional classroom training environments. ITSs have demonstrated significant promise, but 50 years of research have been unsuccessful in making ITSs ubiquitous in military training or the tool of choice in our educational system. This begs the question: “Why?”

Part of the answer lies in the fact that the availability and use of ITSs have been constrained by their high development costs, their limited reuse, a lack of standards, and their inadequate adaptability to the needs of learners. Educational and training technologies like ITSs are primarily researched and developed in a few key environments: industry, academia, and government including military domains. Each of these environments has its own challenges and design constraints. The application of ITSs to military domains is further hampered by the complex and often ill-defined environments in which the US military operates today. ITSs are often built as domain-specific, unique, one-of-a-kind, largely domain-dependent solutions focused on a single pedagogical strategy (e.g., model tracing or constraint-based approaches) when complex learning domains may require novel or hybrid approaches. Therefore, a modular ITS framework and standards are needed to enhance reuse, support authoring, optimize instructional strategies, and lower the cost and skillset needed for users to adopt ITS solutions for training and education. It was out of this need that the idea for GIFT arose.

GIFT has three primary functions: authoring, instructional management, and evaluation. First, it is a framework for authoring new ITS components, methods, strategies, and whole tutoring systems. Second, GIFT is an instructional manager that integrates selected instructional theory, principles, and strategies for use in ITSs. Finally, GIFT is an experimental testbed used to evaluate the effectiveness and impact of ITS components, tools, and methods. GIFT is based on a learner-centric approach with the goal of improving linkages in the updated adaptive tutoring learning effect model (Figure 1; Sottilare, 2012; Fletcher & Sottilare, 2013; Sottilare, 2013; Sottilare, Ragusa, Hoffman & Goldberg, 2013).

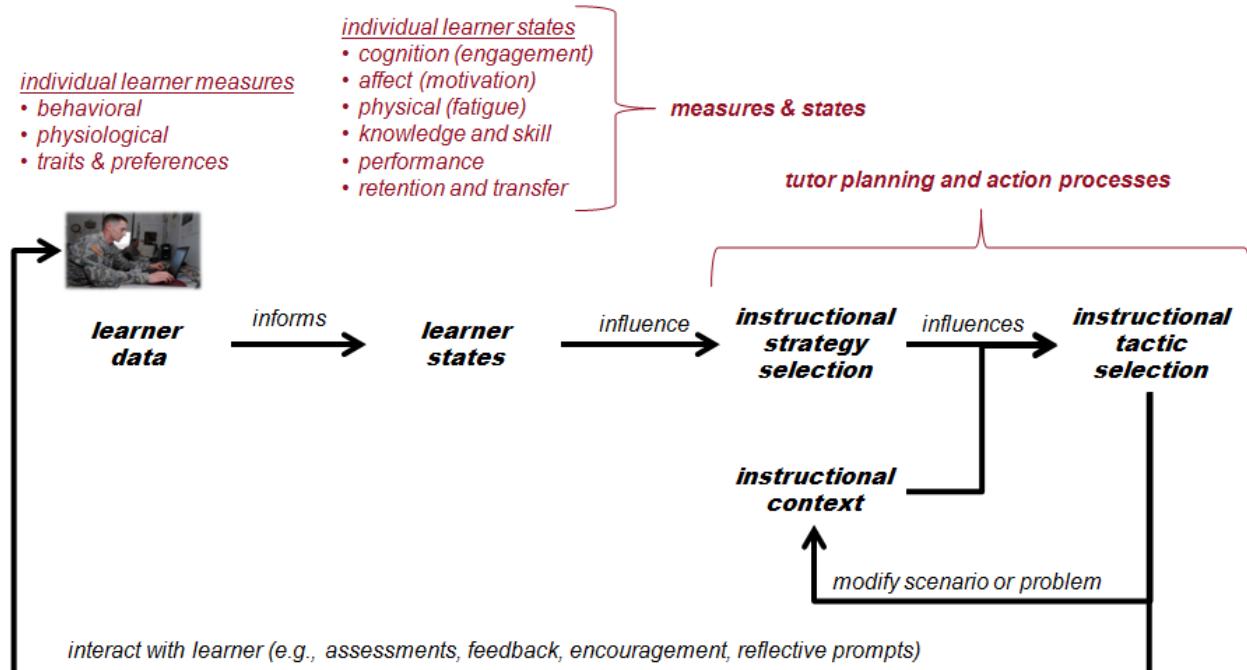


Figure 1. Updated adaptive tutoring learning effect model

A deeper understanding of the learner's behaviors, traits, and preferences (learner data) collected through performance, physiological and behavioral sensors, and surveys will allow for more accurate evaluation of the learner's states (e.g., engagement level, confusion, frustration). This will result in a better and more persistent model of the learner. To enhance the adaptability of the ITS, methods are needed to accurately classify learner states (e.g., cognitive, affective, psychomotor, social) and select optimal instructional strategies given the learner's existing states. A more comprehensive learner model will allow the ITS to adapt more appropriately to address the learner's needs by changing the instructional strategy (e.g., content, flow, or feedback). An instructional strategy better aligned to the learner's needs is more likely to positively influence their learning gains. It is with the goal of optimized learning gains in mind that the design principles for GIFT were formulated.

This version of the learning effect model has been updated to gain understanding of the effect of optimal instructional tactics and instructional context (both part of the domain model) on specific desired outcomes including knowledge and skill acquisition, performance, retention, and transfer of skills from training or tutoring environments to operational contexts (e.g., from practice to application). The feedback loops in Figure 1 have been added to identify tactics as either a change in instructional context or interaction with the learner. This allows the ITS to adapt to the need of the learner. Consequently, the ITS changes over time by reinforcing learning mechanisms.

GIFT Design Principles

The GIFT methodology for developing a modular, computer-based tutoring framework for training and education considered major design goals, anticipated uses, and applications. The design process also considered enhancing one-to-one (individual) and one-to-many (collective or team) tutoring experiences beyond the state of practice for ITSs today. A significant focus of the GIFT design was on domain-dependent elements in the domain module only. This is a design tradeoff to foster reuse and allows ITS decisions and actions to be made across any/all domains of instruction.

One design principle adopted in GIFT is that each module should be capable of gathering information from other modules according to the design specification. Designing to this principle resulted in standard message sets and message transmission rules (i.e., request-driven, event-driven, or periodic transmissions). For instance, the pedagogical module is capable of receiving information from the learner module to develop courses of action for future instructional content to be displayed, manage flow and challenge level, and select appropriate feedback. Changes to the learner's state (e.g., engagement, motivation, or affect) trigger messages to the pedagogical module, which then recommends general courses of action (e.g., ask a question or prompt the learner for more information) to the domain module, which provides a domain-specific intervention (e.g., what is the next step?).

Another design principle adopted within GIFT is the separation of content from the executable code (Patil & Abraham, 2010). Data and data structures are placed within models and libraries, while software processes are programmed into interoperable modules. Efficiency and effectiveness goals (e.g., accelerated learning and enhanced retention) were considered to address the time available for military training and the renewed emphasis on self-regulated learning. An outgrowth of this emphasis on efficiency and effectiveness led Dr. Sottilare to seek external collaboration and guidance. In 2012, ARL with the University of Memphis developed advisory boards of senior tutoring system scientists from academia and government to influence the GIFT design goals moving forward. Advisory boards have been held each year since 2012 resulting in volumes in the Design Recommendations for Intelligent Tutoring Systems series the following year. The learner modeling advisory board was completed in September 2012 and Volume 1 followed in July 2013. An advisory board on instructional management was completed in July 2013 and Volume 2 followed in June 2014. The authoring tools advisory board was completed in June of 2014 and Volume 3 is planned for publication in May or June 2015. Future boards are planned for domain modeling, learner assessment, team training, and learning effect evaluations.

Design Goals and Anticipated Uses

GIFT may be used for a number of purposes, with the primary ones enumerated below:

1. An architectural framework with modular, interchangeable elements and defined relationships to support stand-alone tutoring or guided training if integrated with a training system
2. A set of specifications to guide ITS development
3. A set of exemplars or use cases for GIFT to support authoring, reuse, and ease-of-use
4. A technical platform or testbed for guiding the evaluation, development/refinement of concrete systems

These use cases have been distilled down into the three primary functional areas, or *constructs*: authoring, instructional management, and the recently renamed evaluation construct. Discussed below are the purposes, associated design goals, and anticipated uses for each of the GIFT constructs.

GIFT Authoring Construct

The purpose of the GIFT authoring construct is to provide technology (tools and methods) to make it affordable and easier to build ITSs and ITS components. Toward this end, a set of XML configuration tools continues to be developed to allow for data-driven changes to the design and implementation of

GIFT-generated ITSs. The design goals for the GIFT authoring construct have been adapted from Murray (1999, 2003) and Sottilare and Gilbert (2011). The GIFT authoring design goals are as follow:

- Decrease the effort (time, cost, and/or other resources) for authoring and analyzing ITSs by automating authoring processes, developing authoring tools and methods, and developing standards to promote reuse.
- Decrease the skill threshold by tailoring tools for specific disciplines (e.g., instructional designers, training developers, and trainers) to author, analyze, and employ ITS technologies.
- Provide tools to aid designers/authors/trainers/researchers in organizing their knowledge.
- Support (structure, recommend, or enforce) good design principles in pedagogy through user interfaces and other interactions.
- Enable rapid prototyping of ITSs to allow for rapid design/evaluation cycles of prototype capabilities.
- Employ standards to support rapid integration of external training/tutoring environments (e.g., simulators, serious games, slide presentations, transmedia narratives, and other interactive multimedia).
- Develop/exploit common tools and user interfaces to adapt ITS design through data-driven means.
- Promote reuse through domain-independent modules and data structures.
- Leverage open-source solutions to reduce ITS development and sustainment costs.
- Develop interfaces/gateways to widely-used commercial and academic tools (e.g., games, sensors, toolkits, virtual humans).

As a user-centric architecture, anticipated uses for GIFT authoring tools are driven largely by the anticipated users, which include learners, domain experts, instructional system designers, training and tutoring system developers, trainers and teachers, and researchers. In addition to user models and GUIs, GIFT authoring tools include domain-specific knowledge configuration tools, instructional strategy development tools, and a compiler to generate executable ITSs from GIFT components in a variety of formats (e.g., PC, Android, and IPad).

Within GIFT, domain-specific knowledge configuration tools permit authoring of new knowledge elements or reusing existing (stored) knowledge elements. Domain knowledge elements include learning objectives, media, task descriptions, task conditions, standards and measures of success, common misconceptions, feedback library, and a question library, which are informed by instructional system design principles that, in turn, inform concept maps for lessons and whole courses. The task descriptions, task conditions, standards and measures of success, and common misconceptions may be informed by an expert or ideal learner model derived through a task analysis of the behaviors of a highly skilled user. ARL is investigating techniques to automate this expert model development process to reduce the time and cost of developing ITSs. In addition to feedback and questions, supplementary tools are anticipated to author explanations, summaries, examples, analogies, hints, and prompts in support of GIFT's instructional management construct.

GIFT Instructional Management Construct

The purpose of the GIFT instructional management construct is to integrate pedagogical best practices in GIFT-generated ITSs. The modularity of GIFT will also allow GIFT users to extract pedagogical models for use in tutoring/training systems that are not GIFT-generated. GIFT users may also integrate pedagogical models, instructional strategies, or instructional tactics from other tutoring systems into GIFT. The design goals for the GIFT instructional management construct are the following:

- Support ITS instruction for individuals and small teams in local and geographically distributed training environments (e.g., mobile training), and in both well-defined and ill-defined learning domains.
- Provide for comprehensive learner models that incorporate learner states, traits, demographics, and historical data (e.g., performance) to inform ITS decisions to adapt training/tutoring.
- Support low-cost, unobtrusive (passive) methods to sense learner behaviors and physiological measures and use these data along with instructional context to inform models to classify (in near real time) the learner's states (e.g., cognitive and affective).
- Support both macro-adaptive strategies (adaptation based on pre-training learner traits) and micro-adaptive instructional strategies and tactics (adaptation based learner states and state changes during training).
- Support the consideration of individual differences where they have empirically been documented to be significant influencers of learning outcomes (e.g., knowledge or skill acquisition, retention, and performance).
- Support adaptation (e.g., pace, flow, and challenge level) of the instruction based the domain and learning class (e.g., cognitive learning, affective learning, psychomotor learning, social learning).
- Model appropriate instructional strategies and tactics of expert human tutors to develop a comprehensive pedagogical model.

To support the development of optimized instructional strategies and tactics, GIFT is heavily grounded in learning theory, tutoring theory, and motivational theory. Learning theory applied in GIFT includes conditions of learning and theory of instruction (Gagné, 1985), component display theory (Merrill, Reiser, Ranney & Trafton, 1992), cognitive learning (Anderson & Krathwohl, 2001), affective learning (Krathwohl, Bloom & Masia, 1964; Goleman, 1995), psychomotor learning (Simpson, 1972), and social learning (Sottilare, Holden, Brawner, and Goldberg, 2011; Soller, 2001). Aligning with our goal to model expert human tutors, GIFT considers the intelligent, nurturant, Socratic, progressive, indirect, reflective, and encouraging (INSPIRE) model of tutoring success (Lepper, Drake, and O'Donnell-Johnson, 1997) and the tutoring process defined by Person, Kreuz, Zwaan, and Graesser (1995) in the development of GIFT instructional strategies and tactics.

Human tutoring strategies have been documented by observing tutors with varying levels of expertise. For example, Lepper's INSPIRE model is an acronym that highlights the seven critical characteristics of successful tutors: Graesser and Person's (1994) 5-step tutoring frame is a common pattern of the tutor-learner interchange in which the tutor asks a question, the learner answers the question, the tutor gives short feedback on the answer, then the tutor and learner collaboratively improve the quality of (or embellish) the answer, and finally, the tutor evaluates whether the learner understands the answer. Cade,

Copeland, Person, and D'Mello (2008) identified a number of tutoring modes used by expert tutors, which hopefully could be integrated with ITS.

As a learner-centric architecture, anticipated uses for GIFT instructional management capabilities include both automated instruction and blended instruction, where human tutors/teachers/trainers use GIFT to support their curriculum objectives. If its design goals are realized, it is anticipated that GIFT will be widely used beyond military training contexts as GIFT users expand the number and type of learning domains and resulting ITS generated using GIFT.

GIFT Evaluation Construct

The GIFT Analysis Construct has recently migrated to become the GIFT Evaluation Construct with an emphasis on the evaluation of effect on learning, performance, retention and transfer. The purpose of the GIFT evaluation construct is to allow ITS researchers to experimentally assess and evaluate ITS technologies (ITS components, tools, and methods). The design goals for the GIFT evaluation construct are the following:

- Support the conduct of formative assessments to improve learning.
- Support summative evaluations to gauge the effect of technologies on learning.
- Support assessment of ITS processes to understand how learning is progressing throughout the tutoring process.
- Support evaluation of resulting learning versus stated learning objectives.
- Provide diagnostics to identify areas for improvement within ITS processes.
- Support the ability to comparatively evaluate ITS technologies against traditional tutoring or classroom teaching methods.
- Develop a testbed methodology to support assessments and evaluations (Figure 2).

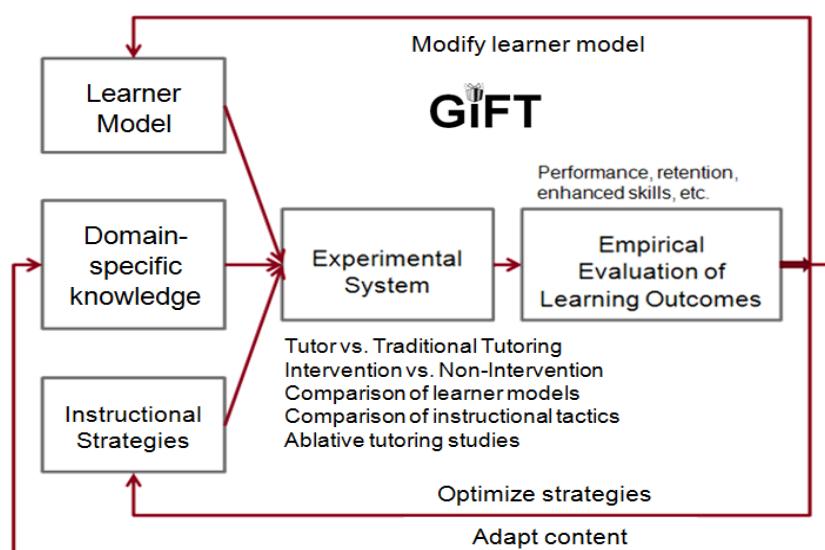


Figure 2. GIFT evaluation testbed methodology

Figure 2 illustrates an analysis testbed methodology being implemented in GIFT. This methodology was derived from Hanks, Pollack, and Cohen (1993). It supports manipulation of the learner model, instructional strategies, and domain-specific knowledge within GIFT, and may be used to evaluate variable in the adaptive tutoring learning effect model (Sotilare, 2012; Sotilare, Ragusa, Hoffman, and Goldberg, 2013). In developing their testbed methodology, Hanks et al. reviewed four testbed implementations (Tileworld, the Michigan Intelligent Coordination Experiment [MICE], the Phoenix testbed, and Truckworld) for evaluating the performance of artificially intelligent agents. Although agents have changed substantially in complexity during the past 20–25 years, the methods to evaluate their performance have remained markedly similar.

The authors designed the GIFT analysis testbed based upon Cohen’s assertion (Hanks et al., 1993) that testbeds have three critical roles related to the three phases of research. During the exploratory phase, agent behaviors need to be observed and classified in broad categories. This can be performed in an experimental environment. During the confirmatory phase, the testbed is needed to allow more strict characterizations of agent behavior to test specific hypotheses and compare methodologies. Finally, in order to generalize results, measurement and replication of conditions must be possible. Similarly, the GIFT analysis methodology (Figure 2) enables the comparison/contrast of ITS elements and assessment of their effect on learning outcomes (e.g., knowledge acquisition, skill acquisition, and retention).

How to Use This Book

This book is organized into five sections:

- I. Perspectives of Authoring Tools and Methods
- II. Authoring Model-Tracing Tutors
- III. Authoring Agent-Based Tutors
- IV. Authoring Dialogue-Based Tutors
- V. Increasing Interoperability and Reducing Workload and Skill Requirements for Authoring Tutors

Section I, *Perspective of Authoring Tools and Methods*, describes a variety of approaches to authoring ITSs and discusses their capabilities, limitations, and potential impact on learning. Section II, *Authoring Model-Tracing Tutors*, examines authoring tools for model-tracing tutors (sometimes referred to as example-tracing tutors), which are based on a problem representation stored in a behavior graph with problem-solving steps and specific methods handling alternative student behaviors. Emerging model-tracing tutoring authoring technologies are discussed with respect to how GIFT should be enhanced to make authoring of model-tracing tutors easier and more efficient. Section III, *Authoring Agent-Based Tutors*, discusses authoring processes guided by intelligent software agents. Section IV, *Authoring Dialogue-Based Tutors*, focuses primarily on interactive conversational tutors where virtual humans guide instruction. Finally, in Section V, we address the need for tools and methods to increase interoperability between authoring toolsets, and also reduce the knowledge and skill needed to author ITSs. A goal for GIFT is to reduce the skill and time needed to author ITSs to a point where domain experts can author ITSs without computer programming and instructional design knowledge/skills.

Chapter authors in each section were carefully selected for participation in this project based on their expertise in the field as ITS scientists, developers, and practitioners. *Design Recommendations for*

Intelligent Tutoring Systems: Volume 3 Authoring Tools is intended to be a design resource as well as community research resource. Volume 3 can also be of significant benefit as an educational guide for developing ITS scientists, as a roadmap for ITS research opportunities.

References

- Aleven, V., McLaren, B., Roll, I. & Koedinger, K. (2006). Toward meta-cognitive tutoring: A model of help seeking with a cognitive tutor. *International Journal of Artificial Intelligence in Education*, 16, 101-128.
- Anderson, J. R., Corbett, A. T., Koedinger, K. R. & Pelletier, R. (1995). Cognitive tutors: Lessons learned. *Journal of the Learning Sciences*, 4, 167-207.
- Anderson, L. W. & Krathwohl, D. R. (Eds.). (2001). *A taxonomy for learning, teaching and assessing: A revision of Bloom's Taxonomy of Educational Objectives: Complete edition*. New York : Longman.
- Baker, R.S., D'Mello, S.K., Rodrigo, M.T. & Graesser, A.C. (2010). Better to be frustrated than bored: The incidence, persistence, and impact of learners' cognitive-affective states during interactions with three different computer-based learning environments. *International Journal of Human-Computer Studies*, 68, 223-241.
- Cade, W., Copeland, J. Person, N., and D'Mello, S. K. (2008). Dialogue modes in expert tutoring. In B. Woolf, E. Aimeur, R. Nkambou & S. Lajoie (Eds.), Proceedings of the Ninth International Conference on Intelligent Tutoring Systems (pp. 470-479). Berlin, Heidelberg: Springer-Verlag.
- D'Mello, S. & Graesser, A.C. (2010). Multimodal semi-automated affect detection from conversational cues, gross body language, and facial features. *User Modeling and User-adapted Interaction*, 20, 147-187.
- D'Mello, S. K., Graesser, A. C. & King, B. (2010). Toward spoken human-computer tutorial dialogues. *Human Computer Interaction*, 25, 289-323.
- Elson-Cook, M. (1993). Student modeling in intelligent tutoring systems. *Artificial Intelligence Review*, 7, 227-240.
- Fletcher, J.D. and Sottilare, R. (2013). Shared Mental Models and Intelligent Tutoring for Teams. In R. Sottilare, A. Graesser, X. Hu, and H. Holden (Eds.) *Design Recommendations for Intelligent Tutoring Systems: Volume I - Learner Modeling*. Army Research Laboratory, Orlando, Florida. ISBN 978-0-9893923-0-3.
- Gagné, R. M. (1985). *The conditions of learning and theory of instruction* (4th ed.). New York: Holt, Rinehart & Winston.
- Goldberg, B.S., Sottilare, R.A., Brawner, K.W. & Holden, H.K. (2011). Predicting Learner Engagement during Well-Defined and Ill-Defined Computer-Based Intercultural Interactions. In S. D'Mello, A. Graesser, , B. Schuller & J.-C. Martin (Eds.), *Proceedings of the 4th International Conference on Affective Computing and Intelligent Interaction (ACII 2011) (Part 1: LNCS 6974)* (pp. 538-547). Berlin Heidelberg: Springer.
- Graesser, A.C., Conley, M. & Olney, A. (2012). Intelligent tutoring systems. In K.R. Harris, S. Graham & T. Urdan (Eds.), *APA Educational Psychology Handbook: Vol. 3. Applications to Learning and Teaching* (pp. 451-473). Washington, DC: American Psychological Association.
- Graesser, A. C., D'Mello, S. K., Hu, X., Cai, Z., Olney, A. & Morgan, B. (2012). AutoTutor. In P. McCarthy & C. Boonthum-Denecke (Eds.), *Applied natural language processing: Identification, investigation, and resolution* (pp. 169-187). Hershey, PA: IGI Global.
- Graesser, A. C. & Person, N. K. (1994). Question asking during tutoring. *American Educational Research Journal*, 31, 104–137.
- Hanks, S., Pollack, M.E. & Cohen, P.R. (1993). Benchmarks, test beds, controlled experimentation, and the design of agent architectures. *AI Magazine*, 14 (4), 17-42.
- Johnson, L. W. & Valente, A. (2008). Tactical language and culture training systems: Using artificial intelligence to teach foreign languages and cultures. In M. Goker & K. Haigh (Eds.), *Proceedings of the Twentieth Conference on Innovative Applications of Artificial Intelligence* (pp. 1632-1639). Menlo Park, CA: AAAI Press.
- Krathwohl, D.R., Bloom, B.S. & Masia, B.B. (1964). *Taxonomy of Educational Objectives: Handbook II: Affective Domain*. New York: David McKay Co.
- Lepper, M. R., Drake, M. & O'Donnell-Johnson, T. M. (1997). Scaffolding techniques of expert human tutors. In K. Hogan & M. Pressley (Eds.), *Scaffolding learner learning: Instructional approaches and issues* (pp. 108-144). New York: Brookline Books.
- Litman, D. (2013). Speech and language processing for adaptive training. In P. Durlach & A. Lesgold (Eds.), *Adaptive technologies for training and education*. Cambridge, MA: Cambridge University Press.

- Matsuda, N., Cohen, W. W. & Koedinger, K. R. (2015). Teaching the teacher: tutoring SimStudent leads to more effective cognitive tutor authoring. *International Journal of Artificial Intelligence in Education*, 25(1), 1-34.
- Murray, T. (1999). Authoring intelligent tutoring systems: An analysis of the state of the art. *International Journal of Artificial Intelligence in Education*, 10(1), 98–129.
- Murray, T. (2003). An Overview of Intelligent Tutoring System Authoring Tools: Updated analysis of the state of the art. In Murray, T.; Blessing, S.; Ainsworth, S. (Eds.), *Authoring tools for advanced technology learning environments* (pp. 491-545). Berlin: Springer.
- Merrill, D., Reiser, B., Ranney, M., and Trafton, J. (1992). Effective Tutoring Techniques: A Comparison of Human Tutors and Intelligent Tutoring Systems. *The Journal of the Learning Sciences*, 2(3), 277-305
- Mitrovic, A., Martin, B., Suraweera, P., Zakharov, K., Milik, N., Holland, J. & McGuigan, N. (2009). ASPIRE: an authoring system and deployment environment for constraint-based tutors. *International Journal of Artificial Intelligence in Education*, 19(2), 155-188.
- Nkambou, R., Mizoguchi, R. & Bourdeau, J. (2010). *Advances in intelligent tutoring systems*. Heidelberg: Springer.
- Nye, B., Hu, X., Graesser, A. & Cai, Z. (2014). Autotutor In The Cloud: A Service-Oriented Paradigm For An Interoperable Natural-Language Its. *Journal of Advanced Distributed Learning Technology*, 2(6), pp 49-63.
- Patil, A. S. & Abraham, A. (2010). Intelligent and Interactive Web-Based Tutoring System in Engineering Education: Reviews, Perspectives and Development. In F. Xhafa, S. Caballe, A. Abraham, T. Daradoumis & A. Juan Perez (Eds.), *Computational Intelligence for Technology Enhanced Learning. Studies in Computational Intelligence* (Vol 273, pp. 79-97). Berlin: Springer-Verlag.
- Person, N. K., Kreuz, R. J., Zwaan, R. A. & Graesser, A. C. (1995). Pragmatics and pedagogy: Conversational rules and politeness strategies may inhibit effective tutoring. *Cognition and Instruction*, 13(2), 161–188.
- Picard, R. (2006). Building an Affective Learning Companion. Keynote address at the 8th International Conference on Intelligent Tutoring Systems, Jhongli, Taiwan. Retrieved from http://www.its2006.org/ITS_keynote/ITS2006_01.pdf
- Psotka, J. & Mutter, S.A. (1988). *Intelligent Tutoring Systems: Lessons Learned*. Hillsdale, NJ: Lawrence Erlbaum Associates.
- Rus, V. & Graesser, A.C. (Eds.) (2009). The Question Generation Shared Task and Evaluation Challenge. Retrieved from <http://www.questiongeneration.org/>.
- Simpson, E. (1972). The classification of educational objectives in the psychomotor domain: *The psychomotor domain*. Vol. 3. Washington, DC: Gryphon House.
- Sleeman D. & J. S. Brown (Eds.) (1982). *Intelligent Tutoring Systems*. Orlando, Florida: Academic Press, Inc.
- Soller, A. (2001). Supporting social interaction in an intelligent collaborative learning system. *International Journal of Artificial Intelligence in Education*, 12(1), 40-62.
- Sottilare, R. & Gilbert, S. (2011). Considerations for tutoring, cognitive modeling, authoring and interaction design in serious games. *Authoring Simulation and Game-based Intelligent Tutoring workshop at the Artificial Intelligence in Education Conference (AIED) 2011*, Auckland, New Zealand, June 2011.
- Sottilare, R., Holden, H., Brawner, K. & Goldberg, B. (2011). Challenges and Emerging Concepts in the Development of Adaptive, Computer-based Tutoring Systems for Team Training. *Interservice/Industry Training Systems & Education Conference*, Orlando, Florida, December 2011.
- Sottilare, R.A., Brawner, K.W., Goldberg, B.S. & Holden, H.K. (2012). *The Generalized Intelligent Framework for Tutoring (GIFT)*. Orlando, FL: U.S. Army Research Laboratory Human Research & Engineering Directorate (ARL-HRED).
- Sottilare, R. (2012). Considerations in the development of an ontology for a Generalized Intelligent Framework for Tutoring. *International Defense & Homeland Security Simulation Workshop* in Proceedings of the I3M Conference. Vienna, Austria, September 2012.
- Sottilare, R., Ragusa, C., Hoffman, M. & Goldberg, B. (2013). Characterizing an adaptive tutoring learning effect chain for individual and team tutoring. In Proceedings of the *Interservice/Industry Training Simulation & Education Conference*, Orlando, Florida, December 2013.
- Sottilare, R. (2013). Special Report: Adaptive Intelligent Tutoring System (ITS) Research in Support of the Army Learning Model - Research Outline. *Army Research Laboratory (ARL-SR-0284)*, December 2013.
- VanLehn, K. (2006) The behavior of tutoring systems. *International Journal of Artificial Intelligence in Education*. 16(3), 227-265.
- VanLehn, K. (2011). The relative effectiveness of human tutoring, intelligent tutoring systems and other tutoring systems. *Educational Psychologist*, 46(4), 197-221.
- Woolf, B.P. (2009). *Building intelligent interactive tutors*. Burlington, MA: Morgan Kaufmann Publishers.

SECTION I

PERSPECTIVES OF AUTHORING TOOLS AND METHODS

R. Sottilare, Ed.

CHAPTER 1 Challenges to Enhancing Authoring Tools and Methods for Intelligent Tutoring Systems

Robert A. Sottilare
US Army Research Laboratory

Introduction

This chapter highlights a vision for intelligent tutoring system (ITS) authoring capabilities with respect to the major challenges or barriers to their adoption. A variety of authoring tools for ITSs have emerged, flourished, and gone extinct over the last 25 years. A few authoring toolsets, which have been introduced in Chapter 1 of this book, continue to evolve. Outside the growing number of commercial tools, two sets of authoring tools have found an active user community to sustain them. Carnegie Mellon University's Cognitive Tutor Authoring Tools (CTAT; Koedinger, Aleven & Heffernan, 2003) and the AutoTutor Authoring Tools (University of Memphis; Graesser et al., 1999) have a long history and remain viable today. Others like the Authoring Software Platform for Intelligent Resources in Education (ASPIRE; Mitrovic et al., 2009) are a bit more recent and still other authoring tools like the Generalized Intelligent Framework for Tutoring (GIFT; Sottilare, Brawner, Goldberg & Holden, 2012) and the Situated Pedagogical Authoring (SPA; University of Southern California, 2013) tools are newer still. Each of these tools has different scope (e.g., authoring for model-tracing, agent-based, or dialogue-based tutors) and a different set of learning theories (e.g., component display theory) that drive their design. A short description of each follows for comparison.

CTAT now has a set of authoring tools for both cognitive and example-tracing tutors. The CTAT authoring process requires definition of a task domain along with appropriate problems. CTAT was developed to support problem-based task domains. It may be more difficult to support the authoring of scenario-based tutors where problem-solving processes are less linear and multiple paths to success are the norm. In order to develop a domain model, a cognitive task analysis is required to understand how students learn the required concepts and evolve their skills. CTAT requires familiarity with the Java Expert System Shell (JESS) production rule language. The authoring tools for example-tracing tutors do not require any programming. CTAT is currently available as binary (executable) code.

The *AutoTutor Authoring Tools* are used to develop interactive tutors where students are taught through natural language discourse. AutoTutor was developed to support specific domains (e.g., Newtonian physics and computer literacy). As the name suggests, the AutoTutor Script Authoring Tool (ASAT) is a tool within the AutoTutor framework used to create AutoTutor scripts. ASAT-X is an extensible markup language (XML)-based tool. The ASAT-V tool is used to view and test AutoTutor visual scripts created by Microsoft Visio. Conversation rules can be very challenging for instructors, course managers, and domain experts. However, the AutoTutor Lite authoring interface is more intuitive. The tools are available as binary code.

ASPIRE is an authoring environment for developing constraint-based ITSs, which can be used by instructors to author ITSs to supplement their courses. ASPIRE supports authoring of the domain knowledge. The use of this knowledge is key to development of the domain model which is the most complex and time-consuming part of an ITS to develop. ASPIRE uses automation and intelligent support to guide authors through the authoring process. In ASPIRE, authoring consists of seven steps (aspire.cosc.canterbury.ac.nz/ ASPIRE-Author.php), some of which are beyond the capabilities of instructors, course managers, and domain experts without the intervention and support of the artificial intelligence (AI)-based scaffolding. A goal of ASPIRE is to allow non-computer scientists to author ITSs.

The *SPA Tools* support the definition of learning objectives, the development of learner measures and assessments, and the design appropriate feedback and scaffolding for reflection and self-directed learning. The goal of SPA is to simplify the process of creating knowledge for automated assessment and feedback in virtual environments and, like AutoTutor, is targeted at training domains where virtual humans play an active role in tutoring. The developers of the SPA tools assert that authoring in an environment that closely emulates the learner's experience eases the technical burdens usually encountered with ITS content creation and improves authoring efficiency. SPA is not available to the public at this time.

The *GIFT authoring tools* currently consist of several separate open-source authoring tools (e.g., course, domain knowledge file, pedagogy configuration, survey) to support various elements of the authoring process. A unifying GIFT Authoring Tool (GAT) is being developed as of the publication of this volume along with cloud-based versions of the entire GIFT. A usability evaluation will drive the development of an intelligently guided authoring experience. The GIFT authoring tools differ from the other authoring tools discussed here in that the GIFT tools have been integrated with external toolsets like the ASAT to support dialogue-based interactions, which can be triggered by GIFT-based tutors, and the Student Information Models for Intelligent Learning Environments (SIMILE) to support assessments where serious games are linked to ITSs. GIFT also provides a tool for automatically evaluating the hierarchical relationships between concepts in text-based material to support rapid development of expert models and other domain knowledge for use in the authoring process. A goal of the GIFT authoring tools is to allow development of effective ITSs by domain experts with little or no knowledge of computer programming or instructional design. This toolset is intended to support authoring across multiple task domains, but will continue to explore opportunities to leverage and integrate existing toolsets. The GIFT authoring tools, along with the rest of the GIFT software (source code), are freely available at www.GIFTtutoring.org.

A Vision for Authoring Capabilities

While it is obvious that we may never realize a single authoring toolset for ITSs, we continue to strive for authoring toolsets that are easy to access and use, and support authoring in multiple task domains (cognitive, affective, psychomotor, and social) resulting in a variety of ITSs (constraint-based, model-tracing, dialogue-based, agent-based). For these reasons, our vision is for a shell tutor or architecture where a variety of ITSs can support training in a variety of task domains.

Customized interfaces are needed to support improved usability novice, journeyman, and expert level authors. To support ease of use, intelligent agents would be used to guide human authors through the process where automation is not practical. The authoring process for this ideal toolset would also be heavily focused on process automation to reduce the burden of content and domain knowledge development to maximum extent possible. Usability and automation in the authoring process are discussed in more detail below.

Enhancing the Usability of Authoring Tools

We chose to examine the authoring process as a domain in which the author is being tutored with respect to best practices and the final ITS product. Using Nielsen's (1994) 10 usability heuristics, we discuss how authoring tools might be improved to support tailored interaction with authors of varied capabilities. We begin by examining the *visibility of system status*. In guiding the authoring process, the system should keep authors informed about the impact of their decisions on the final product, and feedback should be provided in a timely manner.

Next, we examine the *match between system and the real world*. If the author has a background in instructional design, it is desirable to use words, phrases, and concepts familiar to that author and provide information and guide steps in a natural and logical order based on knowledge of the process. What we are describing here is a tailored interface based on a user model that describes their capabilities and preferences.

Another desirable characteristic for our authoring tool interface is centered on *user control and freedom*. The ideal authoring system should support easy undo and redo functions without having to through multiple steps. For our purposes, this means the authoring system will be required to track previous authoring states in much the same way that Microsoft Office products save previous states of Word, PowerPoint, and Excel in memory. Given the ITS authoring process is more complex than an Office document, the specific schema to determine what to keep in memory and how often to update the model will require some research.

Consistency and standards should be realized across all user interface elements. Words, situations, and actions should mean the same thing throughout the user interface. Our authoring interface should also have mechanisms for *error prevention* either by alerting the author through error messages or by checking for errors through agents and then presenting confirmation options to the author before allowing the author to commit to an action. If an action is not permitted, then it would be desirable to have a rule to exclude it. If errors occur, the authoring system should *help the users recognize, diagnose, and recover from errors*. This should include as a minimum some *help messages and documentation*. Documentation should be easy to search, focused by the author's context (where they are in the process), and include a list of concrete steps.

An intelligent troubleshooting mechanism is a desirable authoring tool feature and should include constructive options to solve the problem as well as identify it. One option to develop a library of common errors is to collect user interaction data over time (big data) and mine that data to identify and document common errors and solution options. User-generated content (social media) may be another option for evaluating the effectiveness of solutions.

The heuristic states that the user interface should minimize the author's memory load by making objects, actions, and options visible. The author should not have to remember where a control is or what the next step is in the process. Standards should be developed for ITS authoring controls/objects. Where there are universal graphics for controls (e.g., undo), these symbols should be used instead of creating new, ITS-unique symbols.

Next, we examine the *flexibility and efficiency* of user interfaces for authoring ITSs. The interface should be sensitive to different types of users, their capabilities, and their limitations. Authoring tools should be able to select default conditions for novice users who may not understand the impact of these decisions. The selections made by the system are not seen by the novice user, but may be selected and changed by more experienced authors. Authoring tools should also be able to support shortcuts for frequent actions.

Finally, authoring user interfaces should be *aesthetic and minimalistic*. They should not contain irrelevant information, which contributes to extraneous cognitive load and reduces available resources for processing germane and intrinsic workload. Every extra bit of information competes with the relevant information and diminishes their relative visibility to the author. It may be useful for future authoring systems to reveal additional information to the user when the object, action, or option becomes relevant based on where the author is in the process.

Automation to Enhance Reuse and Reduce Authoring Burden

While the usability discussion above focused on the author's interface with the author tools, this section argues the merits of automation to take the human out of the authoring loop and support the search, retrieval, curation, and development of content and other domain knowledge. Metadata standards are needed to tag content objects for reuse. Intelligent search methods would use this metadata to find, retrieve, and curate appropriate content to support instructional objectives set by the author. Intelligent search would reduce the workload and skill needed to author effective ITSs.

Another area of reuse may be in the design and publishing of standard interface specifications for ITSs. As part of its architectural description, GIFT has published an interface control document, which describes how to push and pull data from GIFT and support real-time interaction with external training platforms (e.g., serious games, virtual simulations). If we describe adaptive training systems in terms of interactions between the learner, the training environment, and intelligent agents within the ITS, being able to reuse external training platforms in conjunction with an ITS reduces the burden of creating a problem space for each individual training scenario, but still allows for an AI to drive instructional decisions and provide tailored training.

Automatic authoring techniques would also allow authors to create content without humans in the loop. For example, GIFT currently has an authoring tool to rapidly develop expert models, which can automatically analyze a text-based corpus and generate a hierarchical representation of the concepts in that corpus. This can be used to generate an expert model and other domain knowledge thereby reducing the authoring burden.

Influence on GIFT Authoring Tool Design

As noted, the major challenges for the ITS authoring process are the time, cost, and skill needed to author effective ITSs. Based on the usability heuristic and automation discussions above, we have identified goals for the GIFT authoring tools as follows:

- Develop an authoring tool user interface that supports Nielsen's usability heuristics and allows instructors and course managers to develop effective ITS without knowledge of computer programming and instructional design.
- Create tools and methods to identify best authoring practices through the mining of user-generated content.
- Develop and publish GIFT metadata standards to support the search, retrieval, and curation.
- Develop search, retrieval, and curation tools to support the reuse of appropriate domain content.
- Examine the end-to-end process to identify the cost of developing ITSs and examine opportunities to automate elements of the authoring process where practicable.
- Create automated authoring tools and validate their performance.

Perspectives on Authoring Tools and Methods

The following chapters in this section discuss various perspectives on authoring tool. In Chapter 2, Dr. Tom Murray discusses a theory-based approach to authoring tool design. Dr. Murray is well known for his work in ITS authoring having conducted extensive reviews of authoring tools (Murray, 1999; Murray, 2003). In Chapter 3, Dr. Benjamin Bell compares and contrasts authoring tools for different ITS genres. Finally, in Chapter 4, Drs. Benjamin Nye, Benjamin Goldberg, and Xiangen Hu discuss design considerations for authoring tools across various tutoring/training domains.

References

- Graesser, A. C., Franklin, S., Wiemer-Hastings, P. & The Tutoring Research Group. (1998). Simulating smooth tutorial dialog with pedagogical value. In *Proceedings of the American Association for Artificial Intelligence* (pp. 163–167).
- Koedinger, K. R., Aleven, V. & Heffernan, N. T. (2003). Toward a rapid development environment for cognitive tutors. In *Proceedings of the 11th International Conference on Artificial Intelligence in Education, AIED 2003* (pp. 455–457).
- Mitrovic, A., Martin, B., Suraweera, P., Zakharov, K., Milik, N., Holland, J. & McGuigan, N. (2009). ASPIRE: an authoring system and deployment environment for constraint-based tutors. *International Journal of Artificial Intelligence in Education*, 19, 155–188.
- Murray, T. (1999). Authoring intelligent tutoring systems: An analysis of the state of the art. *International Journal of Artificial Intelligence in Education*, 10(1), 98–129.
- Murray, T. (2003). An overview of intelligent tutoring system authoring tools: Updated analysis of the state of the art. In T. Murray, S. Blessing & S. Ainsworth (Eds.), *Authoring tools for advanced technology learning environments* (pp. 491–545).
- Nielsen, J. (1994). *Usability Engineering* (pp. 115–148). San Diego: Academic Press.
- Sottilare, R. A., Brawner, K. W., Goldberg, B. S. & Holden, H. K. (2012). The Generalized Intelligent Framework for Tutoring (GIFT). Orlando, FL: U.S. Army Research LaboratoryHuman Research & Engineering Directorate (ARL-HRED).
- University of Southern California. (2013). Situated Pedagogical Authoring (SPA). Playa Vista, CA: Institute for Creative Technologies (ICT).

CHAPTER 2 Theory-based Authoring Tool Design: Considering the Complexity of Tasks and Mental Models

Tom Murray

School of Computer Science, University of Massachusetts

Introduction

In this chapter, I propose some theoretical foundations for future authoring tool design, focusing on operationalizing the construct of complexity—for tool, task, and user. Intelligent tutoring systems (ITSs) are highly complex educational software applications used to produce highly complex software applications. ITS authoring tools are major undertakings and to redeem this investment it is important to anticipate actual user needs and capacities. I propose that one way to do this is to match the complexity of tool design to the complexity of authoring tasks and the complexity capacity of users and user communities. Doing so entails estimating the complexity of the mental models that a user is expected to build in order to use a tool as intended. This chapter presents some exploratory ideas on how to operationalize the concept of complexity for tool, task, and user. I draw from the following theories and frameworks to weave this narrative: complexity science, activity theory, epistemic forms and games, and adult cognitive developmental theory (hierarchical complexity theory).

ITS Authoring Tool Design Tradeoffs

This chapter builds on earlier work (now over a decade old) describing the “state of the art” in ITS authoring tools research and development (R&D) (Murray, 2003). It does not provide any updates on the state of R&D in this field¹, but rather takes a perpendicular tact to look at some fundamental issues in authoring tools design. We start with a review of the design tradeoffs in creating ITS authoring tools.

ITSs are highly complex educational software applications (or learning environments) that can include the following components: user interface (which might include a simulated phenomenon or task environment), Expert Knowledge Model (of the task and/or knowledge), learner knowledge model, pedagogical model, and curriculum model (also collaborative learning environments may include group-level aspects of any of these) (see Woolf, 2010). For several decades developers and researchers have been investigating the possibilities for creating ITS authoring tools because these are hoped to (1) reduce the effort and cost of building or customizing ITSs, and (2) allow non-programmers, including teachers and domain experts (and even students), to participate fully or partly in building or customizing ITSs (Murray et al., 2003; Aleven et al., 2006; Suraweera et al., 2010; Constantine et al., 2013; Ainsworth et al., 2003; Ritter & Blessing, 1998).

There are many design tradeoffs involved—the primary one being that, in general, the easier or more efficient a tool is to use, the more simplistic or constrained are the ITSs that can be built from it. Trivial examples at two extremes are a tool that allows the author to select among checkboxes and lists to order and toggle and sequence features and curriculum items in an otherwise fixed system vs. a tool that is so complicated and multi-featured that building an ITS with it is not much easier than traditional software programming. One can imagine a design tradeoff space (a triangle) among *usability, depth, and flexibility* (see Murray, 2004). Depth, which refers to the structural or casual depth of any of the ITS models (listed

¹ For more recent work in the field, see Aleven & Sewall, 2010; Cristea, 2005; Olsen et al. 2013; Specht, 2012; Suraweera et al., 2010; Mitrovic et al. 2009; Sotilare et al., 2012, 2014; and the chapters in this edited book

above), is usually at odds with flexibility, which is the ability to author a diversity of types of ITSs. Usability is usually at odds with both depth and flexibility, i.e., a system that facilitates building deep models or many types of models tends to be more powerful yet less usable. A main theme of this chapter is to provide some rough metrics to help with these design tradeoffs.

Toward Theoretical Foundations

Unlike educational software (including ITSs), whose user audience is relatively well defined and known, the target users of authoring tools are less well defined and understood (unless the tool is intended for in-house use by a few specialized personnel, in which case, it has limited value as a research case study or data source). The main point of authoring tool (academic) research is to produce results that are generalizable to questions of ITS creation/customization related to production efficiency and accessibility by a non-trivial cohort of potential authors. That is, descriptions of new systems and innovations should be framed in terms of results, principles, or lessons learned that are relevant for other projects. Though efficiency is an important concern, I focus on usability in this chapter.

We can draw from the standard literature on usability for tool design principles, which is important but relatively straightforward, but in addition there are some more *theoretical* issues specific to authoring tools (of any sort, not just for ITSs) that I find quite interesting. Influenced by topics I have studied since my early papers on the subject, I have come to believe that a key issue is in how one matches the complexity of the authoring task to the complexity of the tool and the complexity capacity of the target user. Thus, in the bulk of this chapter, I sketch some preliminary considerations and principles that, though quite speculative, are intended to initiate inquiry in this direction.

Taking a more theoretical approach to ITS (or any) authoring tools is rarely if ever done, but my goal here is to point toward possible theoretical foundations for the (sub-) field. “Theory” can sometimes refer to a mere conceptual framework (without any underlying causal theory), but here I mean cognitive, social, epistemological, and/or information science theories that provide theoretical underpinnings. These areas of foundational theory (especially the learning and cognitive sciences) are now routinely considered in the design of ITSs and other educational software, but are rarely brought into discussions about the design or use of authoring tools.

Design science and usability theory draw on socio-cognitive theories to explore the relationships between the design of artifacts and the needs, capabilities, and limitations of intended users (and other stakeholders) (see Oja, 2010; Norman, 1988; Nielsen, 1993). Originally, these theories were in response to the (now more accepted) realization that domain experts (those who are not instructors), traditional software architects, and academics all historically have difficulty predicting or imagining the needs and limitations of the average software user and the average real-life task scenario (or difficulty predicting the *range* of users and task scenarios). Thus software design, and artifact design, in general, is increasingly understood as needing (1) empirical trial-and-error development, (2) the skills of rigorous empathy and imagination to put oneself in the shoes of a range of types of users and situations, and (3) some basis in underlying psycho-socio-technical theory (Brown & Campione, 1996; Cobb et al., 2003).

As mentioned, user-centered design (#1, 2 above) is important but may not lend itself to scholarly advances in authoring tools, but a more theoretical perspective should constitute a contribution to authoring tool design. The notion of assessing and coordinating complexity among tool, task, and user is a central theme in this particular theoretical exploration. In what follows, I first reflect on the factors leading to my 1999 article on authoring tools. I then consider some challenges facing authoring tool researchers today. Then, in the remainder of the chapter, I propose some theoretical foundations for future

authoring tool design. As mentioned, I draw from the following theories and frameworks to weave this particular theoretical narrative:

- Complexity in software design
- Activity theory
- Epistemic forms and games, and
- Adult cognitive developmental theory (i.e., hierarchical complexity theory).

Theories of complex software design are used to emphasize some of the issues, because ITS authoring tools are *complex artifacts designed to produce complex artifacts*. Complexity science also helps us operationalize what is meant by complexity in general. Activity theory, which highlights the relationships between an artifact and its usage-tasks, usage-rules, and community of practice, provides an orientation and basic vocabulary for the task of ITS design by various types of users in an authoring role. We can ask whether a tool and its “rules” of use afford the accomplishment of a particular task for a particular class of users. Much of the process of matching tool/task complexity to user (and community) complexity capacity revolves around the complexity of the *mental models* that a user is expected to build in order to use a tool as intended. Colin’s work on *epistemic forms and games* provides a highly useful framework for talking about this tool-rule-user match in holistic terms at the right level of granularity. At this point, we have a framework for describing many *sources* of complexity in tools, tasks, and users (cognition or mental models), but no good way to order or coordinate these types of complexity. For that, we draw on hierarchical complexity theory and related theories of adult cognitive development to suggest this order as a final step in matching the complexity of an authoring tool to the complexity capacity of its target users.

Challenges Facing Authoring Tool Research Today

Predicting Future Flying Machines

ITS authoring tool research is in an interesting socio-techno-historical position. Intelligent tutors, despite 30 years of R&D, are not yet common in mainstream education or training, though a few notable systems have achieved wide-spread use (Koedinger et al., 1997; Heffernan & Heffernan, 2014; Graesser et al., 2005; VanLehn et al. 2005; Mitrovic, 2012; Johnson et al., 2008; Sitaram & Mostow, 2012). This may be a completely appropriate development and adoption arc for a technology this complex and innovative, and we have every reason to believe that the results of ITS (and more generally advanced technology learning systems (ATLS)) research will continue to influence on-the-ground, computer-mediated learning. However, authoring tool researchers are in the awkward position of developing the cart before the horse, or worse yet, developing the cart-factory before the horse. It is as if, as the Wright brothers were experimenting with the first airplanes, a group of researchers and academics were observing on the side, working out how to design airplane factories that would make airplane production efficient and flexible. As those first manned flight contraptions were being developed, it would have been difficult to predict what future flying machines would look like, never mind what the market would be like or how to best mass-produce and easily customize them for typical users.

Of course, ITS work is well beyond its first prototypes, so this analogy is stretched. Still, authoring tool designers work under considerable uncertainty as to what types of systems will find their way to substantial use and benefit from the scale and flexibility that authoring tools enable. However, we are talking about software here, not equipment manufacturing. Building abstractions and design tools is a

natural impulse in software design (procedural-, data-, and knowledge-abstraction) are basic computer science principles; see Abelson & Sussman, 1983). As indicated in the history of my own projects, it can be beneficial to build authoring tools merely to facilitate local or small-scale R&D projects. A company that makes a decent profit on one single piece of widely used software (say, an ITS) would benefit from building authoring tools to customize and enter content for the ITS. However, the less generic the system, the more difficult it is to frame *research* questions and findings (especially after others have mapped out the territory).

Old vs. New Conceptions of ITSs

The original understanding of computational “intelligence” in ITSs involved mostly modeling and knowledge representation tasks (or challenges)—learner, domain, and instructional models. The more deeply cognitive science understands knowledge and learning (or finds how little it does understand), the more difficult these modeling tasks appear for authentic situated tasks. In general, the most successful ITSs are those focusing on knowledge that is the easiest to represent, including declarative facts and procedural steps (simple skills, which create complexity as they are combined). Yet developments in learning theory increasingly emphasize the importance of less representable forms of knowledge, such as metacognition, conceptual understanding, problems solving, open-ended inquiry, collaboration, communication, argumentation, hypothetical and analogical thinking, etc.

The more basic forms of knowledge (fact, skills, and concept-map-like relationships) continue to have fundamental importance as building blocks for more sophisticated skills, but the more exciting work in ITS/ATLS has been moving into a wide variety of areas that do not involve “deep modeling” of knowledge or expertise. These new research trends include recognizing and responding to affect; using big data to classify and predict learner behavior (without trying to create runnable models per-se); wearable gadgets; immersive experiences; natural language understanding and production; gamification; and socialmediafication. For a project to be considered “ITS” research, it no longer requires computational intelligence per se, but only the inclusion of some state-of-the-art computational technology (or leading-edge techno-socio-psycho theory). While the idea of a generic ITS framework requires some commonality of basic components and/or representational frameworks, the scope of ITSs is becoming increasingly diverse, and overarching frameworks are increasingly difficult to envision. However, one could counter that as diversity increases, so does the number of projects, so that the actual impact of designing generic frameworks still serves a significant (if smaller percentage-wise) potential user base.

Toward Design Theories

Authoring tools are still essential for scale-up, wide adoption, and easy customization of learning systems, though each may need to be specific to a very specific genre of instructional systems. If so, authoring tool design may become more of an engineering challenge than a research area. However, there are still important theoretical issues that can be investigated, which we explore next.

Engineering challenges involve figuring out how to apply general theories, methods, or principles to specific contexts. These challenges are no less arduous and important, as design principles tend to be rather abstract, and nailing down “how the rubber meets the road” in each context can be the bulk of the work. Also, because theory must ground in and remain responsive to actual examples, ideally there is an ongoing dialogue allowing general principles to be informed by the various methods that have been used to apply them to practical contexts.

Software Usability and Complexity

Usability and Managing Software Development Risks

Bracketing the above concerns, let's assume that ITSs of some sort will indeed become mainstream and that authoring tools will become increasingly important—a safe bet, I think. Other than tools designed for in-house use by highly trained specialists, authoring tools, by their nature, must be usable by some anticipated user audience. As mentioned, with any tool there are *context-specific* usability concerns that can be worked out through good design practices (prototyping, early feedback from authentic users, etc.), but here I look at very *general* usability concerns, having to do with the complexity of these systems.

ITSs are complex software applications and full-featured ITS authoring *tools* can be an *order of magnitude* larger and more complex—just as a machine designed to build many types of lamps is much more complex than a lamp (though the machine itself may be relatively easy for the end-user/author to use, its interiors will be more complex). Next, we look to the literature on the *design and usability of complex software systems* for advice relevant to ITS authoring tool design. This is a first step in imagining a more theory-driven approach to authoring tool design.

Design tasks such as authoring ITSs fall under the “ill-defined” and “wicked” problems characteristic of real-world projects (Conklin, 2005; Mirel, 2004). In his treatment of usability of complex systems, Oga (2010) defines complex software development in terms of Mirel’s definition of complex problem-solving, which involves “ill-defined situations; vague or broad goals; large volumes of data from many sources... nonlinear, often uncharted analytical paths; no pre-set entry or stopping points; many contending legitimate options; collaborators with different priorities; [and] ‘good enough’ solutions with no one right answer.” Chilana et al. (2010) give three additional factors that contribute to the complexity of designing usable software: domain-specific terminology, every situation is unique, and limited access to domain experts. ITS/ATLSs and their authoring tools certainly have all these characteristics.

Oja contends that Nielson’s classic usability heuristics are even more critical for *complex* software development (Nielson, 1994). Nielson’s usability heuristics include reification (visualizing key abstractions and relationships; minimizing working memory load); user control and freedom (not constraining user actions any more than is necessary); flexibility in outcomes (allowing for variations in style and needs); match between system and the real world (using the vocabulary and mental models users already have); assistance with helping users recognize, diagnose, and recover from errors; and efficiency of use.

Echoing the heuristic to “match between system and the real world,” Johnson (2006) analyzed software usability failures in the healthcare sector that imposed significant financial and acceptance burdens within that sector and found that “many usability problems stem from the inability of suppliers and manufacturers to anticipate [user] requirements.” The educational technology R&D community is poised to create ITS authoring tools that could be used on a large scale. As the investment in authoring tools increases, there is a corresponding increased “risk” that investment in design, outreach, etc., will outweigh the benefits if the tools do not directly meet the needs of a wide variety of users (or if the ITSs built with the tools do not reach a large number of learners).

Figure 1 illustrates the type of risk management and risk reduction principles increasingly being used in software and other industries.² Additional investments in software can follow the “80/20” rule, where

² Image adapted from “Risk Management in the (Bio)Pharmaceutical and Device Industry,” L Huber & Labcompliance Inc., http://www.labcompliance.com/tutorial/risk/default.aspx?sm=d_a.

perfecting the last 10% or 20% can take a disproportionate amount of effort. Meanwhile, the return on user value gets proportionately less. The goal is to find the sweet spot where risk is acceptably low and expected value is relatively high (“optimum” in Figure 1). To mitigate this risk, usability principles recommend both empirical and theoretical grounding: i.e., usability evaluation and user-feedback from authentic contexts done “early and often;” and a good theoretical understanding of the user and task. Complexity is a useful construct for operationalizing Johnson’s “[ability] of suppliers and manufacturers to anticipate [user] requirements,” but the construct needs better definition for this to happen—which is what we hope to contribute to here.

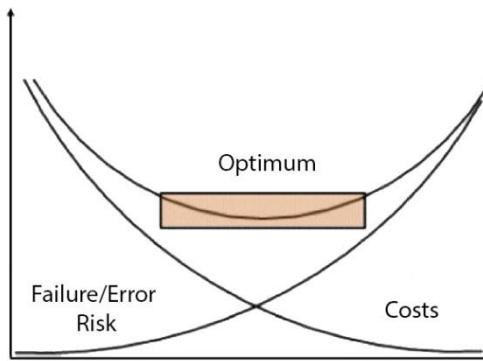


Figure 1: Cost vs. value in software risk assessment

Complexity Science and Information Theory

Next we branch away from complexity in software and usability theory to consider how complexity is theorized in more general terms. Complexity science points to various methods for measuring complexity, which are all related to the amount of information contained in an object, system, or process, with “information” being closely related to the concepts of difference, discernibility, and degrees of freedom. Information and communication theories also quantify information (even “meaning”) in terms of entropy, randomness, chaos, “surprise,” and “shortest possible description” (Grünwald & Vitányi, 2003). There are many individual metrics that contribute to overall complexity, including the number and diversity of components and their structural or functional relationships (Benyamin & McKelvey, 2006). Complexity science also deals with time-based phenomena: change, feedback loops, self-organization, evolution, and emergence in dynamic systems—so-called “complex adaptive systems.”

Campbell (1988) describes three sources of complexity: number of *dimensions* of information, the *rate* of information change, and the number of *alternatives* associated with each dimension (i.e., information diversity). We modify and generalize this scheme as in Figure 2, using the categories of structural, dynamic, and perspectival complexity.

- Structural complexity
 - No. of properties
 - No. of parts
 - No. of types of parts
 - No. of relationships
 - No. of types of relationships
 - No. of steps
- Dynamic complexity
 - Loops, Feedback, recursion, relationships (“non-linearity”)
- Perspectival complexity
 - Multiplicity/alternatives, uncertainty/hypotheticals
 - Variables, decision spaces
 - Stakeholders



Figure 2: Sources of system complexity

For structural complexity, other things being equal, systems are more complex if they have more parts (e.g., an ant colony or a huge Lego project); more types of parts (e.g., a car or human anatomy); more properties in each part; more relationships or constraints among the components (internally and with the external environment); and more types of relationships. In particular, one-to-one mappings (relationships) are the simplest, one-to-many mappings are more difficult, and many-to-many mappings are most complex to manage and conceptualize.

In addition to these structural dimensions (which are metaphorically space-like), systems whose properties, relationships, and objects *change* over time are more complex (the dynamic or temporal dimension). Dynamic complexity can be represented in terms of the laws, rules, mechanisms, or influences that create change in a system. Not only change but feedback loops and nonlinear dynamics, all outside our scope to elaborate on, come into play here.

As indicated above, complexity is related to information intricacy, space of possibility, and even “meaning,” and thus is not simply an objective property of systems, but has a quasi-subjective component that involves human context, activity and the reasons for doing the complexity analysis. In software, information systems and usability analysis, there are cognitive and epistemic considerations. Byström & Järvelin’s analysis of task complexity includes factors such as repetitively, analyzability, a-priori determinability, number of alternative paths, outcome novelty, number of goals and conflicting dependencies, uncertainties between performance and goals, number of inputs, and time-varying conditions of task performance (1995, p. 5). Zhang et al.’s (2009) “epistemic complexity” measures complexity in terms of the movement from facts to explanations and from unelaborated to elaborated knowledge—both of which indicate increasing depth and complexity. Epistemic complexity includes measurement of the “diversity” and “messiness” one encounters in a situation (Bereiter & Scardamalia, 2006). Thus concepts of nuance/subtlety, abstraction/ generalization, uncertainty/ambiguity must be considered.

Therefore, in Figure 2, we have the third category “perspectival” complexity, which is complexity due to multiplicity and uncertainty, including conflicting goals or subtasks; diverse perspectives among stakeholders; stochastic randomness and indeterminacy; and vagueness and uncertainty in any of the structural or dynamic elements (measuring these would be more heuristic than the other two complexity factor types). Perspectival factors relate as much to subjectivity and the nature of cognition as to the objective nature of the artifact.

Usability Complexity and Runnable Artifacts

In terms of software systems, specifically authoring tools, the factors mentioned above can be applied to the software artifacts (code and interface), development (programming or authoring), or the complexity of use (the user interface understanding and the mental model a user must acquire to understand a system). Theoretically, each of the sources of complexity in Figure 2 could be enumerated or estimated and combined to measure the complexity of a system (its code, interface, task, etc.) toward the goal of comparative analysis of the complexity of systems.

Software tools and applications allow us to make and improve things, which we call “authoring.” Artifacts that “run” or behave dynamically are, of course, more difficult to author. With authoring tools and educational software such as Scratch and StarLogo, and scripting languages in Office applications, the line between programming and using software is increasingly blurred. ITS authoring can fall anywhere along a spectrum of complexity from customizing parameters and choosing content to creating teaching strategies, which is closer to software programming.

ITSs are dynamic systems that must be run to test them. They have multiple learning paths and it is intractable to test every possible student behavior. Unpredictable behaviors inevitably occur in complex software (which is why rigorous testing is important). The simplest systems have predictable paths with little interaction or parameterization, such as scripts and story-board-type procedural flows. If an authoring tool allows branches, if/then rules, procedures, loops, parameterized subroutines, or recursion (in rough order of difficulty), the level of authoring complexity jumps dramatically. The author is essentially doing software programming. Writing and debugging computer programs is a complex task requiring special skill and tools. Without these skills, and even with them, it can be quite difficult to determine the source of a run-time software bug.

Creators of authoring tools that allow authors to enter into this level of task complexity must (1) not underestimate the complexity of the task or overestimate the skill of the typical user, and (2) provide real debugging and tracing tools for the systems to be viable. One of Nielsen’s (1994) “Top 10” recommendations for usability is to “help users [authors in this case] recognize, diagnose, and recover from errors.” This can be as simple as providing an Undo feature for authored content, but for systems with dynamic complexity special tools are needed to trace and debug procedural representations.

Like most software systems, ITSs should be designed in user-participatory feedback loops, where, as Benbya & McKelvey note, “the critical factor in all information systems is continual change” (2006, p. 20). This might even imply that viable authoring tools should have some sort of “version control” subsystem.

The above discussion suggests factors that could be considered in characterizing the complexity of software tasks and interfaces. It is implied that for some tasks, such as version control and debugging, there is a need for special skill such as knowledge engineering. Thus it is also important to consider the “complexity capacity” of users and communities of practice—and for this we turn next to activity theory.

Activity Theory—Users, Tasks, Tools, and Communities

We borrow concepts from activity theory, which stresses the mediating role of *tools* (artifacts) and their usage *rules* in collective human activity and development (Jonassen & Rohrer-Murphy, 1999; Stahl, 2006; Engestrom et al., 1999). Here *rules* indicate the (sometimes implicit) skills, understandings, and habits held by a community of practice. Thus, we can frame our exploration of authoring tool usability in terms of the interaction between *users*, *tools*, *rules*, and *tasks*. We can ask whether a tool and its “rules” of

use afford the accomplishment of a particular task for a particular class of users. Clearly, our *users* are authoring *tool* users and the *task* is to design or customize an ITS; later we introduce “epistemic forms/games” as a way to describe the *rules* of use.

Figure 3 illustrates these factors in activity theory terms (adapted from Jonassen & Rohrer-Murphy 1999; Engestrom et al. 1999). Thus, from our focus on the concept of complexity, we must consider the following:

- Task and rule complexity (user activity methods and goals)
- Tool (artifact) complexity
- Socio-cognitive complexity (community of practice and division of labor)

We are concerned with the match between the following:

- User vs. tool complexity
- Task vs. user complexity
- Community of practice vs. tool complexity

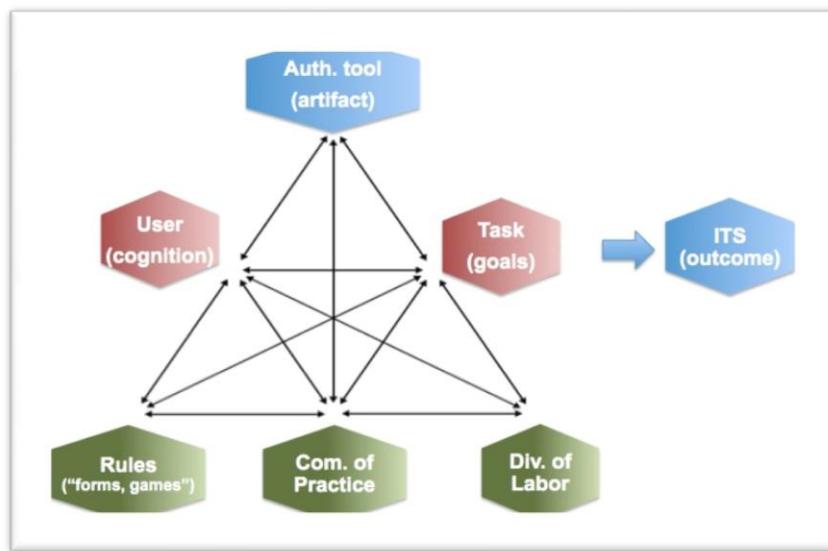


Figure 3: Activity theory

When we speak of users, we are really speaking of users in particular *roles*. This distinction is important when we begin to speak of the complexity capacity of a user (or type of user). We are not referring to a person’s general ability to handle complexity, but to one’s ability within a certain role (ITS author, content developer, tester, etc.), which might depend more on training and experience than on innate intellectual sophistication.

Campbell notes that there are several approaches to assessing complexity: as a subjective psychological experience of the user, as an objective measure of the task, and as an *interaction* between subjective and objective elements (1988, pg. 44). While measuring complexity in terms of user (author) *experience* is

important, methods for doing so are outside our scope here. However, we describe methods for describing user *capacity*, and we assume that, on average, complexity capacity is closely related to the complexity experience of the user (they will be frustrated or confused if their complexity capacity in a particular role is mismatched for the task). In the prior section, we outlined specific methods for assessing task and tool complexity objectively (though perhaps heuristically as estimations). Our eventual goal is to assess the match (or interaction) between user capacities and the measures of tool/task complexity (user capacities is roughly estimated, while tool/task complexity affords a more objective measurement).

Note that in the prior section tool and task complexity were treated together. Unlike simple tools such as a hammer, for which the task a tool is used for (e.g., building a barn) is usually much more complex than the tool itself, for most software tools, the complexity of the tool features can stand as a fair indication of the complexity of the task. This is, of course, not strictly true, as building an ITS involves much more than using an authoring tool (e.g., applying learning theory, paper mock-up design, etc.), but for simplicity we assume that the complexity analysis given above of artifacts (tools) maps well to complexity analysis of tasks. Task-related issues of *how* the tool is used and learned are categorized in rules or community of practice (COP) elements of activity theory, rather than with the artifact.

Epistemic Complexity and Complexity Capacity

Oja quotes Haynes and Kannampallil (2004) who say that “complex software applications require great cognitive skill, integration of knowledge from various areas, and advanced instruction and learning; thus, it is not surprising that ‘screen deep’ interfaces to such systems may not yield the best results in terms of usability.” This is one reason why understanding the intended user is so important—because making a tool more easy to use, i.e., “usable,” may dumb it down too much for some users or tasks, and decrease “user control and freedom” and “flexibility and efficiency of use” (from Nielson’s model) for those contexts. Oja (2010) noted, “As Mirel [2004] points out, most current HCI practices concentrate on ease of use or simplifying the work, and this may lead to ‘producing good designs but for the wrong problems’” (p. 3800). The design goal is thus to make tools “operationally simple, while intellectually sophisticated and nuanced” (Mirel, 2004).

“Cognitive complexity” is one term used to describe a person’s capacity to perform complex mental or behavioral tasks. Cognitive complexity involves not only the number and complexity of the objects and relationships as described above, but also the ability to perceive nuances and subtle differences, i.e., it can involve both integrative and differentiating capacities (Mirel, 2004). Jordan uses the term “complexity awareness” for “a person’s propensity to notice...that phenomena are compounded and variable, depend on varying conditions, are results of causal processes that may be...multivariate and systemic, and are embedded in processes [that involve non-simple information feedback loops]” (2013, p. 41). As mentioned above, Zhang et al. (2009) use the term “epistemic complexity,” which includes an understanding of underlying reasons, theoretical explanations, or hidden mechanisms within phenomena. In what follows, I use the term “complexity capacity” to remind us that cognitive complexity required for a task is about the context and role a person is in, and depends on experience in addition to any general complexity “intelligence” they may have.

In the exploratory discussion of software usability and complexity, I enumerated many factors and it remains for future work to determine how these factors are operationalized, weighted, and combined in any overall complexity metric (a process that may be quite context-specific, as complexity components will have different weights for different situations). As we move from characterizing the complexity of tools (artifacts like software) and tasks (in this case authoring) to that of users, my approach continues to be preliminary and suggestive, with many details remaining to be worked out beyond this chapter. Let’s assume, for simplicity, that we have worked out the details of a scheme such as the one described in prior

sections of this chapter, have devised a method to characterize *task/tool* complexity level, and have collapsed the dimensionality of analysis to rate tasks/tools on a scale of low/medium/high complexity. How might we map this to *user* (or community of practice) complexity capacity? Table 1 illustrates what such a mapping *might* look like, showing types of authors, benefits, and problems typical of each author type, and the level of design complexity one can typically expect in the authoring task.

Table 1 Authoring tool user roles and complexity capacity estimates

Roles (tool use roles)	Benefits (of that role)	Problems (of that role)	Complexity Capacity for ITS Design
Teachers PRACTICAL	Practical experience	Not good at articulating or abstracting expertise	LOW
Domain Experts and content developers PARTIAL	Auth. tool infers the instructional methods	A fixed instructional method	MED
Instructional designers and learning theorists THEORETICAL	Know learning theories and research	Rare; not trained in knowledge engineering	MED
Knowledge engineers and ITS developers EXPERIENCED	Know the tools; are sometimes also plugged into user testing	May not know what it is like to teach or learn the material	MED-HIGH
Computer scientists and software developers (ACTUAL?!)	Complexity capacity. Don't have to build to a real user base.	"it's intuitively obvious to the casual observer..."	HIGH

Teachers have on-the-ground experience of the needs of students and classroom situations, and, while their input should be included in the iterative design process, they cannot be expected to have the skill, nor the time, to use (or learn how to use) complex authoring tools. Domain experts and content developers are more typically used to define knowledge and expertise, though they may have little practical or theoretical knowledge of pedagogy. Instructional designers and learning theorists bring different sources of pedagogical knowledge and epistemological knowledge (understanding how knowledge is structured), though they will often not have the time to dedicate to a steep tool learning curve.

For all of the above user types, the task of representing knowledge in a computationally usable fashion may be foreign—while *knowledge engineers* are trained in exactly that task. It is only with this level of skill and higher that we can expect sophisticated authoring tasks to be managed. Most user communities do not have people with knowledge engineering (or ITS design) skills, meaning that users at this level are usually part of a dedicated ITS design team, which would only exist in an academic lab, a company dedicated to building learning systems, or an educational organization large enough to form such a team to be shared widely (e.g., a university or city school district).³

³ Note that this specific scheme is suggestive and meant to illustrate a framework rather than the “content” of the framework—i.e., I do not need to make a strong argument here that, e.g., “domain experts and content developers” have a limited or “fixed” understanding of instructional methods, as is given in the Table. Of course, the roles in the

The final category of users in Table 1 is computer scientists and software developers. This category connotes the unfortunate yet understandable fact that many ITS authoring tools never see a robust user community and are only used within the confines of the team or organization that built the tool. This stakeholder group tends to be the most sophisticated in terms of designing complex structural and procedural models. The benefit is that more powerful ITSs can be built, but the drawback is that without usability input from “real” users, the tools may be too complex to expect many others to pick up, and the tool designers may be out of touch with the needs of intended users.

In authentic contexts, the actual “capacity” of a user to use a tool to accomplish a task depends on “community of practice” considerations as well as the potential complexity capacity level of the individual (see Figure 3). These considerations include (1) opportunities, investment, and incentives in *training*; (2) community of practice peer and mentor *support*; and (3) *time* available to author. Thus, even if a user, say, an unusual teacher, has a high level of generic complexity capacity, in order to successfully make use of an ITS authoring tool that person would need to be able to invest time in the learning curve, have the support of peers and superiors in adopting this new technology, and have the ongoing time available to do the authoring (along with other job responsibilities). Contexts satisfying these conditions are indeed rare.

In addition, for newly introduced artifacts, there is a dynamic, often evolutionary, interplay between artifacts (their design), the standard and novel ways that artifacts are put to use, and the human capacities enabled by artifacts. That is, new tools create new capacities, which create new possibilities and new goals/tasks, around which new (or improved) communities of practice develop—all of which, in turn, prompt new innovations (tools) to continue the cycle. Benbya & McKelvey (2006, p. 14) refer to the “co-evolutionary” aspects and “adaptive tension” of the “complex adaptive” socio-technological systems and discuss the problem of “accumulating requirements.” So, an important community-of-practice question is, How effective are the feedback and *development learning loops* between users, trainers, and designers?

Thus far I have described what a tool/task/user complexity mapping scheme might look like, without saying much about the nature of user cognitive complexity. A user’s understanding of tools, tasks, and methods can be described in terms of the *mental models* one has of these things (Gentner & Stevens, 1983; Johnson-Laird, 1983). Mental models are cognitive representations of external systems that include structures and processes that a person simulates (runs or visualizes) mentally. One task of the authoring tool is to help the user construct a valid mental model of the ITS building blocks, range of configurations, and design steps that the authoring tool affords.

Oja notes that “cognitive engineering (Gersh et al., 2005) and learner-centered design (Soloway et al., 1994) focus on improving system-human cognitive fit and allowing users to construct better mental models (knowledge) of the system” (p. 3801), and that “reification is the basis for successful communication and the establishment of a shared goal in human-computer collaboration” (p. 3803). Thus, it is important that the authoring tool interface accurately and powerfully reify the structures, objects, constraints, decision rules, and procedures involved in authoring, so that authors can build correct mental models and can use these mental models to coordinate the various steps and roles within a design process. The complexity of mental model that is supported in the authoring tool should match the complexity capacity of the user.

Collins and Ferguson’s work on “epistemic forms” provides a valuable link between task/tool complexity and the user’s complexity capacity in terms of the mental models that the user must construct and

table can be combined in any individual, but it would be rare that, for example, a classroom instructor would also be a learning theorist or knowledge engineer.

maintain. Their concept of “epistemic games” also anticipates the community-of-practice element of activity theory. I discuss epistemic forms and games next.

Epistemic Forms and Games

Collins and Ferguson (1993) first articulated the concepts of epistemic games and epistemic forms (see also, Morrison & Collins, 1994; Shaffer, 2006). Epistemic *forms* are “target structures, like mental models, that guide inquiry” and are “recurring forms that are found among theories in science and history.” Epistemic *games* are “general purpose strategies for analysing phenomena in order to fill out a particular epistemic form” that are shared within a community of practice (Collins and Ferguson, 1993, p. 25). Example epistemic forms include lists, hierarchy or tree structures, tables, networks, if-then rules, and constraint-based systems. They are “generative frameworks with slots and constraints on filling in those slots,” and in this sense are like domain-independent scripts, templates, or grammars that specify the structural properties of a phenomena. They serve as commonly understood mental models for understanding tasks and tools.

The theory of epistemic forms/games considers not only the structure of information, but also the ways (i.e., games) communities use, understand, and build knowledge using that structure. For example, perhaps the simplest epistemic form is the list. Knowing how to play an epistemic game includes knowing its constraints, strategies, and moves. For the “list game,” this includes knowing how to add, remove, combine, split, and arrange (classify, filter, or sort) items, and knowing when the “list form” is most appropriate for a particular problem or inquiry. This framing is compatible with activity theory, which highlights the interplay between cognition, artifact design, and communities of practice.

Morrison and Collins (1994) coined the term “epistemic fluency” to refer to the ability to use and choose appropriately among the repertoire or ecology of epistemic games available within a community of practice. Epistemic games are rarely used in isolation and are combined with other games as well as transformed into other games, as when one representation (a concept network) is seen as more appropriate than another (a table). Tables can be seen as composed of lists; even more complex forms might combine tables with networks (e.g., a network of tables, or a table of networks). Table 2 lists some epistemic forms/games mentioned by Collins and Ferguson (1993).

Table 2 Epistemic forms and games (mental models)
(Collins & Furgeson, 1993)

list	street map
matrix or table	org. chart
molecular model	musical score
periodic table	timeline
web page menu	cause/effect diagram
x-y graph	network
pert chart	relational database
binary tree	sentence diagram
floor plan	term paper outline

Epistemic games can be framed in terms of the key questions driving an inquiry. Knowing an epistemic game includes knowing how to evaluate whether it is being played well. Example quality/validity criteria for the list game include coverage (is anything missing?), similarity (do the items belong together, or should they be split into two lists—apples and oranges?), distinctness (are the items actually different?), and perspicuity (is it sufficiently short, simple, efficient, and understandable?). Vibrant communities of practice can be creating, tweaking, and evolving, and mashing up their epistemic games.

Authoring Tool Epistemic Forms

Epistemic forms/games allow for a compact method of classifying tool/task complexity. In our original discussion of artifact complexity, I suggested that one could enumerate the number and types of parts, properties, relationships, etc., in a system. This may be useful to do but also quite cumbersome. Meanwhile, epistemic forms serve well as a first-pass description of the complexity of end-user software systems. Epistemic forms also address one difficult issue in the characterization of an artifact, which I call the “dimension compression problem:” it may not be difficult to classify and compare artifacts along any single dimension (as in Figure 2), but we have little guidance thus far on how to combine and prioritize the many dimensions into a single (or simple) complexity characterization. Epistemic forms are holistic and representationally efficient in that they incorporate many of these dimensions into each category.

In discussing authoring tools, I am interested specifically in design activities or *design games* (a term not used by Collins and colleagues). In all epistemic games, one of the evaluation criteria is whether one’s product (use of the epistemic form) is understandable or meaningful to others *within* one’s community, while *design games* are distinguished by the additional need to assess how understandable and useable the product will be to *users* (who belong to a community related to but different from the designer community). Thus, the set of design game quality/validity criteria is extended to a group that requires some cognitive empathy (and design/test iterations) to serve well.

In surveying a set of 14 authoring tools mentioned in Murray et al. (2003), one can clearly see a set of epistemic forms that are repeated numerous times throughout most of these systems. This list of forms is not be surprising—they are seen in most software tools, as shown in Figure 4. The basic elements include check boxes and choice lists; sliders, dials, and meters; graphical networks and trees; and interactive hierarchical and tabular textual representations. As discussed, to compare across and within any class of epistemic forms (say, a hierarchical menu system), we can use the elements suggested in the earlier discussion of complexity science, i.e., the complexity of an interface and task includes the number and diversity of such elements and the degree of their inter-relationship or coupling in an overall system.

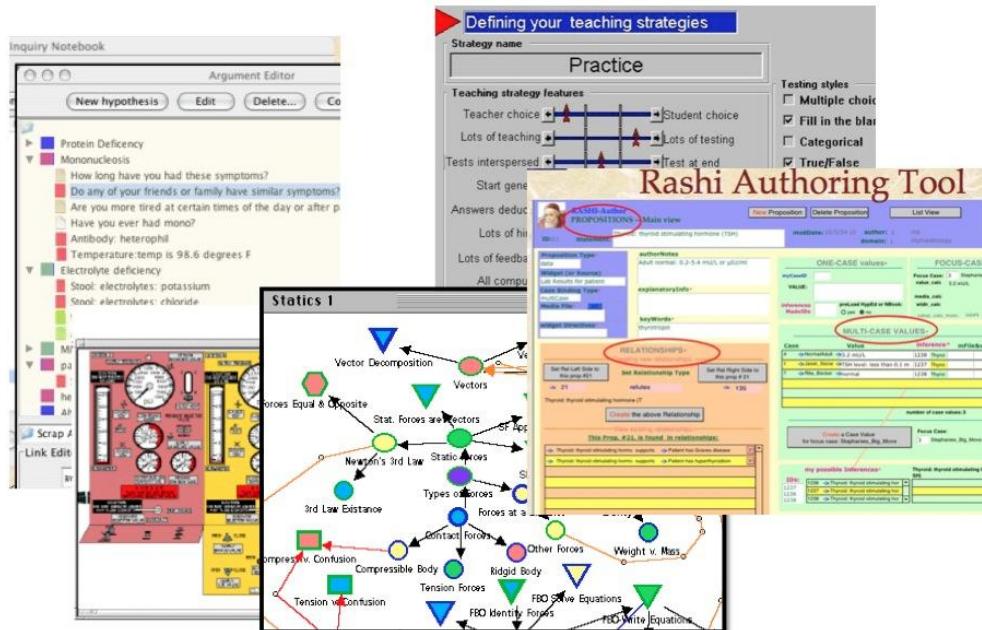


Figure 4: Epistemic forms in authoring tools

Intuitively, one can roughly compare or rate the complexity of epistemic forms. Lists, sliders, and checkboxes are simpler than hierarchies, tables, and concept networks, which are, in turn, simpler than the complex systems/mental models that are composed of dynamic interactions among many simple sub-components. *Hierarchical complexity theory* offers a more rigorous and more theory-based foundation for rating and comparing complexity components, and it was developed to apply to human tasks and skills. Next, I explore HCT as the last theoretical territory of exploration in my journey to link several interdisciplinary fields.

Hierarchical Complexity and Skill/Task Development

Above I drew from information/systems theories and socio-technology theories (activity theory and usability theory) to suggest ways to characterize the complexity of systems in general terms. Epistemic forms provide a way of ameliorating the “dimensionality issue” by enumerating common forms that are more intuitive and ready-to-hand than a list of low-level complexity dimensions. But we are still far from a quantitative or semi-quantitative method for combining the factors involved to be able to make comparative complexity judgments. To move in this direction, I draw from an area of cognitive/learning science that has significant implications for learning theory and ATLS design in general, yet, curiously, is rarely referenced in these fields: Neo-Piagetian developmental theories. Cognitive developmentalists (Neo-Piagetian theorists) have undertaken a deep study of complexity, because human development and learning can be described in terms of “qualitative differences in mental complexity” relative to various tasks, skills, or life contexts (Kegan, 1994, p. 152).

The key insight is that development, and complexity in general, advance through both horizontal and vertical (“hierarchical”) movement, and do so through a particular alternating or spiraling pattern.⁴ The structure and nature of horizontal growth is different than the structure and nature of vertical growth. Vertical growth is more quantized or punctuated, and the vertical leaps involve particular challenges. If we frame authoring tool features, tasks, and epistemic games in terms of vertical and horizontal differences in complexity, we have additional tools for comparing complexity, and we gain insight into why certain forms may be particularly difficult for users to learn.

Neo-Piagetian (adult) developmental theories go beyond early developmental work (e.g., Piaget, Perry, Kohlberg) to add a hierarchical “structural perspective in analyzing changes in the organization of “actions and thought” (Fischer & Yan, 2002, p. 283). These theories propose underlying representations for skills and suggest rules for the transformation of skills to higher-level skills.⁵ These theories apply principles from complexity science to human cognition and behavior, which can be easily mapped onto artifacts (tools). As stated by Commons & Pekker, “Theories of difficulty have generally not addressed the hierarchical complexity of tasks. Within developmental psychology, notions of hierarchical complexity have come into being in the last 20 years. [...] a model of hierarchical complexity, which assigns an order of hierarchical complexity to every task regardless of domain, may help account for difficulty” (2009; p. 2).

Horizontal increases in complexity involve adding more of what already exists to an object, process, or structure (more parts, relationships, steps, etc.—adding more “bits” of information without adding new structural emergence). Commons suggests that increases in the horizontal complexity of tasks (which he calls the “classical” model of information complexity) are analogous to increases in cognitive load

⁴ These developmental models are discussed in more detail in the appendix in Murray (2015).

⁵ Fischer’s Skill Theory (Fisher, 1980; Fisher & Yan, 2002) and other Neo-Piagetian models, including Commons’ Hierarchical Complexity Model (Commons & Richards 1984, Commons et al. 2008), Kegan’s stage model (1994, 1982); and Cook-Greuter’s ego development model (2000, 2005).

(Commons & Pekker, unpublished). Horizontal growth can also be roughly compared to Piaget's assimilation, as it adds new knowledge in the form of existing structures (Piaget, 1972). Vertical growth relates to accommodation, in which new structures are created to understand the world in new ways. Horizontal growth tends to be continuous, while vertical growth follows a more discrete model and occurs after a sufficient amount of horizontal growth allows for a reorganization at the next higher level.

Vertical increases in complexity lead to a new level or stage by applying an operation upon, or "coordinating and transforming," the objects of the lower layer. Each artifact or skill at a given hierarchical level consolidates a set of items at the lower level into a single whole, transcending and yet including them. Completely new properties and concerns arise at each level (a phenomena called emergence). Examples of increasing levels of hierarchical complexity include the development (or evolution) from words to sentences; addition to multiplication; single celled to multi-celled organisms; concrete to formal operational concepts; using to designing an artifact; and doing a task to managing others doing it.

There are numerous operations that can produce the next hierarchical level. Examples include abstraction and generalization operate on lower-level objects to create higher-level ones; compilation or aggregation can create higher-level units; steps are combined together to create processes; going "meta" ("thinking about thinking"); and moving from static to dynamic systems or linear dependency to mutual dependency also involve hierarchical transformations. Kegan notes that increasing complexity and sophistication moves (vertically) from entities to processes, from static to dynamic systems and from dichotomous to dialectical relationships (Kegan, 1994, p. 13).

Horizontal growth also follows a pattern in natural systems including human learning. The sequence is from single objects, to multiple independent objects, to multiple interacting objects, to massively interconnected object, and finally to an emergent whole that transitions to the next hierarchical level. It makes intuitive sense that it is easy to learn a few more words (horizontal), but the leap to speaking sentences is comparatively momentous (which is not to say that it comes online all of a sudden, i.e., children produce quasi-sentences first). Furthermore, this difference is quantitative. If we wanted to measure language complexity, we can count the size of vocabulary and the length of words, but no amount of increase in vocabulary will "equal" the shift from words to sentences.

Hierarchical complexity (which is Commons' term; other developmentalists use different terms) contributes to our analysis of authoring tool complexity in several ways. First, it ameliorates the "dimensionality issue" by providing another tool for organizing the plethora of complexity dimensions, i.e., according to horizontal and vertical differences in complexity, toward our goal of coordinating the complexities of tool vs. task vs. user and in our goal of comparing two (or more) tools (or tasks, or types of users). Second, because it is primarily a learning or developmental theory, it provides important insights into the effort and prerequisite knowledge a new user needs to use an authoring tool. Vertical growth is typically more difficult than horizontal growth, and the emergence of a new level of organization often comes with some disequilibrium or dissonance, which, in turn, means there can be resistance or hesitancy.

Now, we can begin with a rough characterization of the level of software tool complexity that a hypothetical user already has, and then ask whether the features and tasks of an authoring tool represent horizontal or vertical types of learning for the skill acquisition learning curve. We must not assume that new user skill level can be increased in any sort amount of time with something like a training intervention if vertical learning is involved.

Hierarchical Complexity and Epistemic Forms

The analysis of tool/task/user complexity can proceed in basically two directions: more rigorous qualitative analysis and more heuristic quantitative analysis (though any analyses will probably combine qualitative and quantitative methods). For my purposes, I focus on heuristic estimations. My goal is to either start with a particular authoring tool/task and identify the communities of practice and training needs that will match the tool/task; or, starting with a target user group, design the tool/task to match the estimated complexity of a community of practice. One can use the concepts introduced in this chapter, including the dimensions of complexity, types of epistemic forms, and the distinction between horizontal and vertical differences in complexity, to make subjective shoot-from-the-hip assessments and inform design discussions as is usually done in software design. Alternatively, and left for others to carry forward, one can use these concepts to construct detailed quantitative metrics and formulas for calculating task/tool/knowledge complexity—but such is not necessary to make solid progress in matching tools/tasks to users.

Morrison and Collins (1994) mention the “epistemic complexity” of epistemic forms and games, but they do not define it precisely. What I contribute here is an attempt to link epistemic games to cognitive developmental theory in an attempt to create a grounded framework for assessing the relative complexity of epistemic forms/games, which then provides a framework for describing the complexity of authoring tool features. These epistemic forms can be sequenced according to complexity level modeled on the levels mentioned in hierarchical complexity theory, as shown in Table 3.

Table 3 Epistemic forms organized by complexity level

Epistemic Form for Tool/Task/Mental Model	Complexity Level
• Text information fill-in boxes	Simple objects
• Lists, choices, sliders, and check boxes	
• Forms, schemas, or templates	Abstractions and mappings
• Tables and matrices	
• Hierarchies and trees	
• Scripts (with branches)	Formal systems
• Equations and Boolean logic	
• Structural models: concept networks, boxology diagrams	
• Causal and constraint models (and using variables)	Dynamic systems
• Behavioral/procedural models: If/then and rule-based procedural representations	
• (Authoring of) decision trees, Bayesian nets, etc.	
• Coordination of dynamic modules, e.g., complex interactions between expert, student, teaching modules, and dynamic use scenarios.	Architectures and ecosystems (systems of dynamic systems)
• Design that takes into account emergent and chaotic interactions.	

As a final step, in Figure 5, I link these complexity levels to the low/medium/high level of complexity associated with different categories of users from Table 2. Again, this mapping is a heuristic estimation that is intended to illustrate the type of analysis; no strong claims are made for the specific mappings.

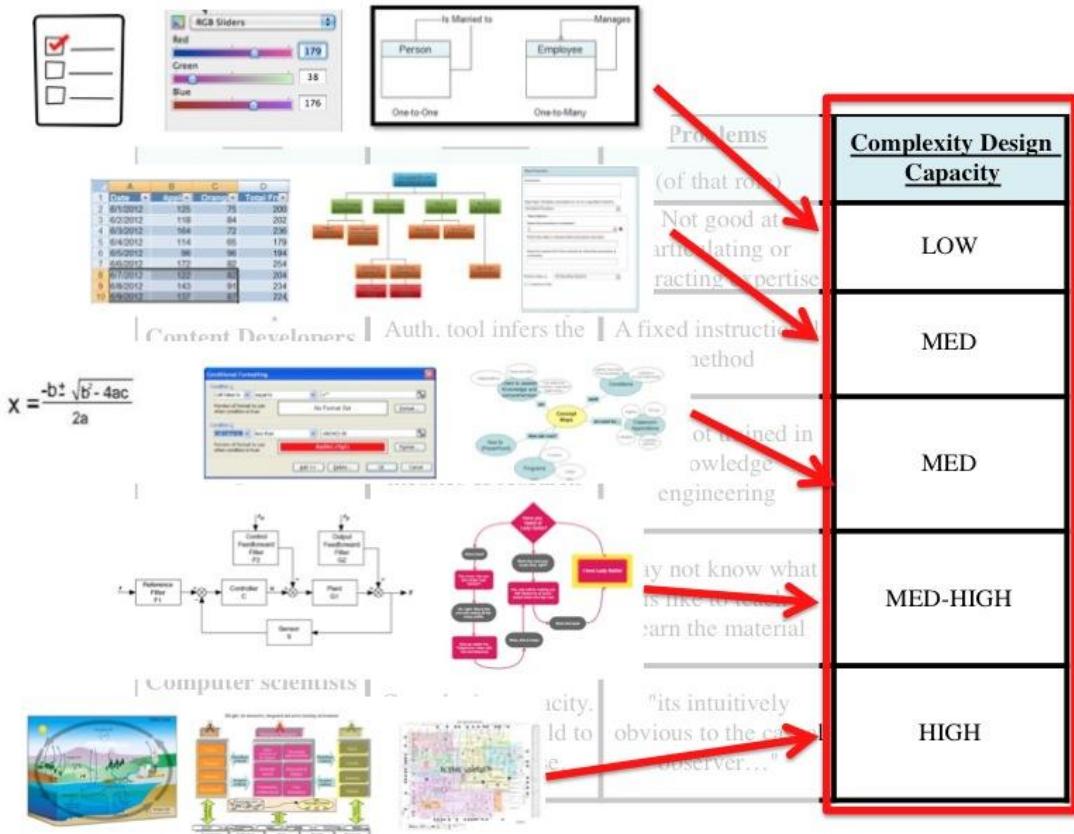


Figure 5: Complexity levels of epistemic forms

Discussion

Beginning with a summary of my article on ITS authoring tool design, I described some of the challenges facing authoring tool designers and researchers today. Consonant with this Special Issue's theme of personal retrospectives on classic papers, I also included a narrative look at what brought me to authoring tools work and mentioned that my academic journey since then has included interdisciplinary tributaries outside of ITS and educational technology per se. The invitation to write this chapter has given me the happy opportunity to apply new frames of reference to an old topic. The reader hoping for definitive answers to questions about software complexity may have been disappointed—what I have done is exploratory theorizing to help frame important questions by suggesting certain theories, principles, and concepts amenable to ITS authoring tool R&D.

In this chapter, I have explored some theoretical bases for assessing the appropriateness of ITS authoring tools, and *any* type of software artifact, to intended user communities. The analysis is based on general notions of complexity from complexity science and hierarchical complexity theory. The importance of considering tools, tasks, user capacity, and community of practice in an integrated way was supported through the inclusion of the models of activity theory and epistemic forms.

Matching tool/task complexity to user/community complexity capacity is important because authoring tools are complex and expensive to build, and, using a “risk analysis” framework, we can say that the more expensive a system is to build, the larger the risk if user needs and capacities are not understood and anticipated. The design goal is to find the sweet spot where risk is acceptably low and expected value is

relatively high. Oja's (2010) study of improving usability in complex software systems concludes that systems should anticipate that projects usually involve a variety of roles and areas of expertise, and that interfaces should allow for the "distribution of tasks according to participant strengths" (p. 3800). Thus, the goal is not so much to match the affordances of an authoring tool to an intended user type, but anticipate the range of user types involved in an ITS design and build tools that clearly meet the needs of each design role. Also, and plans for large scale adoption of authoring tools should include plans for learning and peer-mentoring within specific communication pathways in communities of learning.

The inclusion of complexity science and theories of dynamic systems in our narrative supports a bigger picture consideration of authoring that considers not only how tools should be built to match user capacities, but the reciprocal evolution of tools and human capacities over longer periods of time. As Jerome Bruner notes "through using tools, man changes himself and his culture...human evolution is altered by man-made tools" (1987). Thus, tools can not only support the construction of advanced learning systems, but might also be designed to help users (especially instructors) more deeply understand and incorporate leading-edge learning theories and mental models of the learning process (or build more adequate mental models of their content domain). We can move beyond seeing authoring tools primarily in terms of time and effort savings and consider their role in *empowering* content and pedagogy experts, including teacher, and in terms of propelling the evolution of computer-mediated learning in general.

References

- Abelson, H. & Sussman, G. J. (1983). *Structure and interpretation of computer programs*. MIT Press, Cambridge, MA.
- Ainsworth, S., Major, N., Grimshaw, S., Hayes, M., Underwood, J., Williams, B. & Wood, D. (2003). REDEEM: Simple Intelligent Tutoring Systems from Usable Tools. Chapter 8 in Murray, T., Blessing, S. & Ainsworth, S. (Eds.). *Authoring Tools for Advanced Technology Learning Environments*. Springer: Netherlands.
- Aleven, V. & Sewall, J. (2010, June). Hands-on introduction to creating intelligent tutoring systems without programming using the cognitive tutor authoring tools (CTAT). In *Proceedings of the 9th International Conference of the Learning Sciences-Volume 2* (pp. 511-512). International Society of the Learning Sciences.
- Aleven, V., McLaren, B. M., Sewall, J. & Koedinger, K. R. (2006). The cognitive tutor authoring tools (CTAT): Preliminary evaluation of efficiency gains. In *Intelligent Tutoring Systems* (pp. 61-70). Springer Berlin Heidelberg.
- Benbya, H. & McKelvey, B. (2006). Toward a complexity theory of information systems development. *Information Technology & People*, 19(1), 12-34.
- Bereiter, C. & Scardamalia, M. (2006). Education for the knowledge age: Design-centered models of teaching and instruction. In P. A. Alexander & P. H. Winne (Eds.), *Handbook of educational psychology* (2nd ed., pp. 695-713). Mahwah, NJ: Lawrence Erlbaum Associates.
- Brown, A. L. & Campione, J. C. (1996). Psychological theory and design of innovative learning environments: On procedures, principles, and systems. In L. Schauble & R. Glaser (Eds.), *Innovations in learning: New environments for education* (pp. 289-325). Mahwah, NJ: Lawrence Erlbaum Associates.
- Bruner, J. (1987/2004), 'Life as Narrative', *Social Research*, 71: 691-710.
- Byström, K. & Järvelin, K. (1995). Task complexity affects information seeking and use. *Information processing & management*, 31(2), 191-213.
- Campbell, D. J. (1988). Task complexity: A review and analysis. *Academy of management review*, 13(1), 40-52.
- Chilana, P. K., Wobbrock, J. O. & Ko, A. J. (2010, April). Understanding usability practices in complex domains. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 2337-2346). ACM.
- Cobb, P., Confrey, J., diSessa, A. Lehrer R. Schauble, L. (2003). Design experiments in educational research. *Educational Researcher*, Jan/Feb 2003; 32, 1.
- Collins, A. & Ferguson, W. (1993). Epistemic forms and epistemic games: Structures and strategies to guide inquiry. *Educational Psychologist*, 28(1), 25-42.

- Commons, M. L. & Richards, F. A. (1984). A general model of stage theory. In M. L. Commons, F. A. Richards & C. Armon (Eds.), *Beyond formal operations: Late adolescent and adult cognitive development*, (pp. 120-141). New York: Praeger.
- Commons, M. L. & Pekker, A. (2009, unpublished). Hierarchical complexity and task difficulty.
<http://dareassociation.org/papers.php>. Accessed Monday, November 30, 2009.
- Commons, M. L. & Pekker, A. (2008). Presenting the formal theory of hierarchical complexity. *World Futures: Journal of General Evolution* 64(5-7), 375-382.
- Commons, M. L., Trudeau, E. J., Stein, S. A., Richards, F. A. & Krause, S. R. (1998). Hierarchical complexity of tasks shows the existence of developmental stages. *Developmental Review*, 18, 238-278.
- Conklin, J. (2005). Wicked Problems & Social Complexity. Chapter 1 of *Dialogue Mapping: Building Shared Understanding of Wicked Problems*, Wiley.
- Constantin, A., Pain, H. & Waller, A. (2013). Informing the Design of an Authoring Tool for Developing Social Stories. In *Human-Computer Interaction-INTERACT 2013* (pp. 546-553). Springer Berlin Heidelberg.
- Cook-Greuter, S. R. (2000). Mature ego development: A gateway to ego transcendence. *J. of Adult Development*, 7(4), 227-240.
- Cook-Greuter, S.R. (2005). Ego Development: Nine levels of increasing embrace. Available at www.cook-greuter.com.
- Cristea, A. (2005). Authoring of Adaptive Hypermedia. *Journal of Educational Technology & Society*, 8(3).
- Engestrom, Y., Miettinen, R. & Punamaki, R.-L. (Eds.). (1999). *Perspectives on activity theory*. New York: Cambridge University Press.
- Fischer, K. (1980). A theory of cognitive development: The control and construction of hierarchies of skills. *Psychological Review*, 87(6), 477-531.
- Fischer, K. & Yan, Z. (2002). The development of dynamic skill theory. In *Conceptions of development: Lessons from the laboratory*, 279-312.
- Gentner, D. & Stevens, A. (Eds.). (1983). *Mental models*. Hillsdale, NJ: Lawrence Erlbaum Assoc.
- Gersh, J. R., McKneely, J. A. & Remington, R. W. Cognitive Engineering: Understanding Human Interaction with Complex Systems. *Johns Hopkins APL Technical Digest*, 26, 4 (2005), 377-382.
- Graesser, A.C., Chipman, P., Haynes, B.C. & Olney, A. (2005) AutoTutor: An intelligent tutoring system with mixed-initiative dialogue. *IEEE Transactions in Education*, 48, 612-618
- Grünwald, P. D. & Vitányi, P. M. (2003). Kolmogorov complexity and information theory. With an interpretation in terms of questions and answers. *Journal of Logic, Language and Information*, 12(4), 497-529.
- Haynes, S. R. & Kannampallil, T. G. (2004). Learning, Performance, and Analysis Support for Complex Software Applications. *Proc. of the 3rd Ann. Workshop on HCI Research in MIS*, 30-34.
- Heffernan, N. & Heffernan, C. (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.
- Johnson, W. L. & Valente, A. (2008, July). Tactical Language and Culture Training Systems: Using Artificial Intelligence to Teach Foreign Languages and Cultures. In AAAI (pp. 1632-1639).
- Johnson-Laird, P.N. (1983). *Mental models: Towards a cognitive science of language, Inference, and consciousness*. Cambridge, MA: Harvard University Press.
- Johnson, C. W. (2006). Why did that happen? Exploring the proliferation of barely usable software in healthcare systems. *Quality and Safety in Health Care*, 15, i76-i81.
- Jonassen, D. & Rohrer-Murphy, L. (1999). Activity theory as a framework for designing constructivist learning environments. *Educational Technology, Research & Development*, 47 (1), 61-79.
- Jordan, T., Andersson P. & Ringnér, H. (2013). The Spectrum of Responses to Complex Societal Issues: Reflections on Seven Years of Empirical Inquiry. *Integral Review*, February 2013, Vol. 9, No. 1.
- Kegan, R. (1982). *The Evolving Self*. Harvard University Press.
- Kegan, R. (1994). *In over our heads: The mental demands of modern life*. Cambridge, MA: Harvard University Press.
- Koedinger, K. R., Anderson, J. R., Hadley, W. H. & Mark, M. A. (1997). Intelligent tutoring goes to school in the big city. *International Journal of Artificial Intelligence in Education (IJAIED)*, 8, 30-43.
- Kumar, P., Samaddar, S. G., Samaddar, A. B. & Misra, A. K. (2010, June). Extending IEEE LTSA e-Learning framework in secured SOA environment. In *Education Technology and Computer (ICETC), 2010 2nd International Conference* (Vol. 2, pp. V2-136). IEEE.
- Mirel, B. (2004). *Interaction Design for Complex Problem Solving*. San Francisco, CA: Morgan Kaufman.

- Mitrovic, A. (2012). Fifteen years of Constraint-Based Tutors: What we have achieved and where we are going. *User Modeling and User-Adapted Interaction*, vol. 22(1-2), 39-72, 2012.
- Mitrovic, A., Martin, B., Suraweera, P., Zakharov, K., Milik, N., Holland, J., McGuigan, N. (2009). ASPIRE: an authoring system and deployment environment for constraint-based tutors. *Artificial Intelligence in Education*, vol. 19(2), 155-188, 2009.
- Mizoguchi, R. and Murray, T. (Eds.) (1999); Proceedings of “Ontologies for Intelligent Educational Systems,” Workshop at AIED-99, LeMans France, July 1999.
- Morrison, D. & Collins, A. (1995). Epistemic Fluency and Constructivist Learning Environments. *Educational Technology*, 35(5), 39-45.
- Murray, T. & Woolf, B. (1992). Tools for Teacher Participation in ITS Design. In Frasson, Gauthier & McCalla (Eds.) *Intelligent Tutoring Systems, Second Int. Conf.*, Springer Verlag, New York, pp. 593-600.
- Murray, T. (1996, May). Having It All, Maybe: Design Tradeoffs in ITS Authoring Tools. In *Intelligent Tutoring Systems: Third International Conference, ITS '96*, Montreal, Canada, June 12-14, 1996. Proceedings (Vol. 1086, p. 93). Springer.
- Murray, T. (1999). Authoring Intelligent Tutoring Systems: Analysis of the state of the art. *Int. J. of AI and Education*. Vol. 10 No. 1, pp. 98-129.
- Murray, T. (2003). An Overview of Intelligent Tutoring System Authoring Tools: Updated analysis of the state of the art. In *Authoring tools for advanced technology learning environments* (pp. 491-544). Springer: Netherlands.
- Murray, T., Blessing, S. & Ainsworth, S. (Eds) (2003). *Authoring Tools for Advanced Technology Learning Environments: Toward cost-effective adaptive, interactive, and intelligent educational software*. Springer: Netherlands.
- Murray, T. (2004). Design Tradeoffs in Usability and Power for Advanced Educational Software Authoring Tools. *Educational Technology Journal*, Sept-Oct 2004, pp. 10-16.
- Murray, T. (2015). Coordinating the Complexity of Tools, Tasks, and Users: Toward a Theory-based Approach to “Authoring Tool Design,” to appear in the International Journal of Artificial Intelligence and Education, Vol. 25.
- Nielsen, J. (1994, April). Enhancing the explanatory power of usability heuristics. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 152-158). ACM.
- Nielsen, J. *Usability Engineering*. Boston, MA: AP Professional (1993).
- Norman, D. (1988). *The Design of Everyday Things*. Doubleday: NY.
- Oja, M. K. (2010). Designing for collaboration: improving usability of complex software systems. In *CHI'10 Extended Abstracts on Human Factors in Computing Systems* (pp. 3799-3804). ACM: Chicago.
- Olsen, J. K., Belenky, D. M., Aleven, V. & Rummel, N. (2013, January). Intelligent Tutoring Systems for Collaborative Learning: Enhancements to Authoring Tools. In *Artificial Intelligence in Education* (pp. 900-903). Springer Berlin Heidelberg.
- Piaget, J. (1972). *The principles of genetic epistemology*. Basic Books, NY.
- Ritter, S. & Blessing, S. (1998). Authoring tools for component-based learning environments. *Journal of the Learning Sciences*, 7(1) pp. 107-132.
- Shaffer, D. W. (2006). Epistemic frames for epistemic games. *Computers & Education*, 46(3), 223-234.
- Sitaram, S. & Mostow, J. (2012, May 23-25). Mining Data from Project LISTEN’s Reading Tutor to Analyze Development of Children’s Oral Reading Prosody. In Proceedings of the 25th Florida Artificial Intelligence Research Society Conference (FLAIRS-25), 478-483. Marco Island, Florida.
- Soloway, E., Guzdial, M. & Hay, K. E. Learner-centered design: the challenge for HCI in the 21st century. *Interactions*, 1, 2 (1994), 36-48.
- Sottilare, R. A., Brawner, K. W., Goldberg, B. S. & Holden, H. K. (2012). The generalized intelligent framework for tutoring (GIFT). *Orlando, FL: US Army Research Laboratory-Human Research & Engineering Directorate (ARL-HRED)*.
- Sottilare, R., Graesser, A., Hu, X. & Goldberg, B. (2014). *Design Recommendations for Intelligent Tutoring Systems: Volume 2: Instructional Management*. U.S. Army Research Laboratory Human Research & Engineering Directorate.
- Specht, M. (2012). E-Learning Authoring Tools. In *Encyclopedia of the Sciences of Learning* (pp. 1111-1113). Springer US.
- Stahl, G. (2006). *Group Cognition: Computer Support for Building Collaborative Knowledge*. Cambridge, MA: MIT Press.

- Suraweera, P., Mitrovic, A. & Martin, B. (2010). Widening the knowledge acquisition bottleneck for constraint-based tutors. *International Journal of Artificial Intelligence in Education*, 20(2), 137-173.
- 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.
- Woolf, B. & McDonald, D. (1984). Design issues in building a computer tutor. *IEEE Computer*, Sept. 1984.
- Woolf, B. P. (2010). *Building intelligent interactive tutors: Student-centered strategies for revolutionizing e-learning*. Morgan Kaufmann.
- Zhang, J., Scardamalia, M., Reeve, R. & Messina, R. (2009). Designs for collective cognitive responsibility in knowledge-building communities. *The Journal of the Learning Sciences*, 18(1), 7-44.

CHAPTER 3 One-Size-Fits-Some: ITS Genres and What They (Should) Tell Us About Authoring Tools

Benjamin Bell

Aqru Research and Technology, LLC

Introduction

The process of creating sophisticated Intelligent Tutoring System (ITS) can be costly, complex, and tedious, and relies on collaborative expertise from multiple disciplines. Authoring tools streamline and accelerate the construction of ITS by providing a framework within which an author can design a learning system. Some authoring systems are general-purpose tools that provide an author with a great deal of leeway. Others embody a set of assumptions about what the authored product will look like and how it will behave. However, the authoring tool ecosystem has evolved with little discussion of ITS genres and the desired properties of tools supporting authoring within each genre. Instead, authors of instructional software often determine *a priori* what the authoring tool(s) will be and then commence the design process informed by a combination of past experience, online research, discussion with colleagues, and product availability.

I hypothesize that authors seldom think about the genre of the learning system they wish to create and even more seldom use that genre as a filter in selecting the appropriate tools. Moreover, even the author who engages in this deliberative process is unlikely to find authoring tools that are explicitly aligned with specific genres of ITS.

In this chapter, I discuss the characteristics of ITSs that can be used to derive a set of genres, and the relationships between those characteristics and desired properties of ITS tools corresponding to each. I use examples of authoring tools to contrast general-purpose and specialized tools, and illustrate the utility of aligning authoring tools to corresponding genres.

Related Research

This chapter discusses various genres of ITSs and what they have to say about ITS authoring tools. The purpose is not to propose an exhaustive ontology of tutoring systems, but highlight how fundamental properties of tutorial interactions and simulations influence thinking about authoring tools.

Why ITS Categories Matter

Numerous ontologies have been proposed for characterizing ITS. Since this chapter explores the influence of ITS genres on authoring tools, the properties relevant to this discussion are not those focusing on the user experience so much as those governing the design and construction of an ITS (though these are often related). I can go further and suggest that instructional strategies similarly do not define what genre an ITS should be identified with, so much as how they are built (though these too are related). Put another way, the relevant distinguishing characteristics of an ITS are related to the questions “how do I build one?” and “what’s hard about that?”.¹

¹ Murray (2003) proposes as a fundamental question “who should author ITS?” which is an important question but less relevant to characterizing ITS categories.

This is not a radical departure from traditional ITS research by any means. A view of ITSs that has endured for four decades and remains influential today identifies the three elements of an ITS as (1) the *expert model* (domain knowledge), (2) the *student model* (knowledge about the user), and (3) the *tutor* (knowledge of teaching strategies) (Hartley & Sleeman, 1973). This decomposition factors in neither user experience nor instructional strategies, but is essentially an architectural blueprint. Researchers thus generally converged around the general notion that building an ITS was a process of creating, more or less independently, expert models, student models, and tutoring strategies (Burns & Capps, 1988).

Debate about specific approaches generally focused within each of these three components. A good deal of research has resulted in an array of theoretical frameworks that remain vigorously investigated to this day. Reviews of expert modeling appear in Ahuja & Sille (2013); Paviotti, Rossi & Zarka (2012); and Sani & Aris, (2014). For a review of student modeling, see Pavlik, Brawner, Olney & Mitrovic (2013). A review of tutoring strategies is presented in Sotilare, DeFalco & Connor (2014).

An extension to the canonical ITS model has acknowledged the interface between the user and tutor as an integral component (Miller, 1998; Sotilare, 2012). *Interface* as used here refers to how the user and tutor interact, not simply to how the display appears. Miller (1998) distinguished two principal metaphors for this interaction: *first-person* interfaces, where the user directly manipulates displayed objects; and *second-person* interfaces, which allow the user to command actions. First-person interfaces can provide the user with a feeling of working directly with the domain. This interface metaphor is a natural way for a user to engage in a simulation because changes in the system, process, environment, or device being simulated can be effected in a manner that resembles the physical world. That this type of interaction raises questions about authoring tools should be clear (I discuss this later).

In a second-person interface, a user commands actions to an implicitly or explicitly represented agent. Agency is thus delegated to the system through what can be an abstraction (e.g., a menu), an embodiment of a person (e.g., a depiction of a tutor), or some other representation of a non-human but still interactive entity (e.g., a helpful paper clip). The modality of this interaction can vary. Basic interface controls provide a (usually) graphically oriented palette of user commands (such as “skip,” “go back,” or “help”). These commands are distinctive from controls embedded within the simulated environment (a steering wheel, a syringe, etc.). Menus are another common means to embody an abstraction of an agent, and have evolved to be highly context-dependent.

The basic elements governing the interaction between the user and tutor thus appear to have become established in the canonical ITS (Sotilare, 2012). However, answering the questions “how do I build one?” and “what’s hard about that?” has become less straightforward as ITSs have grown less homogeneous. This heterogeneity among ITSs matters, in part, because of the implications for authoring tools. In the next section, I briefly discuss two genres, linked with first- and second-person metaphors, respectively, and a third that borrows from both traditions. These are representative of the last few decades of ITS research that have been influential in the technology-mediated learning community. For purposes of discussion, I label the first two *simulation-based* learning and *discourse-based* learning. The third genre, which adopts elements of both first-person and second-person interfaces, is labeled *situation-based* learning. Although the terms “simulation” and “situation” are related, I draw an important pragmatic distinction between a computational simulation (of a device, process, system or environment) and a collection of situations that a learner could encounter through taking actions or asking questions—where each circumstance itself is static but where the overall user experience could feel dynamic.² The

²For instance, a finite state machine could occupy both simulation and situation paradigms, but since a state has inspectable, static properties, for our purposes such an architecture fits more within the situation-based learning genre.

discussions that follow are summary in nature; the reader is referred to more comprehensive reviews cited in each section.

Simulation-Based Learning

The emergence of desktop simulation and its rapid trajectories toward greater fidelity and lower cost have created rich opportunities for automated learning while raising fundamental questions for the ITS community. The general construct of a simulated world is captured in the term *reactive environment* (Shute & Psotka, 1996) to describe an ITS in which “the system responds to learners’ actions in a variety of ways that extend understanding and help change entrenched belief structures using examples that challenge the learner’s current hypotheses.” (p. 579). As a result of much research into the teaching potential of simulations, the canonical ITS has expanded to include a simulated environment. Researchers have used various labels to describe such components (and the encapsulating tutoring system), including *environment module* (Burton, 1988), *microworld* (Frederiksen & White, 2002), *simulation-centered tutor* (Munro, et al., 1997), or *discovery learning environment* (Veermans, van Joolingen & de Jong, 2000).

Although a simulator is not in itself a tutoring system, there has been significant progress in the use of desktop simulation to advance learning objectives in ITSs, particularly those employing games. The pervasive presence of this approach is reflected in the literature, which discusses, alternately, embedding a simulation within a tutor (Jong & van Joolingen, 1998; Towne & Munro, 1988) and embedding a tutor within a simulation (Rickel & Johnson, 1999; Fowler, Smith & Litteral, 2005; Wray, Woods & Priest, 2012; Bell, Johnston, Freeman & Rody, 2004; Bell, Jarmasz & Nelson, 2011). Since this distinction is largely an implementation question and not a theoretical one, the term “intelligent game-based learning environment” is often used to refer to a pairing of simulation and tutoring capabilities, irrespective of system architecture (Lester, Lobene, Mott & Rowe, 2014).

Discourse-Based Learning

The prevalent metaphor for driving second-person interfaces is discourse. Improved capabilities for natural language interaction have enabled more conversational forms of this kind of interaction. Also, remarkable gains in speech recognition have yielded second-person interfaces that support spoken discourse between a user and the agent that is interpreting and carrying out the user’s instructions. In this regard, technology has caught up with the visions of earlier ITS researchers, exemplified by Miller’s observation that “the image of an interface as a ‘second person’ agent working for the user is perhaps most clearly captured by a natural language interface” (1998, p. 155).

Discourse as a tutorial strategy is intended to operate in an ITS much like it does when practiced by a skilled human tutor (Van Lehn, 2011). Using discourse as a tutoring technique is distinct from using discourse to train the skills related to engaging in discourse (e.g., in language training, see Johnson & Valente, 2008). In discourse-based learning, the tutor uses conversation and its varied constructs (questions, answers, reflection, rhetoric) to elicit thought, reasoning, problem solving, and question-posing from the student.

This chapter does not survey the literature on dialogue-based tutors though recent reviews appear in Brawner & Graesser (2014) and Rus, D’Mello, Hu & Graesser (2013). Instead, I use as an exemplar an influential and representative body of research in dialogue-based tutors led by Graesser and colleagues called AutoTutor and its variants (Graesser, et al., 2004; Graesser, Chipman, Haynes & Olney, 2005; D’Mello & Graesser, 2012). AutoTutor embodies a theory of dialogue-based instruction based on authentic (human) tutoring behaviors. The theory has evolved from proposing numerous dialogue moves (e.g., question, prompt, correct, hint) (Graesser, et al., 1999; Graesser, et al., 2001) to proposing an

integrative dialogue model called *Expectation and Misconception Tailored* (EMT) (Graesser, et al., 2012). AutoTutor thus offers a useful example for my discussion later of how authoring tools address discourse-based ITS.

Situation-Based Learning

The third example of an ITS genre fits neither wholly within first-person interfaces nor wholly within second-person interfaces. Situation-based learning though has great contemporary importance and addresses a conceptual flaw in traditional ITS models that did not call for any sort of authentic context—the requirements for a user model, domain model, and tutoring strategies (and later, an interface) did not implicate a need for setting instruction against a backdrop relevant to the target skills and knowledge. Learning sciences researchers though recognized that instructional systems could be more effective when coupled with circumstances in which the user naturally encounter, learn, and apply the skills and knowledge being taught.

Collins, Brown & Newman (1989) describe natural alignment of how people learn with the use of an authentic context in which to embed learning. They use the term *cognitive apprenticeship* to describe the application of traditional apprenticeship learning to class instruction, which they argue is especially relevant to learning higher-order metacognitive skills and problem-solving strategies as employed by expert practitioners. *Situated learning theory* (Brown, Collins & Duguid, 1989) asserts that learning in context is more consistent with how people acquire knowledge and skills as supported by research in education and cognitive science. The authors “argue that approaches such as cognitive apprenticeship that embed learning in activity and make deliberate use of the social and physical context are more in line with the understanding of learning and cognition that is emerging from research” (Brown, Collins & Duguid, 1989, p. 32). Bransford and colleagues (1990) present a framework for *anchored instruction* that makes the role of an authentic context explicit by structuring learning through realistic, complex problems embedded within a narrative. Another body of research influenced by these contextual approaches yielded a long series of ITS conforming to a framework called goal-based scenarios (GBS) (Schank, Fano, Bell & Jona, 1994).

Although these theories differ in surface features, they share the essential principles of goal-driven inquiry in pursuit of authentic, complex and ill-defined problems (Bell & Zirkel, 2001), embedded within a fictional narrative context (Riedl & Young, 2014). The shared emphasis on creating an authentic context for learning, and for embedding instruction within a suitable culture of practice, has implications for ITS authoring as discussed later.

Implications for ITS Authoring Tools

In the previous section I presented three representative ITS genres that have each emerged from, and altered the canonical ITS model. This section considers the authoring process and its challenges in the more contemporary context of ITS genres as they have evolved in recent research. In his analysis of ITS authoring tools, Murray (1999) proposed distinguishing those that are pedagogy-oriented (supporting the sequencing and teaching of generally static content) from those that are performance-oriented (enabling interactive environments with opportunities to learn and apply skills and get feedback). Murray (2003) identified four categories of pedagogy-oriented ITS authoring tools: curriculum sequencing and planning, tutoring strategies, multiple knowledge types, and intelligent/adaptive hypermedia; and three specific categories of performance-oriented ITS authoring tools: device simulation and equipment training, domain expert system, and special purpose.

The three categories mentioned previously do not neatly align with Murray’s categories, but are useful for contextualizing the present discussion. Simulation-, discourse-, and situation-based learning, while not

intended as an elaborated ontology, are used here to organize a brief consideration of how ITS tools can best support the authoring process along with a few select exemplars.

Intelligent Tutoring Demands Intelligent Authoring

The act of constructing an ITS has been viewed largely as the assembly and integration of disparate but interacting components, where “traditional intelligent tutoring systems (ITSs) are typically constructed out of four primary components or modules: the user interface, expert model, student model, and instructional module” (Jona & Kass, 1997, p. 39), and ITS authoring tools have evolved largely along these lines (Macmillan, Emme & Berkowitz, 1988; Murray, 2003; Murray & Woolf, 1992; Russell, Moran & Jordan, 1988; Brawner, Holden, Goldberg & Sottilare, 2012). To support construction of each of these modules, authoring tools came to “consist of specialized editors for building each component (i.e., a user interface builder, expert model editor, etc.)” (Jona & Kass, 1997, p. 39).

More recently, work being done under the Generalized Intelligent Framework for Tutoring (GIFT) initiative has called for authoring support of five functions: user models, domain knowledge, instructional strategies, user-tutor interfaces, and integrating tutor components (Sottilare, 2012). In calling attention to integrating components, GIFT researchers explicitly acknowledge the importance of and the challenges surrounding the integration of ITS components.

GIFT and its precursors are thus generally aligned with a software engineering approach to supporting the process of creating complex ITS components. However it is also important to consider the underlying learning principles that are implicitly adopted or explicitly enforced by an authoring tool architecture and to examine how an ITS authoring tool provides support to an author in properly adhering to those learning principles.

In previous work, we suggested that authoring tools that observe this component-oriented approach “are too general to serve as a specification for a piece of educational software, and are too general to be of much help in guiding a designer in creating such software” (Bell, 2003, p. 349). Lack of specificity though could have implications for ITS tools beyond just limiting their utility. Kass & Jona (1997) argue that “while the idea of modular, interchangeable components sounds quite appealing from a software engineering perspective, we’re skeptical about the educational validity of this idea, and of the implicit model of learning which underlies it” (p. 39). In other words, authoring tools premised solely on a software engineering approach may lack a theoretical basis for guiding the creation of effective instruction.

An alternative is to think about ITS tools as a general label for the space of discrete, specific, and standalone authoring environments, each conforming to a different teaching architecture (instructional approach). One benefit to this approach is that there is arguably a wide range of categories of ITSs, so this approach avoids the problem of how to create a truly supportive and sound “universal” tool. Another benefit is that the enormous range of potential actions and interactions, which a universal tool would need to support the authoring of, would require vast representational knowledge capturing the structure of what users do when engaging with ITS. With tools that are specific to an instructional architecture, the representational challenges become far more tractable. Third, an ITS tool specific to an instructional architecture is in principle more capable of providing informed authoring guidance than a tool that, by necessity, could offer guidance in only very general terms. An ITS authoring tool built with a specific instructional approach in mind can thus avoid sacrificing value as an intelligent guide in the name of generality.

This sort of approach does not come without some cost: research, analysis, and validation is required in order to derive teaching architectures that are viable (meaning, ITS tools could effectively support authoring) and effective (meaning, ITS created from such tools could have instructional utility). There is therefore a dual process that “entails creating a fully designed and implemented teaching architecture along with a special-purpose tool for instantiating that architecture in a variety of domains” (Jona & Kass, 1997, p. 39).

In the next sections, I revisit the three categories introduced previously (learning driven by simulations, by discourse, and by situations) and discuss implications for authoring tools that embody the notion of category-tailored ITS authoring.

Implications of Simulation-Based Learning for ITS Authoring

The first-person interface metaphor was introduced previously as the basis for a great deal of productive research that has explored the instructional potential of simulations as powerful environments for tutoring systems. The process of authoring an ITS in this category is unlikely to conform very closely with the general model of ITS authoring, so it is also unlikely that a general-purpose tool is the ideal authoring environment. One challenge faced by the ITS author is how to structure event sequences and transitions to achieve the desired learning outcomes. Open-word simulations do not ensure that learning objectives are achieved (or even encountered); a helpful tool should coach the ITS author in exercising some measure of controls expressed in frameworks such as Guided Experiential Learning (Clark, Yates, Early & Moulton, 2010). So here we see a need for ITS authoring tools to “understand” simulation in a way that diverges from traditional authoring.

For instance, in a simulation-driven ITS the nature of the student model may not be along the traditional lines of a separate, explicit module. Student models are useful for recognizing what a user’s goals are in selecting a course of action (Greer & Koehn, 1995; Whitaker & Bonnell, 1990) and identifying typical error modes a user may be displaying (Burton & Brown, 1978; Brown & VanLehn, 1988; Brusilovsky, 1994). However, exploratory environments enabled by simulation can reduce the reliance on a student model, if not eliminate the need entirely. Student models emerged as a means to interpret and track student actions and intentions. A simulation though can be seen as a dynamic record of a user’s path, since the state of the simulated world at any moment in time is attributable to the user’s intentions and how the user effected change in the world in service of those intentions. The environment thus can reveal how far toward some defined objective the user has progressed and what the user has done correctly or erroneously (Livak, Heffernan& Moyer, 2004). This is a simplification but expresses the basic argument that I can elaborate with a brief example.

Consider a flight simulator embedded within an ITS designed to train Air Force student pilots in proper radio communications procedures. A student model could be developed that cues a tutor to recognize what goals a user is pursuing (reducing power, extending flaps, and lowering the gear signals a goal to land) and what behaviors might be attributable to a common type of error (e.g., failing to report gear down at a required position in the pattern). However, the simulation, as a realistic, doctrinally correct model, *knows* (in some sense) what reducing power, deploying flaps, and lowering the gear signals in terms of intent; it also knows that failing to communicate at a mandatory reporting point is an error. The simulation would therefore be able to cue the tutor to derive an appropriate intervention, such as commanding the synthetic instructor pilot to tell the user, “you need to make a gear-down call.” One is

left to conclude that either there is no student model in this instance or that the student model is not a separate module but is embedded in the simulated world (the environment and the agents that occupy it).³

Either way, one consequence of an ITS that provides a rich set of affordances through which a learner explores and influences the environment is that the construct of a separate student model becomes less relevant. It can also be problematic, since an “open world” simulation of any reasonably complex phenomenology greatly complicates encapsulating all possible solution paths in a model (Derry & Lajoie, 1993). Derry and Lajoie (1993) present five additional factors that cast doubt on the learner modeling paradigm:

- (1) learner error patterns, or bugs, cannot be fully pre-determined;
- (2) the presentation of static content and feedback is antithetical to principles of tutorial dialogue;
- (3) reflection and diagnosis should be performed by the learner, not the tutor;
- (4) learner modeling is technically very difficult; and
- (5) the assumptions on which most modeling approaches are based are applicable to procedural learning, whereas the emphasis should be on critical thinking and problem solving.

What does this say about authoring tools? Though much contemporary ITS research adopts as an assumption the requirement for a student model, we can say at the very least that the need for a student model is governed by the instructional objectives of the ITS. And the capabilities of simulation-based learning ITSs can often reduce, or eliminate, the need for a student model. So to support ITS authoring, tools should support tagging and tracking the user’s actions in order to correlate user activity with the state of the world, in support of feedback and assessment.⁴

More broadly, a capability that should be characteristic of tools for authoring ITSs in this genre is to help the author define states of the world and transitions in the world that correspond to instructionally significant events. How simulations can be controlled to achieve a desired instructional outcome has been the subject of much research. “Open world” learning environments require structure to align the interaction with instructional objectives. Lane & Johnson (2008) discuss the need for guided practice, to make more tractable the problem of managing tutoring given the additional dimensions of time and movement that simulations add to ITS. Guided Experiential Learning (Clark, Yates, Early & Moulton, 2010), mentioned previously, proposes a structured, seven-step process to ensure an instructionally effective sequence is observed in using discovery learning environments.

Constraining simulations in instructionally meaningful ways has enabled much recent work that blends instructional strategies and simulation-driven ITS. Researchers have been investigating methods for identifying “teachable moments” (Havighurst, 1952) during exploratory interactions. One technique is to align the content (i.e., the target skills and knowledge) to a user’s goals and then employ the user’s behaviors to trigger the presentation of the corresponding content (Mall, et al., 2014). Another is to modulate responses provided through the simulation (e.g., through animated agents) to increase or decrease feedback and advance the instructional aims of the interaction (Lane & Johnson, 2008). Related to this approach is the explicit modification of the world state to condition the environment for addressing specific learning objectives (Magerko, Stensrud & Holt, 2006).

³ This example was taken from a simulation-driven learning environment developed for US Air Force pilot training (Bell, Bennett, Billington, S., Ryder & Billington, I., 2010).

⁴ A simulation scenario can be an objective and complete assessment rubric as numerous researchers have observed (Schank, 2001; Fowlkes, Dwyer, Oser & Salas, 1998; Bell, et al., 2010).

Although these techniques show promise, there remains the question of how to incorporate them into authoring tools. Creating the simulation environment itself is not the province of an ITS authoring tool; simulation construction is a complex and distinctive process and demands a different skill set and correspondingly tailored suite of tools. Instead, ITS authoring should evolve to allow training developers to integrate tutoring with simulations.

Implications of Discourse-Based Learning for ITS Authoring

The second-person interface metaphor discussed previously has cultivated a rich body of research exploring the interactions between a user and an ITS. The central challenges faced by an author of a dialogue-driven ITS are unique to this genre and thus would be optimally overcome by an authoring tool that understands discourse-based learning.

One obvious way in which creating discourse-driven ITS diverges from the traditional model (and thus from traditional notions of authoring tools to support that model) is the blurring of any distinction between the “tutor module” and the “interface module.” Tutoring knowledge can be encapsulated directly within the discourse space and in how that space is traversed (by dialogue events triggering state transitions). Not all discourse models operate precisely in this way but the challenges of the authoring process largely remain across implementations. A model introduced previously that illustrates the novel requirements for authoring discourse-driven ITS is AutoTutor. The process of creating AutoTutor applications implicitly merges the tutoring and interface modules, and requires that an author develop a well-elaborated conversation space.

Among the authoring challenges that set this ITS genre apart from the other two examples is that the dialogue must be structured to completely address the intended learning objectives. It also follows that a tool to support this kind of authoring should embody whatever dialogue theory the author is implementing. In other words, creating an AutoTutor application is best supported by an ITS authoring tool that understands the expectation and misconception-tailored (EMT) discourse model, so that the tool can effectively coach an author in structuring the dialogue in a way that ensures the instructional objectives are achieved.

An example of such a tool is the AutoTutor Authoring Tools (ASAT), designed to support authors in creating the underlying rules to achieve the intended tutoring dialogue (Graesser et al., 2004). Similar tools have been developed to support the authoring of ITS implemented via a related framework called AutoTutor Lite (Wolfe, et al., 2013). A salient characteristic of these tools relevant to the current discussion is that they explicitly embody an instructional theory and are therefore able to support the ITS author. The authoring tool guides the author by presenting the elements of the dialogue that have to be defined and the actions and transitions that make the dialogue dynamic and instructionally relevant.

Implications of Situation-Based Learning for ITS Authoring

As discussed, situation-based learning is somewhat of a hybrid, adopting both first- and second-person interface elements. In this section, I consider the process of authoring this kind of ITS and present examples of ITS tools developed to support the process. Situation-based learning can be, and usually is, implemented as a network or graph of states (each corresponding to a situation the user has arrived at through actions taken). Authoring a branching simulation requires defining the state space, creating the transitions among states, and elaborating each state. The mechanics can be relatively straightforward. The challenge is more in developing a coherent and compelling narrative that squarely addresses the learning objectives of the ITS. As a result, the creation of a branched-simulation ITS has often been coincident with the evolution of an authoring tool used to support that ITS or its immediate progeny. For instance,

early work by Ohmaye (1998) in creating a language tutor yielded an architecture and authoring tool for developing dialogue-driven branched simulations that was further refined by Guralnick (1996) and Jona (1998).

This approach has been generally referred to as outcome-driven simulation, a term coined by Christopher Riesbeck at Northwestern University in 1994 that “refers to a class of applications where users adopt a role in a fictional scenario, and where the decisions and action that the user takes moves the scenario forward in time to new situations that are relevant to the pedagogical objectives” (Gordon, 2004, p. 230).

This simple architecture can create quite vivid user experiences, creating the impression of a dynamic simulation, and continues to be employed to both create learning applications and to develop authoring tools tailored to supporting this genre of ITS. Gordon (2004) described a process to support authoring and its application to leadership training for US Army officers. A related authoring tool based on this architecture was developed along with several applications to support cultural awareness training (Deaton, et al., 2005). This approach remains widely used to this day. For instance, a suite of medical training applications is being developed around an outcome-driven simulation ITS authoring tool (King, Scott, Davidson & Bope, 2014).

This work is related to the goal-based scenario (GBS) framework introduced previously (Schank, Fano, Bell & Jona, 1994). The GBS research team proposed five categories of ITS, named for the principal activity anchoring learning: advise, investigate and decide, run, script, and persuade (Jona & Kass, 1997). The research team then developed specialized ITS authoring tools to support the construction of GBS within each specific category, and conducted numerous evaluations and user trials (e.g., Bell, 1998). These authoring frameworks continued to evolve, and today several tools are in use that support authoring of GBS as well as specific sub-categories of these situation-based ITS (e.g., investigate and decide; see Bell, 2003; Dobson, 1998; Dobson & Riesbeck, 1998; Riesbeck & Dobson, 1998).⁵

These variants share an approach to instruction effected through the states and transitions. Tutoring is implicitly represented in the states and transitions defined by the author, and dynamically engages the user as states are traversed, with transitions triggered by the user’s decisions and actions.

What these examples of authoring tools tell us is that outcome-driven simulation is created through an authoring process that is distinctive from the traditional ITS component approach. Outcome-driven simulation as an instance of situation-based learning has advantages in terms of facilitating authoring and ensuring instructional goals are achieved. This approach “demonstrates that training experiences in virtual reality environments need not be constrained by the modeling limitations of current constructive simulations, and that by focusing on specific decision situations we can design immersive training environments that are tightly structured around training goals” (Gordon, 2004, p. 237).

Discussion

This chapter illustrates that ITS as a research enterprise has matured and diversified, reflecting a broader theme of this volume, which similarly segments the ITS space (though along different lines—model-tracing, agent-based, and dialogue-based). The categories themselves are less important than the question of authoring tools, and specifically, whether we should strive to create a universal ITS tool or acknowledge the diversity of ITS and develop authoring tools specific to different types.

⁵ The GBS Tool and commercial variants are developed and marketed by Socratic Arts, www.socraticarts.com.

This call for distinctive authoring tools is not meant to suggest that different domains call for different tools. Authoring tools should be agnostic with respect to domain; whether an author wishes to teach about combat casualty care or playground etiquette is not the issue—the *how* is more determinant than the *what*. It does not even matter greatly whom the intended audience is. As Murray (2003) points out, “the key differences among ITS authoring systems are not related to specific domains or student populations, but to the domain-independent capabilities that the authored ITSs have.” (p. 495).

While the domain itself may not be a factor in authoring tool design, research being conducted within the GIFT community is drawing an important distinction between *well-defined* and *ill-defined* domains. The implication is that “with a two-dimensional approach to domain definition, instructional strategies can be specified based on the component characteristics identified within the domain designation” (Goldberg, Holden, Brawner & Sotilare, 2011). So at least at this high level, we see a trajectory for GIFT to support distinctive authoring processes based on domain.

GIFT researchers also call for specificity at the domain level when authoring assessments. “The fundamental problems of domain dependent components are how to assess student actions, how to respond to instructional changes, how to respond to requests for immediate feedback, and an interface which supports learning” (Goldberg, Holden, Brawner & Sotilare, 2011). However, the GIFT framework manages these differences through domain modeling tools, which allow for author-customized and domain-specific feedback and assessment but which do not present distinctive approaches to instruction.

A dimension more discriminating for authoring tools than domain is instructional approach. An assertion drawn from this chapter is that ITS tools should (and do) embody a specific instructional theory to support authoring ITS that also embody that theory (e.g., Adenowo & Patel, 2014; Aleven, McLaren & Sewall, 2009; Gordon, 2004; Ramachandran & Sorensen, 2007). The GIFT framework takes a more generic approach than authoring in specific ITS genres—ITS authors instead “can use GIFT to author strategies aligned with a particular instructional theory” and “have access to libraries of strategies that are tailored to the user and can be used to develop timely feedback mechanisms.” (Sotilare, Goldberg, Brawner & Holden, 2012). GIFT thus seeks a best-of-both-worlds solution, by offering a generic suite of authoring tools while supporting the construction of ITS in a range of instructional traditions.

Where the GIFT approach diverges from proponents of theory-specific ITS authoring tools is not so much in the *availability* of strategies but in the knowledge that the authoring tool can bring to bear in helping the author to properly select and apply those strategies. As an analogy, consider the difference between a website that lists airline routes and schedules and an experienced travel agent. The GIFT approach can offer a library of tutoring strategies that provides the savvy ITS author with flexibility but which does little to support a novice ITS author (for instance, a subject matter expert) in creating effective instruction.

Recommendations and Future Research

The mission of GIFT is more than supporting authoring—GIFT is oriented around providing three services: authoring of components, management of instructional processes, and an assessment methodology (Sotilare, Goldberg, Brawner & Holden, 2012). This volume’s focus on authoring tools provides a range of perspectives on tutoring approaches and how to best support authors in creating effective ITS.

GIFT is addressing a difficult problem during a time of rapid change. The convergence of ITS with immersive games, for instance, creates numerous authoring challenges, such as how to support the creation of reactive agents characteristic of ITS and proactive agents characteristic of simulations (Brawner, Holden, Goldberg & Sotilare, 2012). Such trends are blurring long-standing boundaries. The

blending of tutoring and gaming is likely to raise questions about the relative importance of a tutor in an ITS (and implications for authoring tool design). Jona & Kass (1997) may have anticipated the ascendance of gaming in questioning the assumption “that the central component of a learning environment is the tutor, and that the critical learning events are interactions between the learner and the tutor.” They assert instead that “this view is not compatible with what many who study education and human learning have found. For example, many progressive educators would argue that the most important aspect of a learning environment is a complex, realistic activity in which the learner becomes engaged, and *not* the tutoring received” (p. 39, original emphasis).

GIFT, in fulfilling the vision of technology that is generalizable and integrated, is promoting modularity, reuse and broad applicability (Sottilare, Goldberg, Brawner & Holden, 2012). It remains to be seen whether this approach is theory-neutral or an aggregation of multiple theories. Also, some might question whether an ITS authoring tool (which is intended to support the creation of end-products grounded in some instructional theory) can even be theory-neutral. As observed by Jona & Kass (1997): “The mix and match approach is not theoretically neutral with regard to the questions of what really drives learning, and what are the central features of a learning environment” (p. 39).

I conclude by revisiting the general aims of an authoring tool. Reducing the effort needed to produce ITS can include the following:

- assuming responsibility for mechanical aspects of the task;
- furnishing predefined elements that an author can package together to suit a particular need; and
- guiding the author.

GIFT in its early stages has established a promising framework for supporting some of the mechanical aspects of ITS construction (Sottilare, Goldberg, Brawner & Holden, 2012). Accelerating the authoring process with predefined elements has more numerous and nuanced dimensions. There is some appeal to thinking of authoring as aligning old components in new ways; however, authors would require visibility into the properties of these components, what can be customized, and how to link them. There are numerous metaphors that might inspire novel approaches to addressing this need. For instance, preparing a new dish is something people generally do by adapting a recipe that not only lists the ingredients but also instructs in how the ingredients are to be combined and even what substitutions might be tried. Applying this metaphor to GIFT, we can envision a library of completed ITSs, each cataloging its components, their properties, and instructions on adapting each for reuse. As this framework becomes populated with more content, GIFT will advance in its support for providing such predefined elements and libraries that ITS authors can incorporate and adapt. It should be emphasized, however, that simply making a large collection of ITS components available to an author is not sufficient; in the cooking analogy, it is more akin to roaming the aisles of a grocery store than to browsing a recipe book.

It is just this sort of guidance for the author that will require increased attention. As the research reviewed in this chapter suggests, an authoring tool should have some understanding of what the author wishes to create and be able to offer useful and specific support. The forms such support take can vary from recommendations about low-level dialogue elements to presenting a worked example similar to the author’s intended ITS (as recommended in Hsu & Moore, 2011) and supporting its incremental adaptation (which we term *guided-case adaptation*, see Bell, 1998).

The needs remains for ongoing evaluation, of GIFT’s authoring tools and of the products emerging from authors using the framework. As the ranks of GIFT contributors continue to expand, greater opportunities will become available to study how GIFT supports ITS authoring, and ITS will emerge that will provide

researchers with artifacts to evaluate. The instructional effectiveness of products created using GIFT will provide formative direction to the evolution of the framework, and will ultimately be a persuasive indicator of the value of GIFT to the ITS community.

References

- Adenowo, A. A. A., and Patel, A. M. (2014). A metamodel for designing an intelligent tutoring systems authoring tool. *Computer and Information Science*, 7(2), 82-98.
- Ahuja, N.J., and Sille, R. (2013). A Critical Review of Development of Intelligent Tutoring Systems: Retrospect, Present and Prospect. *International Journal of Computer Science Issues*, Vol. 10, Issue 4, No 2, July 2013.
- Aleven, V., McLaren, B. M., and Sewall, J. (2009). Scaling up programming by demonstration for intelligent tutoring systems development: An open-access website for middle-school mathematics learning. *IEEE Transactions on Learning Technologies*, 2(2), 64-78.
- Bell, B.L. (1998). Investigate and Decide Learning Environments: Specializing Task Models for Authoring Tool Design. *The Journal of the Learning Sciences*, 7(1), 65-105.
- Bell, B. (2003). Supporting Educational Software Design with Knowledge-Rich Tools (2003). In T. Murray, S. Blessing, and S. Ainsworth (Eds.), *Authoring Tools for Advanced Technology Learning Environments: Toward cost-effective adaptive, interactive, and intelligent educational software*. Kluwer Academic Publishers: Dordrecht, Netherlands.
- Bell, B., Billington, S., Bennett, W., Billington, I., and Ryder, J. (2010). Performance gains from speech-enhanced simulation in military flying training. *Journal of Defense Modeling and Simulation*, 7(2), 67-87.
- Bell, B., Jarmasz, J., and Nelson, I. (2011). Development of Scenario-Based Pre-deployment Counter-IED Training. In *Proc. of Interservice/Industry Training, Simulation, and Education Conference* (I/ITSEC), Dec, 2011.
- Bell, B., Johnston, J., Freeman, J., and Rody F. (2004). STRATA: DARWARS for Deployable, On-Demand Aircrew Training. In *Proc. of the Interservice/Industry Training, Simulation, and Education Conference* (I/ITSEC), December, 2004.
- Bell, B.L., and Zirkel, J. (2001). "Goal Directed Inquiry via Exhibit Design: Engaging with History through the Lens of Baseball". *Journal of Interactive Learning Research*, 12(1), 3-39.
- Bransford, J.D., Sherwood, R.D., Hasselbring, T.S., Kinzer, C.K., and Williams, S.M. (1990). Anchored instruction: Why we need it and how technology can help. In D. Nix and R. Sprio (Eds), *Cognition, education and multimedia*. Hillsdale, NJ: Erlbaum Associates.
- Brawner, K. and Graesser, A.C. (2014). Natural Language, Discourse, and Conversational Dialogues within Intelligent Tutoring Systems: A Review. In R. Sottilare, A. Graesser ,X. Hu and B. Goldberg (Eds.), *Design Recommendations for Intelligent Tutoring Systems: Volume 2 Instructional Management*. US Army Research Laboratory.
- Brawner, K., Holden, H., Goldberg, B., Sottilare, R. (2012). Recommendations for Modern Tools to Author Tutoring Systems. In *Proc. of the Interservice/Industry Training, Simulation, and Education Conference* (I/ITSEC), December, 2012.
- Brown, J.S., Collins, A., and Duguid, P. (1989). Situated cognition and the culture of learning. *Educational Researcher*, 8(1), 32-42.
- Brown, J. S., and VanLehn, K. (1988). Repair theory: A generative theory of bugs in procedural skills. In A. Collins & E. E. Smith (Eds.), *Readings in Cognitive Science* (pp. 338-361). Los Altos, CA: Morgan Kaufmann.
- Brusilovsky, P. (1994). The Construction and Application of Student Models in Intelligent Tutoring Systems. *Journal of Computer and Systems Sciences International*, 32(1), 70-89.
- Burns, H.L., and Capps C.G. (1988). Foundations of intelligent tutoring systems : an introduction. Martha C. Polson, J. Jeffrey Richardson. *Foundations of Intelligent Tutoring Systems*, Hillsdale, N.J.: Lawrence Erlbaum Associates, pp.1-19.
- Burton, R.R. (1988). The environment module of intelligent tutoring systems. In Polson, M.C. and Richardson, J.J. (Eds.), *Foundations of intelligent tutoring systems*. Hillsdale: Lawrence Erlbaum Associates.
- Burton, R. and Brown, J. (1978). Diagnostic models for procedural bugs in basic mathematical skills. *Cognitive Science*, 2, 155-191.
- Clark, R. E., Yates, K., Early, S., and Moulton, K. (2010). An analysis of the failure of electronic media and discovery-based learning: Evidence for the performance benefits of guided training methods. In K. H.

- Silber & R. Foshay (Eds.), *Handbook of Training and Improving Workplace Performance* (Vol. I: Instructional Design and Training Delivery), pp. 263-329. New York: Wiley and Sons.
- Collins, A., Brown, J.S., and Newman, S.E. (1989). Cognitive apprenticeship: Teaching the crafts of reading, writing, and mathematics. In L. B. Resnick (Ed.) *Knowing, learning, and instruction: Essays in honor of Robert Glaser* (pp. 453-494). Hillsdale, NJ: Lawrence Erlbaum Associates.
- Deaton, J., Barba, C., Santarelli, T., Rosenzweig, L., Souders, V., McCollum, C., Seip, J., Knerr, B. and M. Singer (2005). Virtual environment cultural training for operational readiness (VECTOR). *Journal of Virtual Reality*, 8(3) (May 2005), 156-167.
- Derry, S. J., and Lajoie, S. P. (1993). A middle camp for (un)intelligent instructional computing: An introduction. In S. P. Lajoie and S. J. Derry (Eds.), *Computers as cognitive tools* (pp. 1-11). Hillsdale, NJ: Lawrence Erlbaum Associates
- D'Mello, S. K. and Graesser, A. C. (2012). AutoTutor and affective AutoTutor: Learning by talking with cognitively and emotionally intelligent computers that talk back. *ACM Transactions on Interactive Intelligent Systems*, 2(4), 23: 1-38.
- Dobson, W. (1998). Authoring Tools for Investigate-and-Decide Learning Environments. Ph.D. Dissertation, Northwestern University Department of Computer Science, Evanston, IL, June, 1998.
- Dobson, W., and Riesbeck, C.K. (1998). "Tools for Incremental Development of Educational Software Interfaces". In *CHI '98. Conference Proceedings on Human Factors in Computing Systems*, 384-391.
- Fowler, S., Smith, B., and Litteral, D.J. (2005). A TC3 Game-based Simulation for Combat Medic Training. In *Proc. of Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC)*, Dec, 2005.
- Fowlkes, J. E., Dwyer, D., Oser, R. L. & Salas, E. (1998). Event-Based Approach to Training (EBAT). *The International Journal of Aviation Psychology*, 8 (3), 209-221.
- Frederiksen, J., and White, B. (2002). Conceptualizing and constructing linked models: Creating coherence in complex knowledge systems. In P. Brna, M. Baker, K. Stenning and A. Tiberghien (Eds.), *The Role of Communication in Learning to Model*. (pp. 69-96). Mahwah, NJ: Erlbaum.
- Goldberg, B.S., Holden, H.K., Brawner, K.W., and Sottilare, R.A. (2011). Enhancing Performance through Pedagogy and Feedback: Domain Considerations for ITSs. In *Proc. of Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC)*, Dec, 2011.
- Gordon, A.S. (2004). Authoring branching storylines for training applications. In *Proceedings of the 6th International Conference of the Learning Sciences (ICLS '04)*. pp 230-237.
- Graesser, A., Chipman, P., Haynes, B. and Olney, A. (2005). AutoTutor: An intelligent tutoring system with mixed-initiative dialogue. *IEEE Transactions on Education*, 48(4), 612-618.
- Graesser, A., D'Mello, S., Hu, X., Cai, Z., Olney, A. and Morgam, B. (2012). AutoTutor. Applied natural language processing: Identification, investigation, and resolution. Hershey, PA: IGI Global.
- Graesser, A. C., Lu, S., Jackson, G. T., Mitchell, H. H., Ventura, M., Olney, A. and Louwerse, M. M. (2004). AutoTutor: A tutor with dialogue in natural language. *Behavior Research Methods, Instruments and Computers*, 36(2), 180-192.
- 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.
- Graesser, A. C., Wiemer-Hastings, K., Wiemer-Hastings, P. and Kreuz, R. (1999). AutoTutor: A simulation of a human tutor. *Cognitive Systems Research*, 1(1), 35-51.
- Greer, J. and Koehn, G.M. (1995). The peculiarities of plan recognition for intelligent tutoring systems (1995). In *Proc. of the IJCAI Workshop on the Next Generation of Plan Recognition Systems*, pp. 54–59, 1995.
- Guralnick, D. (1996). Training systems for script-based tasks. Ph.D. dissertation, The Institute for the Learning Sciences, Northwestern University.
- Hartley, J. R., and Sleeman, D. H. (1973). Towards more intelligent teaching systems. *International Journal of Man-Machine Studies*, 2, 215-236.
- Havighurst, R. J. (1952). Developmental tasks and education. New York: David McKay.
- Hsu, C-H, and Moore, D. R. (2011). Formative research on the Goal-based Scenario model applied to computer delivery and simulation. *The Journal of Applied Instructional Design*, 1(1), 13-24.
- Johnson, L. W. and Valente, A. (2008). Tactical language and culture training systems: Using artificial intelligence to teach foreign languages and cultures. In *Proceedings of the Twentieth Conference on Innovative Applications of Artificial Intelligence* (pp. 1632-1639). Menlo Park, CA: AAAI Press.
- Jona, M (1998). Representing and Applying Teaching Strategies in Computer-based learning-by-doing Tutors. In R.C. Schank (Ed.), *Inside Multi-media Case Based Instruction*. Malwhah, NJ: Lawrence Erlbaum Associates.

- Jona, M.K., and Kass, A.M. (1997). A Fully-Integrated Approach to Authoring Learning Environments: Case Studies and Lesson Learned. In *Intelligent Tutoring System Authoring Tools: Papers from the 1997 Fall Symposium*, Technical Report FS-97-01, AAAI, P. 39.
- Jong, T. de, and van Joolingen, W.R. (1998). Scientific discovery learning with computer simulations of conceptual domains. *Review of Educational Research*, Vol. 68 No. 2, pp. 179-201.
- King, K.S., Scott, R., Davidson, M., and Bope, E. (2014). Branching Simulation Designs for Virtual Patients. Presented at the MedBiquitous Conference, Baltimore, MD.
- Lane, H.C. and Johnson, W.L. (2008). Intelligent Tutoring and Pedagogical Experience Manipulation in Virtual Learning Environments, in D. Schmorow, J. Cohn & D. Nicholson (Eds), *The PSI Handbook of Virtual Environments for Training and Education: Developments for the Military and Beyond*. Praeger Security International: Westport, CN.
- Lester, J., Lobene, E., Mott B. & Rowe, J. (2014). Serious Games with GIFT: Instructional Strategies, Game Design, and Natural Language in the Generalized Intelligent Framework for Tutoring. In R. Sotilare, A. Graesser ,X. Hu and B. Goldberg (Eds.), *Design Recommendations for Intelligent Tutoring Systems: Volume 2 Instructional Management*. US Army Research Laboratory.
- Livak, T., Heffernan, N. T. & Moyer, D. (2004) Using cognitive models for computer generated forces and human tutoring. Presented at the *13th Annual Conference on Behavior Representation in Modeling and Simulation*. Simulation Interoperability Standards Organization, Arlington, VA.
- Macmillan, S., Emme, D., and Berkowitz, M. (1988). Instructional Planners: Lessons Learned. In Psotka, J., Massey, L.D., and Mutter, S.A. (Eds.), *Intelligent Tutoring Systems, Lessons Learned*. Lawrence Erlbaum: Hillsdale, NJ.
- Magerko, B., Stensrud, B. and Holt, L. S. (2006). *Bringing the schoolhouse inside the box - A tool for engaging, individualized training*. Paper presented at the 25th Army Science Conference, Orlando, FL.
- Mall, H., Martin, E., Robson, R., Ray, F., Veden, A., and Robson, E. (2014). In Search of the Teachable Moment. In *Proceedings of the 2014 Interservice/Industry Training, Simulation, and Education Conference*.
- Miller, J. R. (1988). The role of human-computer interaction in intelligent tutoring systems. In M. C. Polson and J.J. Richardson (Eds.), *Foundations of Intelligent Tutoring Systems*. Hillsdale, N.J.: Lawrence Erlbaum Associates, pp. 143-189.
- Munro, A., Johnson, M.C., Pizzini, Q.A., Surmon, D.S., Towne, D.M., and Wogulis, J.L. (1997). Authoring Simulation-Centered Tutors with RIDES. *International Journal of Artificial Intelligence in Education*, 1997(8), 284-316.
- Murray, T. (1999). Authoring Intelligent Tutoring Systems: An analysis of the state of the art. *International Journal of Artificial Intelligence in Education (IJAIED)*, 1999, 10, pp.98-129.
- Murray, T. (2003). An overview of intelligent tutoring system authoring tools. In T. Murray, S. Blessing, and S. Ainsworth (Eds.), *Authoring Tools for Advanced Technology Learning Environments: Toward cost-effective adaptive, interactive, and intelligent educational software*. Kluwer Academic Publishers: Dordrecht, Netherlands.
- Murray, T. and Woolf, B.P. (1992). A Knowledge Acquisition Tool for Intelligent Computer Tutors. *SIGART Bulletin*, 2, 9–21.
- Ohmaye, E. (1998). Simulation-based language learning: an architecture and a multimedia authoring tool. In R.C. Schank (Ed.), *Inside Multi-media Case Based Instruction*. Malwah, NJ: Lawrence Erlbaum Associates.
- Paviotti, G., Rossi, P.G., and Zarka, D. (2012). *Intelligent Tutoring Systems: An Overview*. Lecce: Pensa Multimedia, Italy.
- Pavlik, P.I. Jr., Brawner, K., Olney, A., and Mitrovic, A. (2013). A Review of Student Models Used in Intelligent Tutoring Systems. In R. Sotilare, A. Graesser ,X. Hu and H. Holden (Eds.), *Design Recommendations for Intelligent Tutoring Systems: Volume 1 Learner Modeling*. US Army Research Laboratory.
- Ramachandran, S., and Sorensen, B. (2007). From Simulations to Automated Tutoring. *Proceedings of the Fifteenth Conference on Medicine Meets Virtual Reality (MMVR 2007)*, Long Beach, CA.
- Rickel J., and Johnson, W.L. (1999). Animated agents for procedural training in virtual reality: perception, cognition, and motor control. *Applied Artificial Intelligence* 1999(13): 343-82.
- Riedl, M.O. and Young, R.M. (2014). The Importance of Narrative as an Affective Instructional Strategy. In R. Sotilare, A. Graesser ,X. Hu and B. Goldberg (Eds.), *Design Recommendations for Intelligent Tutoring Systems: Volume 2 Instructional Management*. US Army Research Laboratory.
- Riesbeck, C.K., and Dobson, W. (1998). Authorable Critiquing for Intelligent Educational Systems. In *Proceedings of the 1998 International Conference on Intelligent User Interfaces*, January 6-9, 1998, San Francisco, CA.

- Rus, V., D'Mello, S., Hu, X. and Graesser, A. C. (2013). Recent advances in intelligent systems with conversational dialogue. *AI Magazine*, 42-54.
- Russell, D., Moran, T.P., and Jordan, D.S. (1988). The Instructional Design Environment. In Psotka, J., Massey, L.D., and Mutter, S.A. (Eds.), *Intelligent Tutoring Systems, Lessons Learned*. Lawrence Erlbaum: Hillsdale, NJ.
- Sani, S., and Aris, T.N.M. (2014). Computational Intelligence Approaches for Student/Tutor Modelling: A Review . In *Proceedings of the 2014 Fifth International Conference on Intelligent Systems, Modelling and Simulation*. Langkawi, Malaysia: IEEE.
- Schank, R.C. (2001). Designing World Class E-Learning: How IBM, GE, Harvard Business School, and Columbia University Are Succeeding at E-Learning. New York: McGraw-Hill.
- Schank, R.C., Fano, A., Bell, B.L., and Jona, M.Y. (1994). The Design of Goal Based Scenarios. *The Journal of the Learning Sciences*, 3(4), 305–345.
- Shute, V. J., and Psotka, J. (1996). *Intelligent tutoring systems: Past, present, and future*. In D. Jonassen (Ed.), *Handbook of research for educational communications and technology* (pp. 570-600). New York, NY: Macmillan.
- Sottilare R. (2012). Considerations in the development of an ontology for a Generalized Intelligent Framework for Tutoring. International Defense and Homeland Security Simulation Workshop, in *Proceedings of the I3M Conference*. Vienna, Austria, September 2012.
- Sottilare, R.A., DeFalco, J.A., and Connor, J. (2014). A Guide to Instructional Techniques, Strategies and Tactics to Manage Learner Affect, Engagement, and Grit. In R. Sottilare, A. Graesser ,X. Hu and B. Goldberg (Eds.), *Design Recommendations for Intelligent Tutoring Systems: Volume 2 Instructional Management*. US Army Research Laboratory.
- Sottilare, R.A., Goldberg, B.S., Brawner, K.W., and Holden, H.K. (2012). A Modular Framework to Support the Authoring and Assessment of Adaptive Computer-Based Tutoring Systems (CBTS). In *Proc. of Interservice/Industry Training, Simulation, and Education Conference* (I/ITSEC), Dec, 2012.
- Towne, D.M., and Munro, A. (1988). The Intelligent Maintenance Training System. In Psotka, Massey, and Mutter (Eds.), *Intelligent Tutoring Systems, Lessons Learned*. Hillsdale, NJ: Lawrence Erlbaum.
- VanLehn, K. (2011). The relative effectiveness of human tutoring, intelligent tutoring systems and other tutoring systems. *Educational Psychologist*, 46(4), 197-221.
- Veermans, K., Joolingen, W.R. van, and de Jong, T. (2000). Promoting Self-Directed Learning in Simulation Based Discovery Learning Environments Through Intelligent Support. *Interactive Learning Environments*, 8, 257-277: Taylor and Francis.
- Whitaker , E.T., and Bonnell, R.D. (1990), Plan recognition in intelligent tutoring systems. In *Proceedings of Intelligent Tutoring Media*, 1(2), 73-82.
- Wolfe, C. R., Widmer, C. L., Reyna, V. F., Hu, X., Cedillos, E. M., Fisher, C. R., and Weil, A. M. (2013). The development and analysis of tutorial dialogues in AutoTutor lite. *Behavior Research Methods* 45(3), 623-36.
- Wray, R. E., Woods, A., and Priest, H. (2012). Applying Gaming Principles to Support Evidence-based Instructional Design. In *Proceedings of the 2012 Interservice/Industry Training, Simulation, and Education Conference*, Orlando.

CHAPTER 4 Generalizing the Genres for ITS: Authoring Considerations for Representative Learning Tasks

Benjamin D. Nye¹, Benjamin Goldberg², Xiangen Hu^{1,3}

¹University of Memphis, ²ARL-LITE Lab, ³Central China Normal University

Introduction

Compared to many other learning technologies, intelligent tutoring systems (ITSs) have a distinct challenge: authoring an adaptive inner loop that provides pedagogical support on one or more learning tasks. This coupling of tutoring behavior to student interaction with a learning task means that authoring tools need to reflect both the learning task and the ITS pedagogy. To explore this issue, common learning activities in intelligent tutoring need to be categorized and analyzed for the information that is required to tutor each task. The types of learning activities considered cover a large range: step-by-step problem solving, bug repair, building generative functions (e.g., computer code), structured argumentation, self-reflection, short question answering, essay writing, classification, semantic matching, representation mapping (e.g., graph to equation), concept map revision, choice scenarios, simulated process scenarios, motor skills practice, collaborative discussion, collaborative design, and team coordination tasks. These different tasks imply a need for different authoring tools and processes used to create tutoring systems for each task. In this chapter, we consider three facets of authoring: (1) the minimum information required to create the task, (2) the minimum information needed to implement common pedagogical strategies, (3) the expertise required for each type of information. The goal of this analysis is to present a roadmap of effective practices in authoring tool interfaces for each tutoring task considered.

A long-term vision for ITSs is to have generalizable authoring tools, which could be used to rapidly create content for a variety of ITSs. However, it is as-yet unclear if this goal is even attainable. Authoring tools have a number of serious challenges from the standpoint of generalizability. These challenges include the domain, the data format, and the author. First, different ITS domains require different sets of authoring tools, because they have different *learning tasks*. Tools that are convenient for embedding tutoring in a 3D virtual world are completely different than ones that make it convenient to add tutoring to a system for practicing essay-writing, for example. Second, the data produced by an authoring tool need to be consumed by an ITS that will make pedagogical decisions. As such, at least some of the data are specific to the pedagogy of the ITS, rather than directly reflecting domain content. As a simple example, if an ITS uses text hints, those hints need to be authored, but some systems may just highlight errors rather than providing text hints. As such, the first system actually needs more content authored and represented as data. With that said, typical ITSs use a relatively small and uniform set of authored content to interact with learners, such as correctness feedback, corrections, and hints (VanLehn, 2006). Third, different authors may need different tools (Nye, Rahman, Yang, Hays, Cai, Graesser & Hu, 2014). This means that even the same content may need distinct authoring tools that match the expertise of different authors.

In this chapter, we are focusing primarily on the first challenge: *differences in domains*. In particular, our stance is that the “content domain” is too coarse-grained to allow much reuse between authoring tools. This is because, to a significant extent, content domains are simply names for related content. However, the skills and pedagogy for the same domain can vary drastically across different topics and expertise levels. For example, algebra and geometry are both high school level math domains. However, in geometry, graphical depictions (e.g., shapes, angles) are a central aspect of the pedagogy, while algebra tends to use graphics very differently (e.g., coordinate plots). As such, some learning tasks tend to be shared between those subdomains (e.g., equation-solving) and other tasks are not (e.g., classifying shapes).

This raises the central point of our chapter: the learning tasks for a domain define how we author content for that domain. For example, while algebra does not involve recognizing many shapes, understanding the elements of architecture involves recognizing a variety of basic and advanced shapes and forms. In total, this means that no single whole-cloth authoring tool will work well for any pair of algebra, geometry, and architectural forms. However, it also implies that a reasonable number of task-specific tools for each learning task might allow authoring for all three domains. To do this, we need to understand the common learning tasks for domains taught using ITSs and why those tasks are applied to those domains. In the following sections, we identify and categorize common learning tasks for different ITS domains. Then, we extract common principles for those learning tasks. Finally, we suggest a set of general learning activities that might be used to tutor a large number of domains.

What is a Learning Task?

Before we begin, it is important to define what we mean by a learning task. Functionally speaking, a learning task is an activity designed to help the participant(s) learn certain knowledge or skills. Any learning task has a three essential parts:

- (1) Task State (S_T) - the context and status of the task,
- (2) Task Interface (I_T) - the representation used to present the task and its available actions,
- (3) Task Goals (G_T) - importance or value given to states or state trajectories, which may be stated in the task, given prior to the task (e.g., by a teacher), or chosen by the learner.

A directed learning task, such as one run by a teacher or an ITS (as opposed to an undirected sandbox activity), also has complementary parts related to the instructor's control over the system:

- (1) Pedagogical State (S_P) - the context and status relevant to pedagogical decision making,
- (2) Pedagogical Interface (I_P) - the pedagogical actions available during a task, and
- (3) Pedagogical Goals (G_P) - importance or value given to reaching certain pedagogical states.

The relationships between these parts are noted in Figure 1. From an ITS authoring standpoint, both the task and the pedagogical model need to be authored. In this representation, the pedagogical state includes the task goals, task state, and the learner's state (e.g., a student model). In this respect, the pedagogical state is more complex than the task state. However, excluding the learner's internal state (which is only observable through the history of task interactions) and the task goals (which are typically not changed during a given task), the pedagogical state is *by definition* less complex than the task state. Considering a task as a Markov decision process, the pedagogical state trajectory S_P cannot consider any more information from the task beyond its trajectory of task states (S_T). In most cases, the representation of the pedagogical state is far simpler and based on features sets such as classifying good/bad answers, identifying specific misconceptions or bugs, and other assessments that reduce even rich environments (e.g., 3D simulators) into streams of simpler features that form the pedagogical state used for triggering interventions such as hints (Kim et al., 2009; Nye, Graesser & Hu, 2014; VanLehn, 2006).

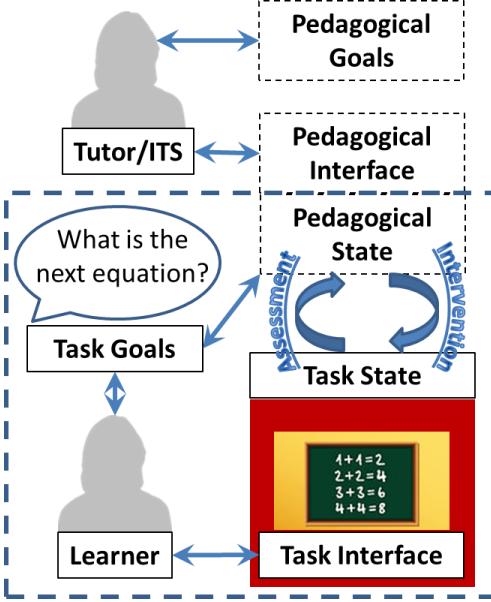


Figure 1: Tasks and pedagogy

This implies that ITS authoring should be greatly constrained by the learning task. At face value, it seems like there might be exceptions: a tutor for metacognitive skills might need to know almost nothing about learner's performance on their primary learning task, if it only suggests self-reflection and an unassessed summarization. However, it can be argued that such a task-agnostic tutor has that capability only because it generates its *own* learning tasks (e.g., journals for summarization, delays for self-reflection). This has two implications:

- First, it implies that all ITS authoring is tied to a specific set of tasks.
- Second, it implies that multiple learning tasks may be interleaved or even occurring simultaneously.

In simulation-based training for complex tasks, such as flight simulators or cross-cultural competencies, working on multiple tasks simultaneously might even be a major part of the pedagogy (Silverman, Pietrocola, Nye, Weyer, Osin, Johnson & Weaver, 2012). So long as interactive feedback on each learning task is independent (i.e., feedback on one task does not directly impact the pedagogical state of other tasks), authoring for such tasks can typically be done independently as well.

So then, this is what we mean by a learning task from an authoring standpoint: (1) a task with a distinct pedagogical state, (2) whose dynamics during that task are mainly or wholly derived from the task state, and (3) which includes the actions performed by the learner (or learners). Moreover, as simplifying assumptions, we posit that the pedagogical goals and problem goals remain static for the typical ITS learning task. For example, even in complex training environments, switching the task goals typically implies ending a task and starting a new one. The counterexample to this case would be a learning task specifically targeting the learner's skills at goal-setting or adapting to changing task goals. Such tasks are uncommon and no major authoring tools target such tasks. Finally, we assume that changes to the learner during an ITS task (e.g., learning, affect changes) are primarily influenced by and observable based on interactions with the task interface. If they were not, this would be problematic: the ITS would have little ability to tell if its interventions are effective if some external factors are causing learning and/or task performance (e.g., a second user helping). With that said, such confounds are possible, such as when

multiple users share an ITS intended for one user (Ogan, Walker, Baker, Rebolledo Mendez, Jimenez Castro, Laurentino & de Carvalho, 2012). However, as no known authoring tools develop ITS content for such situations, these are also considered edge cases that we exclude from this analysis authoring learning tasks for ITSS.

A Review of Authoring-Relevant Characteristics of Learning Tasks

Significant literature focuses on taxonomies of learning tasks and the types of knowledge they are designed to convey to a learner. Notable examples include Bloom's Taxonomy and its revisions (Bloom, 1956; Anderson, Krathwohl, Airasian, Cruikshank, Mayer, Pintrich, Raths & Wittrock, 2000), guidelines for learning activities and resources (R. Clark, 2002; R. Clark & Mayer, 2011), and theories of different types of knowledge components involved in learning (Koedinger, Corbett & Perfetti, 2012). These three perspectives each look at a different facet of learning tasks: (1) the task *activity* (Bloom, 1956), (2) the *pedagogical goals* for the learning task (R. Clark, 2002), and (3) the *knowledge components* theorized to encode the knowledge (Koedinger et al., 2012). Figure 2 shows different possible combinations of learning tasks and pedagogical goals.

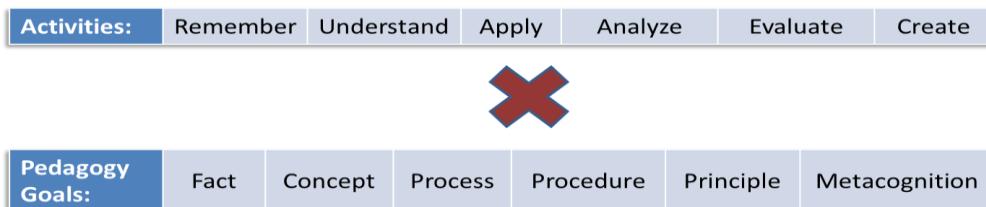


Figure 2: Combinatorial combinations for learning tasks and pedagogical goals

Bloom's taxonomy is the most widely used taxonomy to label learning tasks and has undergone a number of revisions (D. Clark, 2014). Bloom's revised taxonomy (2000) of cognitive knowledge considers six levels: remembering (e.g., list facts), understanding (e.g., summarize in own words), applying (e.g., solve a math problem), analyzing (e.g., identify a statistical trend), evaluating (e.g., select the best-value car for an average consumer), and creating (e.g., build a robot for some task out of parts). Ruth Clark (2002) presented a complementary taxonomy for different types of knowledge associated with pedagogical goals for learning tasks, which built on Merrill (1983). Her categories included facts (unique instances), concepts (classes of instances), processes (representations of how a system works), procedures (steps to reach a task state), and principles (causal relationships and general dynamics).

In addition to these pedagogical goals, metacognitive knowledge can also be a pedagogical goal: where the learner gains understanding or skills to monitor their own cognitive state or learning (Biswas, Jeong, Kinnebrew, Sulcer & Roscoe, 2010; Azevedo, Johnson, Chauncey & Burkett, 2010; Goldberg & Spain, 2014). While metacognitive knowledge may fall into other categories, it can involve learning to monitor an additional information channel other than the task state (i.e., their own mental state). As such, at least some types of metacognitive learning are probably qualitatively different than other types of knowledge (possibly closer to affective or psychomotor skills).

Koedinger et al. (2012) looked at the next step for learning activities, which was the cognitive components relevant to assessment and cognitive encoding of knowledge. They considered four facets for encoding knowledge: the task feature dynamics (static vs. variable), the required learner response (static vs. variable), the relationship between task features (explicit vs. implicit), and the availability of a rationale (e.g., a “why” justification for the relationship between features). These different categorizations

determine when a learner would need to encode, such as a rule (e.g., $y*x = x*y$) or simply an association (e.g., x and y were observed together).

Considering these approaches to categorizing learning activities, key facets emerge for different learning tasks. These fall into three design concerns: pedagogical goals (what the student should learn), task design (the learning environment and its affordances), and task interface (how the task is represented and presented). Together, these concerns constrain the pedagogical interface for how an ITS interacts with learners and what needs to be authored.

Task Dynamics

A major constraint on ITS authoring is the dynamics of the task state itself. For example, some learning tasks are *static* and have *no* dynamic features (e.g., memorizing a shape or a fact). Koedinger et al. (2012) highlighted the distinction between tasks whose features are static (e.g., the same across all instances) versus those that are variable (e.g., some features vary across instances, requiring the learner to generalize across them). We further subdivide variable tasks into a few types, as shown in Figure 3. In our conceptualization, three types of variation can occur in the state of a task. The first type, which we call *variable instance*, is across presented tasks, such as presenting a series of pictures and requiring the learner to identify which ones contain triangles. Other variable tasks change *during* the process of solving the task. The second type, *reactive*, is a task whose state changes due to the learners' actions (e.g., step-by-step equation solving). In the third type, *time-varying*, the task state changes over time regardless of user input (e.g., a video or simulation that unfolds over time). When the task is both time-varying and reactive to user input, we consider it *interactive* (e.g., a 3D game world).

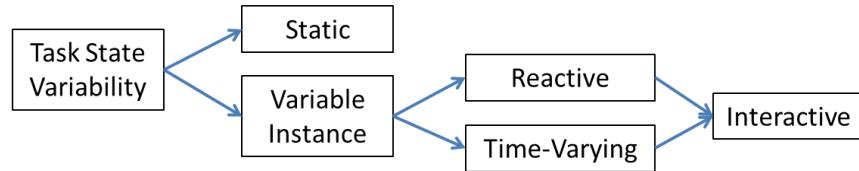


Figure 3: Different types of task variability

These distinctions constrain authoring: static tasks are not typically taught by ITSs at all, because many are rote learning that respond equally well to simpler drill-and-practice methods. However, some intelligent systems, such as Pavlik et al.'s (2007) FaCT system, improve recall-type tasks by optimizing spacing effects and the sequence of instance presentations. Static tasks that do not change based on user input are limited to interventions such as highlighting salient features, demonstrating how to find the right solution, responding to a single answer from the learner, or presenting different tasks (e.g., learning prerequisites). It is still possible to adapt to the learner's responses, such as with systems that provide hints and retry attempts in response to wrong answers on multiple choice questions (Conejo, Guzmán, de la Cruz & Millán, 2006). However, most ITSs tend to focus on reactive and interactive tasks, because learner actions during the task allow a greater ability to target feedback and hints.

Task Assessment

A second major constraint is how well can the ITS measure progress toward pedagogical goals. Since tasks are used to assess learning, measuring progress toward pedagogical goals requires measuring progress toward task goals. In many ways, the ability to measure such progress distinguishes between well-defined and ill-defined domains (Fournier-Viger, Nkambou & Nguifo, 2010; Nye, Bharathy,

Silverman & Eksin 2012). Any task has two possible levels of introspection: the value for the task state and the value of learner actions. When the goals are known, it is often possible to infer the value of actions from the state if the outcomes are predictable, but this is not always (e.g., due to competing goals to choose between). Table 1 categorizes different combinations of knowing the value of states and the value of learner actions. Knowing the value of states allows measuring good outcomes, while knowing the value of actions allows measuring good process.

Table 1: Measures for task goal progress

	Action Values Known	Action Value Unknown
State Utilities Known	Can evaluate all actions and suggest good solution paths <i>Ex. Economic decision tree</i>	Know the value of states, but can't surely deduce what actions do <i>Ex. Stock market simulation</i>
State Transition Gradients Known	Can rank states relatively and can suggest next step or bug fix. <i>Ex. Solving an Algebra problem</i>	Know better/worse changes, but can't suggest concrete actions <i>Ex. Automated essay assessment</i>
State Values Categorical	Can detect good/bad actions, but can't rank improvement <i>Ex. Going out of bounds in a race</i>	Ill-defined task, due to emergent dynamics or subjective values <i>Ex. Building a better world</i>

If a *state utility* function is available, all states and transitions between states have a known value. For a completely measurable task, the relative value of actions is also known, such as a well-formed economics problem where some actions lead to more profit than others. In other cases, the ultimate impact of actions is uncertain (e.g., a chaotic system like the stock market), but good outcomes can still be measured. Generative simulations with emergent behavior often have this quality (Nye et al., 2012).

When an ITS can detect improvements between states but can't evaluate states exactly, then *state transition gradients* are known. So long as the relationship between learner actions and transitions is known (e.g., problem-solving in algebra), formative assessments such as model tracing and example tracing can be used (Aleven et al., 2006). When specific learner actions cannot be evaluated easily (e.g., editing a learner's essay), ITSs can still provide feedback on the task state. Design-based ITSs often use this approach, such as essay-writing ITS that can assess an essay and suggest guidelines to improve the state of the essay (Roscoe & McNamara, 2013). Likewise, when relative value of overall task states is unknown, it is sometimes still possible to assess learner actions. This approach to measurement considers the process, rather than the outcomes. Constraint-based ITSs are often applied to these kind of tasks (Mitrovic, 2003). If it is impossible to assess the quality of either the task state or the learner's actions, the task is ill-defined.

In general, when considering Figure 2, higher levels of Bloom's Taxonomy tend to involve tasks closer to the bottom and right of Table 1. The ability to measure progress on task goals is the first constraint on ITS authoring, since it directly constrains the types of feedback and interventions available to the ITS. If it is impossible to evaluate the quality of actions, it is impossible to provide a "correction" or suggest a concrete "next step" action. As such, there is no need to author one. As such, more complex tasks and generative models tend to offer fewer affordances for authoring traditional ITS content, and tend to rely on the natural dynamics of the simulation to provide reactive feedback (i.e., intelligent environments, rather than intelligent tutors).

Task Interface

The interface of the task consists of how it is presented to the learner and of how the learner presents input. More generally, task interfaces are part of the communication module of a classical four-component ITS diagram (Woolf, 2010). They are the input and output with the learner for the learning task. Typical inputs to an ITS include discrete selections (e.g., multiple choice), continuous selections (e.g., manipulating sliders for a simulation), formal representations (e.g., math equations, graphs), freeform input (e.g., natural language, freehand sketches), and controlling an avatar (e.g., 3D worlds). The modality of learner input is a further constraint on authoring: the pedagogical interface needs to turn input from the task into something actionable. Ironically, this means that feature-rich inputs, such as natural language, are typically simplified into much simpler representations such as discrete selections (e.g., good/bad answers). The representation of the user input is the final major constraint on authoring.

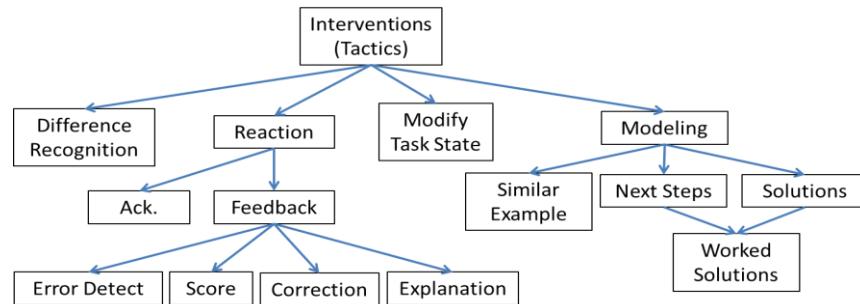


Figure 4: Common interventions for an ITS

Based on the pedagogical features extracted from the task state and user input, the ITS needs to author various interventions. When extending an ITS to new learning tasks, these interventions are typically a major focus for authoring. When, why, and how the ITS applies its pedagogical strategies and tactics is the main repository of an author's domain pedagogy expertise. Figure 4 displays a variety of options for an ITS to intervene during a task. The most rudimentary of these is to *recognize a difference* between a detected state and some other state (e.g., "You seem to like chocolate ice cream more than the average learner."). Since this response assigns no specific value judgment, it can be used for entirely ill-defined domains by using techniques such as novelty detection (Markou & Singh, 2003). It is also possible to *modify the task state or features* even for tasks that lack clear assessments of state or action value, such as through random perturbations. However, typically an ITS changes the state of a task to make it easier or harder (elastic difficulty). This is done by reducing the degrees of freedom (fewer options), completing a task step, or increasing the salience of important task features (e.g., highlighting). Another approach is to *react* in response to user input. Even if no assessment can be made for that input, their input can always be *acknowledged* (Ack.). While this might seem like a weak tactic, it is often used to prompt learners to self-reflect, write in a journal, or mark the start or completion of other metacognitive activities.

On well-defined tasks, ITS tactics often take the form of various types of *feedback* or *modeling* effective solution paths (VanLehn, 2006). Feedback is a response to a user action that either presents an answer or otherwise modifies the task state. Common feedback methods include reacting to errors or good answers (binary assessments), scoring (continuous or ranked assessments), corrections (providing a fix to the answer), and explanations (stating why an answer is right or wrong). Modeling a good answer or solution path is also common. It can be used as feedback or provided at some point during the task state (e.g., provide the worked solution if the user cannot solve the problem). A few types of modeling are possible, including presenting the solution to a similar task example, providing the next step(s) to the current task, or providing a good final solution for the student to look at (e.g., a good essay on the same topic they are

trying to write about). If the full set of steps and the solution is provided, then the intervention was a worked solution.

The authoring effort for these types of feedback varies significantly. Meaningful corrections and explanations require a much deeper connection to domain pedagogy than simpler feedback such as detecting the existence of errors. Likewise, adding explanations to modeling interventions greatly increases authoring effort, because it moves beyond simply assessing task performance and starts to model how a human teacher or instructor might correct student errors or explain the process of working on the task. However, this authoring effort probably supports some of the most effective ITS tutoring behaviors, since it is ideally based on the expertise accumulated from hundreds of hours of human teaching interactions.

Discussion: Common Learning Tasks and Tools

Based on these task features, it is possible to break down a variety of common ITS learning tasks and examine how their distinguishing characteristics are reflected in their authoring tools. Three types of tasks can be considered: well-defined tasks (mostly reactive to user input, values for user actions can be known, user inputs are formal and decidable), less-well-defined tasks (highly interactive, freeform input, lack well-defined goals etc.), and task sandboxes (e.g., complex simulators used to build pedagogical tasks).

Well-Defined Tasks

Table 2 lists common, well-defined learning tasks. Two salient learning activities and pedagogical goals for each class of task are noted. With that said, specific tutors may use different types of scaffolding to use the same task to focus on different pedagogical goals and activities, so there can be significant variation on these. In terms of pedagogy, well-defined tasks probably allow the widest set of interventions: because the task state and user input allow clear assessments and have constrained solution paths, the ITS has a fairly clear view of the task and associated pedagogical state.

The most established ITS tasks center on multi-step problem-solving, such as step-by-step math or physics (Ritter, Anderson, Koedinger & Corbett, 2007; Aleven, McLaren, Sewall & Koedinger, 2006; VanLehn et al., 2005), diagnosing systems and repairing them (Lajoie & Lesgold, 1989), and building dynamical system models (Biswas et al., 2010; Iwaniec, Childers, VanLehn & Wiek, 2014). Step-by-step problem-solving tasks can also be presented in 2D or 3D worlds (Rowe, Shores, Mott & Lester, 2011). Step-by-step problem-solving ITSs tend to author hints and feedback that is conditional on the current task state and the current action (or actions). They also typically provide a full bottom-out worked solution when needed. In-game worked solutions are not common for ITS with avatar input (e.g., 3D worlds), though sometimes recorded cut-screen/video solutions are available. The specific ITS intervention content that is presented is typically tied to general rules that are shared across many task examples (e.g., hints related to the commutative property of addition). As such, authoring these tasks tends to require authoring: (1) A well-defined state representation (e.g., a chess board), (2) a set of domain rules that transform state (e.g., piece move rules), (3) a goal state for the task, (4) a set of expert rules that rely on features of the task state, (5) sometimes buggy rules that represent specific misconceptions to remedy, and (6) templates for feedback and hints that are associated with certain task states or production sequences. In some cases, authoring the task interface is also part of the ITS tool set. The Cognitive Tutor Authoring Tools (CTAT; Aleven et al., 2006) offers an example of fairly mature tools for problem-solving tasks.

Authoring tools for these tasks focus on defining ideal and buggy production rules that can be used to classify task states and learner behavior as they complete the task (Aleven et al., 2006). Ideal production

rules can be used to identify points for positive feedback on a good action or provide hints about good next steps for a problem. These ideal next steps are inferred by evaluating the chains of actions required to reach the task goal (i.e., solution). Similarly, buggy rules can be used to detect specific misconceptions for the learner when they perform certain sequences of actions. Instance-based authoring can be used to infer these rules instead of explicit authoring, through systems such as SimStudent (Matsuda, Cohen & Koedinger, 2014). Feedback and hints tend to be provided through parameterized templates that can refer to task features. A simpler approach to authoring these tasks involves forcing the learner to stay on a linear or simple branching example-tracing approach, with hints that are specific to certain states or transitions in a problem template (Razzaq, Patvarczki, Almeida, Vartak, Feng, Heffernan & Koedinger, 2009). At least for certain mathematics topics, tutoring a single solution strategy (or even a single path) is nearly as effective as a more complex structure (Waalkens, Aleven & Taatgen, 2013; Weitz, Salden, Kim & Heffernan, 2010).

As such, two alternatives exist to rule authoring: (1) instance-based inference and (2) template-specific tutoring. In the first case (e.g., SimStudent), rules are inferred from expert (and perhaps non-expert) solution paths. This requires an authoring tool that shows a complete interface to the problem, as well as an external judgment of the user's expertise level. This allows skipping explicit rule authoring. In the second case, task templates can be authored with tutoring associated with specific task paths. This type of authoring is also used for other tasks (e.g., constrained choice dialogues with branching), so it is a valuable general-purpose authoring interface in its own right. Much like making inferences across multiple instances, an authoring tool for integrating tutoring templates with task paths also needs to give the author a good view of the task state that is similar to the student's view.

Table 2: Common well-defined ITS tasks

Task	Activities (Top-2)	Pedagogy Goal (Top-2)	Variability	State Values	Action Values	Task Inputs	Interventions to Author
<i>Step-By-Step Math</i>	Apply, Understand	Procedure, Principle	Reactive	Gradients/ Ranks	Known	Formal Expression	Feedback (Any), Next Steps, Similar Example, Worked Solution
<i>Diagnosis & Repair</i>	Analyze, Apply	Process, Procedure	Reactive or Interactive	Gradients/ Ranks	Known	Formal Model	Feedback (Any), Next Steps, Similar Example, Worked Solution
<i>Dynamical Systems</i>	Create, Analyze	Procedure, Process	Reactive or Interactive	Utility or Gradients	Known	Formal Model	Feedback (Any), Next Steps, Similar Example, Worked Solution
<i>Classifying</i>	Understand, Analyze	Concept, Process	Static	Category	Known	Discrete Selection	Feedback (Error, Correct, Expl.), Similar Example
<i>Bug Detect</i>	Analyze, Understand	Process, Concept	Static	Category	Known	Discrete/ Continuous Selection	Feedback (Error, Correct, Expl.), Similar Example

Representation Map	Understand, Apply	Concept, Process	Reactive	Gradients/ Ranks	Known	Formal Models	Feedback (Any), Next Steps, Worked Solution
Concept Map Revise	Understand, Analyze	Concept	Reactive	Gradients/ Ranks	Known	Formal Model	Feedback (Any), Next Steps, Worked Solution
Constrained Choice	Analyze, Evaluate	Process, Principle	Reactive or Interactive	Utility or Gradients	Known	Discrete Choice	Feedback (Any)

A second major class of problems includes pattern matching and classification of examples, such as biological taxonomies (Olney et al., 2012), or identifying errors in a complex task, such as bugs in a computer program (Carter & Blank, 2013). These ITSs tend to provide hints and feedback based on the difference in features between the chosen classification and the ideal one, with strong use of explanation but seldom presenting a step-by-step process. A third class of well-defined ITS tasks includes building formal semantic models from freeform representations (e.g., concept map revision) and converting between different well-defined representations, such as from a graph to an equation (Olney et al., 2012). These also tend to keep track of the difference in features between the current and ideal models, but can also suggest next-step changes because the model can be modified.

Authoring tools for these tasks tend to rely on defining ontologies, concept maps, or other structures that define the features of classes and examples. Each instance in a task can be authored by tagging its features or class memberships, after which hints, counter-examples, or other feedback need to be created. If the pedagogy goals also include following a certain step-by-step process to make classification distinctions, authoring may also require tutoring similar to branching example tracing. Across these types of tasks, a simplified ontology class and instance editor could be quite effective, if it provided clear intervention templates to target differences or similarities between patterns.

Finally, there are constrained choice problems, such as ITS-supported multiple choice or branching dialogues (Kim et al., 2009). These can actually be quite varied, but tend to provide interventions that are either dependent only on the current state (e.g., a hint for choosing the wrong answer) or that are exhaustively defined by a branching state path. This means that authoring such tasks tends to be easier up-front than a problem-solving ITS, but harder to reuse for related tasks. In general, authoring these tasks should be similar to linear example-tracing. However, because the tasks may involve fewer general principles that repeat across examples, the authoring is likely to have more explanations and need fewer templates and parameters.

Less-Well-Defined Tasks

Ill-defined and less-well-defined tasks are presented in Table 3. These tasks tend to be less-well-defined because either the goals are not fully defined, the inputs require natural language processing or are otherwise not formally evaluable, or the task requires the learner to produce a full artifact before it can be evaluated. Structured argument tasks, such as those used for law (Pinkwart, Ashley, Lynch & Aleven, 2009) or policy (Easterday & Jo, 2014), work similarly to causal concept map tasks. However, they differ because the goals for argumentation are not always well-defined (i.e., the learner must first choose what to argue). As such, authoring typically requires generating an extensive formal model of free-text sources. This model may be hand-authored or extracted from the associated reference texts. Learners will then need to generate explanations that are logically or causally consistent with the underlying formal model, while supporting the argument goal that the learner has selected. These ITSs tend to also require a reusable set of hint and error-correction templates (e.g., for different logical inconsistencies). For specific common misconceptions, rules or constraints may also be used to trigger explanations or modeling

behavior (e.g., presenting an analogous case or example). Case-based reasoning is one mechanism for identifying similar examples (Kolodner, Cox & González-Calero, 2005) and can also be used as a pedagogical strategy for these domains.

The next category of tasks requires the user to create significant artifacts, such as essays or computer programs (Roscoe & McNamara, 2013; Kumar et al., 2013). These ITSs tend to calculate an overall quality score, based on a number of calculated features that it can highlight or give hints for improvement. However, unlike an ITS for algebra, these tutors cannot explicitly correct most problems (e.g., a programming ITS typically cannot fix the learner's code). ITS authoring for these tasks requires defining a set of features that are used to determine quality of the task artifact. Typically, this training is done using supervised learning or hand-authoring. Tutoring often focuses on feedback and hints related to specific features that need improvement for the artifact, as well as an overall quality score.

Table 3: Common less-well-defined ITS tasks

Task	Activities (Top-2)	Pedagogy Goal (Top-2)	Variability	State Values	Action Values	Task Inputs	Interventions to Author
Structured Argument	Evaluate, Analyze	Principle, Concept	Reactive	Gradients /Ranks	Varies	Formal Model	Feedback (Error, Score, Explain), Similar Example
Functional Coding	Create, Understand	Procedure, Concept	Reactive	Gradients /Ranks	Not Known	Mixed Formal and Freeform	Feedback (Error, Score, Explain), Similar Example
Essay Writing	Create, Analyze	Procedure, Concept	Reactive	Gradients /Ranks	Not Known	Freeform (NLP)	Feedback (Score, Explain), Similar Example
Summaries	Understand	Concept, Process	Reactive	Gradients /Ranks	Not Known	Freeform (NLP)	Feedback (Score, Explain)
Expectation Coverage	Understand, Analyze	Concept, Principle	Interactive	Gradients /Ranks	Known	Freeform (NLP)	Feedback (Any), Next Steps, Worked Solution
Short Answer	Understand, Analyze	Concept, Fact	Reactive/ Interactive	Gradients /Ranks	Known	Freeform (NLP)	Feedback (Any), Similar Example
Open Self-Reflection	Understand	Concept, Process	Static	Not Known	Not Known	Freeform	Difference Recog., Acknowledge
Choice Search	Evaluate, Analyze	Process, Principle	Interactive	Utility or Gradients	Not Known	Freeform or Avatar	Feedback (Any), Similar Example
Setting Goals and Priorities	Evaluate, Analyze	Principle, Process	Interactive	Not Known	Not Known	Varies (Formal or Freeform)	Difference Recog., Similar Example

The next set of less-well-defined ITS focus on helping the learner understand, analyze, and evaluate information. They include self-reflection, expectation coverage tasks (Graesser, Chipman, Haynes & Olney, 2005), summarization and paraphrasing tasks (McNamara, Levinstein & Boonthum, 2004), and short-answer tasks. All of these tasks focus on helping the learner understand semantic content and its relationships. Open self-reflection tasks focus on the metacognitive practice of reflecting on the content. As such, in many cases the quality of content is not assessed (e.g., a journaling task). Instead, ITS content focuses on encouraging the habit of self-reflection. In general, many metacognitive tutors focus on building habits, such as encouraging question-asking or hint button use (Azevedo et al., 2010; Roll, Aleven, McLaren & Koedinger, 2011). Open self-reflection tasks and other content-agnostic tasks tend to require little content authoring, and often require only a set of simple prompt and acknowledgement templates. These reflection prompts can be triggered by task events or even by general timers.

Expectation coverage tasks unfold over many dialogue turns, which assume a part-whole relationship for multiple expectations as part of a full explanation. As such, these ITS must detect multiple related subtopics and provide feedback and hints on each one. Short answer tasks are even more constrained, and their answers tend to be binned into good answers, specific misconceptions, or general bad answers. Expectation coverage tasks tend to contain short answer tasks inside of them, when specific knowledge needs to be assessed. A variety of authoring representations exist for evaluating semantic statements, which fall into three main categories: instance-based authoring, feature authoring, and grammar-based authoring. Instance-based authoring involves generating various classes of answers (i.e., good/bad), which are then used to match against using various algorithms. Feature-based authoring involves creating special features, such as regular expressions or keywords that capture key defining features between different types of answers. Finally, grammar-based authoring involves developing domain-specific parsers that extract domain-relevant relationships from the text.

In all cases, these techniques are used to bin learner answers into specific speech act categories, which can then be associated with feedback, hints, or modeling interventions. Summarization tasks work similarly, but require the learner to rephrase a passage. A successful summary requires the answer to have similar semantics, but dissimilar surface features (i.e., it cannot be identical). These tend to focus on understanding the content, but their quality tends to be rated on a continuous scale, because there are competing feature sets. In addition to assessments of learner input, rules are also required to allow the dialogue to progress naturally. In general, a limited set of templates can be sufficient to handle typical ITS tasks. While there are some indications that different levels of knowledge might benefit from different dialogue interactions (Nye et al., 2014), similar logical rules for managing dialogue interactions can cover a variety of domains.

Task Sandbox Environments

As a final task category, open-ended searches and decision-points for choices are common, particularly in virtual worlds and scenario-based learning. These include looking for a satisfactory or optimal set of actions to some learning task. In many cases, the action sets vary by context and are not known *a priori*. If the quality of choices can be ranked or their component features ranked, the ITS can provide feedback, hints, and explanations about the quality of actions (Kim et al., 2009; Sotilare, Goldberg, Brawner & Holden, 2012). However, for a simulation, this information may only be available after the completion of a scenario. The next level of complexity occurs when the task goals are not fully defined, but must be set or prioritized by the learner. For subjective or “wicked” tasks where actions change how goals are understood, goal selection tasks are almost unavoidable (Nye et al., 2012). This tends to occur almost exclusively in simulations or design tasks, where defining and monitoring goals are a major part of the tasks and learning content. These tasks tend to be very hard to tutor directly and sometimes rely on detecting certain common or uncommon patterns, which are then brought to the attention of the learner.

For complex simulations, many current pedagogical methodologies focus on after-action review procedures. These tend to include a mixture of artifact evaluation (i.e., considering metrics collected from a simulation run) and self-reflection. After-action reviews have historically been facilitated by a human-in-the-loop and are geared toward focused reflection and knowledge elicitation. The underlying task consists of a series of choices and decision points in a specific scenario, which are translated to overarching learning objectives. These choices are then considered similarly to other types of learning tasks, such as following procedural rules while receiving feedback about deviations from desired performance.

Rather than being more complicated to author, the pedagogy for complex choice tasks is often as simple (or simpler) than highly constrained domains such as mathematics. This is because well-defined domains give many opportunities for clear pedagogical interventions: the state of the task is fully known, completely based on the learner's inputs, and allows immediate feedback. By comparison, a game-based task requires game messages that are sent to assessment models that infer the pedagogical state. As a result, ITS authoring is limited to the data made available by a task interface that was not originally intended to offer pedagogically useful assessments (e.g., a 3D game engine). As such, an additional authoring layer needs to convert the raw task state into a much more pedagogical state. This requires an operational task analysis and authoring tools that transform various task events into pedagogically useful assessments. This extra assessment layer makes complex environments more difficult to author, which ultimately limits the interventions that can be authored (e.g., hints and feedback).

While serious games and simulation-based training environments can alleviate this problem in the development phase, many do not. In fact, simulation-based training solutions have increasingly moved toward commercial and open-source game engines to reduce production costs, such as Virtual Battle Space 3 (VBS3), Unity, and the Unreal Game Engine. These sandbox authoring environments enable developers to build complex task scenarios for both individuals and collaborative/team-based interactions. However, the data generated by these systems follow generic protocols for distributed delivery, such as distributed interactive simulation (DIS) and high-level architecture (HLA) that lack any concept of pedagogy or semantics (Hofer & Loper, 1995; Kuhl, Dahmann & Weatherly, 2000).

The best solution so far to this problem has been to explicitly build a layer of metrics onto the task environment, which are then consumed by the ITS as its pedagogical state. Basically, a simpler task state is constructed from features in the task sandbox, which is then linked to assessments. For example, Generalized Intelligent Framework for Tutoring (GIFT) provides a generalized architecture that can consume game-message traffic and use this to infer pedagogical conditions linked to a concept hierarchy (Sottilare et al., 2012). Much of the data captured associate with entity state (i.e., avatars, non-player characters, weapons, machines, vehicles, etc.) location, movement, and action. In short, much of the task state is too low-level or downright irrelevant to ITS behavior. A subset of these data are continuously communicated to GIFT and routed to the domain knowledge file (DKF) for managing assessment practices. The DKF is where an author structures: the concept hierarchy associated with a set of tasks, how data are integrated into a concept assessment, and how those data are managed at runtime. Assessments can be authored directly within a DKF or it can be supported by an external assessment engine, such as the Student Information Models for Intelligent Learning Environments (SIMILE; Goldberg, 2013), where the DKF acts by routing data to the appropriate concept assessments (Figure 5).

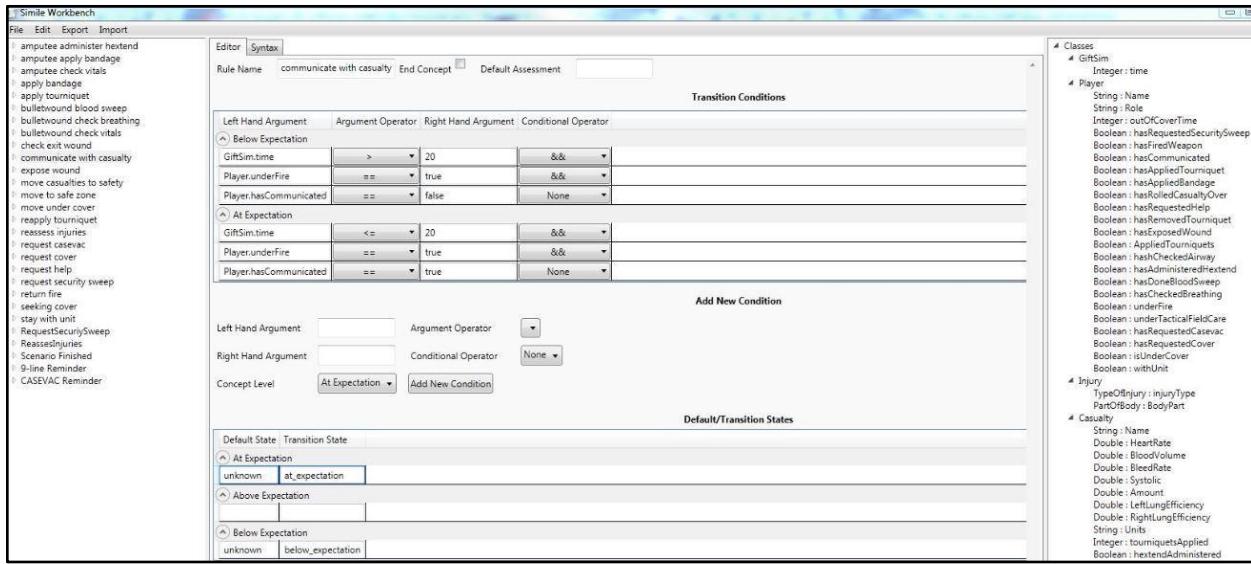


Figure 5. SIMILE workbench with authored assessments for vMedic

However, the reverse direction (i.e., offering specific interventions) has the same complications. Conditions need to consider both the real-time performance and user intention, as well as the possible actions that are available to the user (which must also be relayed to the ITS, to enable suggestions). In GIFT’s current use-cases, this information includes tagged locations for the user’s position in the environment, the set of entities and objects that are around the user, what entities the user can currently observe, the actions available, and the timers related to task execution.

For example, consider the task of “maintaining cover” while patrolling a compound, which requires time, location, and entity state data. Waypoints and areas of interest are defined around the compound wall so that the player can be tracked to monitor patrol progress. An author can then define assessments based on if the user has reached certain waypoints within a specific timeframe. In addition, if a scenario author determines that a user should adjust their entity’s state within specific areas of interest, such as adjusting their stance from standing to kneeling due to a wall being low, then the author can associate assessments to inform student action in relation to performance criteria. By knowing this context, it is also possible to deliver real-time feedback based on the actions and current assessment information.

To further complicate the issue, these types of interactive environments are excellent for collaborative and team-based learning events. From the ITS perspective, this requires additional modeling dependencies that associate interaction and intention with team oriented skills and attributes. While there is extensive literature on what makes effective teams and effective team training approaches (Salas et al., 2008), how to establish these practices in an automated fashion is a challenge. Beyond modeling individualized tasks and how interaction in a virtual environment can infer competencies, additional assessments must analyze group-level data that is aggregated across a set of users. These assessments include team cohesion, trust, communication, and shared cognition. While this field is a wide-open research area, architectures like GIFT must be designed to facilitate the type of modeling techniques that are based on trends across users rather than within users. In terms of authoring, the challenge is taking available data and translating them to team-based inferences that can designate performance across a set of concepts. In addition, how to react to these assessments needs to be explored, such as how interventions are handled and how they are communicated to a team.

Recommendations and Future Research

Across this book, examples and lessons learned for authoring each type of learning task are discussed. By identifying common learning tasks in ITSs, it should be possible to develop general authoring interfaces that make authoring for each type of task intuitive and effective. In some cases, highly effective authoring models already exist and might serve as exemplars for task-specific authoring tools in generalized ITSs such as GIFT. Ideally, these authoring tools should collect information in ways that are familiar to instructors and other domain pedagogy experts. Form-based authoring, example-based authoring, and supervised tagging are all reasonable approaches that are particularly attractive.

However, there are also learning tasks that do not yet have well-established techniques that allow non-technical domain experts to easily author content. For example, authoring real-time assessments for complex tasks remains more of an art than a science. At least some of this authoring involves mapping simulation or virtual world events to pedagogical features. For this type of authoring, even if game worlds had integrated pedagogical tools, a tool to easily map raw simulation data to metrics may be hard to use by the domain experts, even if it is well designed. Similarly, authoring for ITS tasks with multiple learners is a poorly understood area. For example, team-based tutoring requires assessment and intervention at multiple levels (e.g., individual and group). Further research on such tasks and exploration of different types of authoring approaches may be needed before good examples of such authoring tools become clear.

References

- Aleven, V., McLaren, B. M., Sewall, J. & Koedinger, K. R. (2006). The cognitive tutor authoring tools (CTAT): Preliminary evaluation of efficiency gains. In M. Ikeda, K. D. Ashley & T. Chan (Eds.) *Intelligent Tutoring Systems (ITS) 2006* (pp. 61-70). Springer Berlin Heidelberg.
- Azevedo, R., Johnson, A., Chauncey, A. & Burkett, C. (2010). Self-regulated learning with MetaTutor: Advancing the science of learning with MetaCognitive tools. In M. S. Khine & I. M. Saleh (Eds.) *New Science of Learning* (pp. 225-247). Springer New York.
- Biswas, G., Jeong, H., Kinnebrew, J. S., Sulcer, B. & Roscoe, R. D. (2010). Measuring Self-Regulated Learning Skills through Social Interactions in a teachable Agent Environment. *Research and Practice in Technology Enhanced Learning*, 5(2), 123-152.
- Carter, E. & Blank, G. D. (2013). An Intelligent Tutoring System to Teach Debugging. In H. C. Lane, K. Yacef, J. Mostow & P. Pavlik (Eds.) *Artificial Intelligence in Education (AIED) 2013* (pp. 872-875). Springer Berlin Heidelberg.
- Clark, D. (2014). Bloom's taxonomy of learning domains. Retrieved August, 26, 2014.
- Clark, R. C. (2002). Applying cognitive strategies to instructional design. *Performance Improvement*, 41(7), 8-14.
- Clark, R. C. & Mayer, R. E. (2011). *E-learning and the science of instruction: Proven guidelines for consumers and designers of multimedia learning*. John Wiley & Sons.
- Conejo, R., Guzmán, E., de-la Cruz, J. L. P. & Millán, E. (2006). An empirical study about calibration of adaptive hints in web-based adaptive testing environments. In V. Wade, H. Ashman & B. Smyth (Eds.) *Adaptive Hypermedia and Adaptive Web-Based Systems* (pp. 71-80). Springer Berlin Heidelberg.
- Easterday, M. W. & Jo, I. Y. (2014). Replay Penalties in Cognitive Games. In S. Trausan-Matu, K. Boyer, M. Crosby & K. Panourgia (Eds.) *Intelligent Tutoring Systems (ITS) 2014* (pp. 388-397). Springer Berlin Heidelberg.
- Fournier-Viger, P., Nkambou, R. & Nguifo, E. M. (2010). Building intelligent tutoring systems for ill-defined domains. In R. Nkambou, R. Mizoguchi & J. Bourdeau (Eds.) *Advances in Intelligent Tutoring Systems* (pp. 81-101). Springer Berlin Heidelberg.
- Goldberg, B. & Spain, R. (2014). Creating the Intelligent Novice: Supporting Self-Regulated Learning and Metacognition in Educational Technology. In R. Sottilare, A. Graesser, X. Hu, and B. Goldberg (Eds.) *Design Recommendations for Intelligent Tutoring Systems, Vol. 2: Instructional Management* (pp. 109-134). U.S. Army Research Laboratory.

- Goldberg, B. (2013). *Explicit Feedback Within Game-Based Training: Examining the Influence of Source Modality Effects on Interaction*. Ph.D., University of Central Florida.
- 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.
- Hofer, R. C. & Loper, M. L. (1995). DIS today [Distributed interactive simulation]. *Proceedings of the IEEE*, 83(8), 1124-1137.
- Iwaniec, D. M., Childers, D. L., VanLehn, K. & Wiek, A. (2014). Studying, teaching and applying sustainability visions using systems modeling. *Sustainability*, 6(7), 4452-4469.
- Kim, J. M., Hill, Jr, R. W., Durlach, P. J., Lane, H. C., Forbell, E., Core, M., ... & Hart, J. (2009). BiLAT: A game-based environment for practicing negotiation in a cultural context. *International Journal of Artificial Intelligence in Education*, 19(3), 289-308.
- Koedinger, K. R., Corbett, A. T. & Perfetti, C. (2012). The Knowledge-Learning-Instruction framework: Bridging the science-practice chasm to enhance robust student learning. *Cognitive Science*, 36(5), 757-798.
- Kolodner, J. L., Cox, M. T. & González-Calero, P. A. (2005). Case-based reasoning-inspired approaches to education. *The Knowledge Engineering Review*, 20(03), 299-303.
- Kuhl, F., Dahmann, J. & Weatherly, R. (2000). *Creating computer simulation systems: an introduction to the high level architecture*. Prentice Hall PTR Upper Saddle River.
- Kumar, A. N. (2013). Using Proplets for problem-solving exercises in introductory C++/Java/C# courses. In *IEEE 2013 Frontiers in Education Conference* (pp. 9-10). IEEE Press.
- Lajoie, S. P. & Lesgold, A. (1989). Apprenticeship Training in the Workplace: Computer-Coached Practice Environment as a New Form of Apprenticeship. *Machine-Mediated Learning*, 3(1), 7-28.
- Markou, M. & Singh, S. (2003). Novelty detection: a review- part 1: Statistical approaches. *Signal Processing*, 83(12), 2481-2497.
- Matsuda, N., Cohen, W. W. & Koedinger, K. R. (Online First). Teaching the Teacher: Tutoring SimStudent Leads to More Effective Cognitive Tutor Authoring. *International Journal of Artificial Intelligence in Education*, 1-34.
- McNamara, D. S., Levinstein, I. B. & Boonthum, C. (2004). iSTART: Interactive strategy training for active reading and thinking. *Behavior Research Methods, Instruments & Computers*, 36(2), 222-233.
- Merrill, M. D. (1983). Component Display Theory. In C. M. Reigeluth (Eds.), *Instructional Design Theories and Models: An Overview of their Current States* (279-333). Hillsdale, NJ: Lawrence Erlbaum.
- Mitrovic, A. (2003). An intelligent SQL tutor on the web. *International Journal of Artificial Intelligence in Education*, 13(2), 173-197.
- Nye, B. D., Bharathy, G. K., Silverman, B. G. & Eksin, C. (2012). Simulation-Based training of ill-defined social domains: the complex environment assessment and tutoring system (CEATS). In S. A. Cerri, W. J. Clancey, G. Papadourakis & K. Panourgia (Eds.) *Intelligent Tutoring Systems (ITS) 2012* (pp. 642-644). Springer Berlin Heidelberg.
- 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.
- Nye, B. D., Rahman, M. F., Yang, M., Hays, P., Cai, Z., Graesser, A. & Hu, X. (2014). A tutoring page markup suite for integrating Shareable Knowledge Objects (SKO) with HTML. In *Intelligent Tutoring Systems (ITS) 2014 Workshop on Authoring Tools*, (pp. 1-8). CEUR.
- Ogan, A., Walker, E., Baker, R. S., Rebollo Mendez, G., Jimenez Castro, M., Laurentino, T. & de Carvalho, A. (2012). Collaboration in Cognitive Tutor use in Latin America: Field study and design recommendations. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 1381-1390). ACM.
- Olney, A. M., D'Mello, S., Person, N., Cade, W., Hays, P., Williams, C., ... & Graesser, A. (2012). Guru: A computer tutor that models expert human tutors. In S. A. Cerri, W. J. Clancey, G. Papadourakis & K. Panourgia (Eds.) *Intelligent Tutoring Systems (ITS) 2012* (pp. 256-261). Springer Berlin Heidelberg.
- Pavlik Jr., P. I., Presson, N., Dozzi, G., Wu, S., MacWhinney, B., Koedinger, K. R. (2007). The FaCT (Fact and Concept Training) System: A New Tool Linking Cognitive Science with Educators. In McNamara, D., Trafton, G. (eds.) *Proceedings of the Twenty-Ninth Annual Conference of the Cognitive Science Society*, pp. 397-402. Lawrence Erlbaum: Mahwah.
- Pinkwart, N., Ashley, K., Lynch, C. & Aleven, V. (2009). Evaluating an intelligent tutoring system for making legal arguments with hypotheticals. *International Journal of Artificial Intelligence in Education*, 19(4), 401-424.

- Razzaq, L., Patvarczki, J., Almeida, S. F., Vartak, M., Feng, M., Heffernan, N. T. & Koedinger, K. R. (2009). The Assistant Builder: Supporting the life cycle of tutoring system content creation. *IEEE Transactions on Learning Technologies*, 2(2), 157-166.
- 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.
- Roscoe, R. D. & McNamara, D. S. (2013). Writing pal: Feasibility of an intelligent writing strategy tutor in the high school classroom. *Journal of Educational Psychology*, 105(4), 1010.
- 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.
- Rowe, J. P., Shores, L. R., Mott, B. W. & Lester, J. C. (2011). Integrating learning, problem solving, and engagement in narrative-centered learning environments. *International Journal of Artificial Intelligence in Education*, 21(1), 115-133.
- Salas, E., DiazGranados, D., Klein, C., Burke, C. S., Stagl, K. C., Goodwin, G. F. & Halpin, S. M. (2008). Does team training improve team performance? A meta-analysis. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 50(6), 903-933.
- Silverman, B. G., Pietrocola, D., Nye, B., Weyer, N., Osin, O., Johnson, D. & Weaver, R. (2012). Rich socio-cognitive agents for immersive training environments: case of NonKin Village. *Autonomous Agents and Multi-Agent Systems*, 24(2), 312-343.
- Sottilare, R. A., Goldberg, B. S., Brawner, K. W. & Holden, H. K. (2012). A modular framework to support the authoring and assessment of adaptive computer-based tutoring systems (CBTS). In *Interservice/Industry Training, Simulation and Education Conference (I/ITSEC) 2012*.
- VanLehn, K. (2006). The behavior of tutoring systems. *International Journal of Artificial Intelligence in Education*, 16(3), 227-265.
- 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.
- Waalkens, M., Aleven, V. & Taatgen, N. (2013). Does supporting multiple student strategies lead to greater learning and motivation? Investigating a source of complexity in the architecture of intelligent tutoring systems. *Computers & Education*, 60(1), 159-171.
- Weitz, R., Salden, R. J., Kim, R. S. & Heffernan, N. T. (2010). Comparing worked examples and tutored problem solving: Pure vs. mixed approaches. In S. Ohlsson & R. Catrambone (Eds.) *Proceedings of the Thirty-Second Annual Meeting of the Cognitive Science Society* (pp. 2876-2881).
- Woolf, B. P. (2010). *Building intelligent interactive tutors: Student-centered strategies for revolutionizing e-learning*. Morgan Kaufmann.

SECTION II

AUTHORING MODEL- TRACING TUTORS

Xiangen Hu, Ed.

CHAPTER 5 A Historical Perspective on Authoring and ITS: Reviewing Some Lessons Learned

Benjamin D. Nye¹ and Xiangen Hu^{1,2}

¹University of Memphis, ²China Central Normal University

Introduction

This section discusses the practices and lessons learned from authoring tools that have been applied and revised through repeated use by researchers, content authors, and/or instructors. All of the tools noted in this section represent relatively mature applications that can be used to build and configure educationally effective content. Each tool has been tailored to address both the tutoring content and the expected authors who will be using the tool. As such, even tools which support similar tutoring strategies may use very different interfaces to represent equivalent domain knowledge. In some cases, authoring tools even represent offshoots where different authoring goals led to divergent evolution of both the authoring tools and the intelligent tutoring systems (ITSs) from a common lineage. Understanding how these systems adapted their tools to their particular authoring challenges gives concrete examples of the tradeoffs involved for different types of authoring. By reviewing the successes and challenges of the past, the chapters in this section provide lessons learned for the development of future systems.

Authoring Tools for Adaptive and Data-Driven Systems

In general, for ITS authoring tools, discussion often centers on tools for creating content, such as new problems or new dialogues that interactively help the learner step-by-step. While these are a key part of the authoring process, mature authoring tools tend to cover a wider array of authoring and configuration options. These activities range from small activities like selecting HTML pages to larger tasks such as manually selecting or sequencing curriculum topics. In other cases, the problem is not so much authoring as versioning: maintaining and updating content in a reliable way. Within this section, all of these activities are considered as facets of the larger authoring lifecycle.

This lifecycle typically includes the following steps:

- (1) Creating initial content module (e.g., a problem),
- (2) Interacting with module like a student,
- (3) Revising the module,
- (4) Selecting and composing modules for inclusion in a given curriculum,
- (5) Collecting data on student interaction, and
- (6) Revising module based on collected data.

From the standpoint of content quality, each of these steps contributes to development of effective tutoring and learning. Efficient tools for certain stages of this lifecycle may be less effective for other stages. For example, while a series of simple may be efficient for entering the initial content, that same interface would not necessarily make it easy to find and correct a specific field during the revision step.

As such, all systems must make choices about the authoring activities that receive the most support, often based on the types of expected authors. With this in mind, the chapters in this section describe a variety of approaches to authoring.

In Chapter 6, Blessing, Aleven, Gilbert, Heffernan, Matsuda, and Mitrovic discuss different approaches to “Authoring Example-based Tutors for Procedural Tasks.” This chapter discusses the convergence of multiple lines of authoring tools for step-based problem solving tutors toward example-based authoring. Example-based authoring, also sometimes called instance-based authoring, provides an interface where the author builds tutoring content and student support (e.g., hints) for an individual example or limited class of parameterized examples. By comparison, traditional authoring techniques often required implementing a full set of explicit domain rules. A number of advantages for such tutors are provided, which are evident in the authoring tools presented. For some systems, such as ASSISTments and Cognitive Tutor Authoring Tools (CTAT), this approach was chosen to lower barriers to authoring so that instructors could develop ITS content. For other systems, such as the Extensible Problem-Solving Tutor (xPST), the approach allows tightly integrating tutoring with a wide variety of content, ranging from 3D games to web pages. Finally, in systems such as Authoring Software Platform for Intelligent Resources in Education (ASPIRE) and SimStudent, algorithms are used to generalize domain rules and constraints that enable the ITS to tutor a wider variety of problems than were explicitly authored. Particularly since domain content experts are much more likely to be able to author examples than create formal representations of their rules, this approach is appealing for well-defined procedural tasks.

In Chapter 7, Matuk, Linn, and Gerard describe the authoring capabilities of the Web-based Inquiry Science Environment (WISE) system. While WISE does not currently focus on adaptive elements, the system has a strong focus on both theory-based (the knowledge-integration framework) and data-driven development and revision of content. This system demonstrates the potential reach of a well-designed system designed around teachers, with over 10,000 teachers registered to use WISE. Their main principles are to provide tools accommodate a range of abilities, allow users to reuse, revise, and extend what others have made, reporting student data as evidence to inform revision, and allowing flexibility for authors to repurpose the system for their goals. Compared to many authoring systems, WISE strongly supports later parts of the authoring lifecycle (i.e., selecting content and data-driven revision).

In Chapter 8, Jacovina, Snow, Dai, and McNamara describe the authoring tools for iSTART-2 and Writing Pal. These systems use natural language processing techniques to support reading comprehension strategies and essay-writing skills, respectively. Authoring tools within these systems are novel in a few ways. First, the tools explicitly contain distinct features that are intended for researchers (e.g., randomizing the use of a certain feedback strategy) versus for teachers (e.g., modifying or selecting content). In general, authoring in these systems attempts to mirror the student experience with the system but with buttons to edit content or behavior. Second, the tools are being designed to allow authoring behavior that is associated with stealth assessments, such as feedback or experimental activities. Compared to other systems in this section, this work explores the potential for collecting and applying rich metrics on student behavior (e.g., the narrativity of a student’s essays).

In Chapter 9, Charlie Ragusa outlines the design principles of the Generalized Intelligent Framework for Tutoring (GIFT) authoring tools, which are currently being used by multiple groups to integrate tutoring into environments as varied as 3D worlds and PowerPoint presentations. A major focus of this chapter is the need and development of collaborative authoring tools: frameworks that allow multiple authors with complementary expertise to contribute effectively. These processes are essential, since the knowledge needed to author an ITS tends to be spread across multiple experts.

Finally, in Chapter 10, Steve Ritter describes practices related to authoring and refining ITS content across the lifecycle of a commercial product, based on practices used by the widely used Cognitive Tutor

system. This chapter focuses significantly on methods to leverage student data to improve an ITS over time. The discussion revolves around the types of changes that are often necessary (e.g., parameters, design of the tasks, content) and methods to determine the changes (e.g., manually, automatically calculated, crowdsourced). Versioning issues are noted with data-driven models, such as data becoming less-applicable if the design of the task has changed. Also, suggestions are made for which types of changes are best suited for certain methods (e.g., certain parameter changes can be automatically rolled out). These issues reflect the realities of balancing data-driven design with a regularly-used product that must also behave reliably for users on a day-to-day basis.

Themes and Lessons Learned

Across these chapters, some common themes emerged for systems that have matured to reach wider user bases. Strong themes included the following:

- (1) User-Centric Design: Authoring tools that are tailored for the specific authors who are intended to use them. In some cases, building multiple tools that serve qualitatively different types of authors. Both systems with wide user bases of authors (ASSISTments and WISE, both with >1k teachers) strongly focused on serving the common needs of teachers, which include being able to modify and add content. This was also a significant theme for multiple other systems (e.g., iSTART-2).
- (2) Workflows: In some cases, multiple tools and qualitatively different approaches are used to build, refine, and enhance different parts of a system. The GIFT discussion focuses extensively on collaborative authoring. The Cognitive Tutor product lifecycle discussion also describes a multi-faceted authoring process.
- (3) Constraints: Authoring tools constrain the author (by design). For each of the systems with large student user bases (Cognitive Tutor, ASSISTments, and WISE, all with > 75k students), authoring and configuration was often significantly constrained. In many cases, this was to simplify the authoring process. However, systems may also attempt to limit certain types of configurations or authoring that are not pedagogically sound within the system. This raises the issue that sometimes the options that are not given for authoring can be as important as those that are.
- (4) Content vs. Adaptivity: Different authoring tools and processes emphasize different parts of the content authoring cycle, with systems for teachers tending to support simple content creation revision (WISE, iSTART-2 for teachers, ASSISTments) and systems with stronger use by the research community providing more tools for training step-based adaptivity (CTAT, SimStudent, GIFT, ASPIRE).
- (5) What You See Is What You Get (WYSIWYG): Nearly all of the systems in this chapter describe methods to quickly view the content after it is authored, incrementally and iteratively (CTAT, SimStudent, xPST, ASSISTments, iSTART-2, WISE, ASPIRE). By allowing authors to see what they are creating in real time, these tools enable a more direct authoring process.
- (6) Generalization Algorithms: While some of these systems use complex formal representations (e.g., ontologies, production rules), the field has taken steps toward authoring using examples. As such, research on methods to identify general principles or rules from examples has become an important topic (SimStudent, ASPIRE).

(7) Versioning and Maintaining Content: For systems with large user bases, these chapters touched on the complexities and advantages of maintaining a large system, such as supporting modified content, tracking its evolution, and retaining only content with signs of effectiveness evident in the student data (Cognitive Tutor and WISE).

Based on these lessons learned, a few areas of focus emerge. First, support for example-based authoring and other WYSIWYG approaches is probably essential to help instructors author new ITS-tutored activities. Second, collecting and presenting centralized data about an existing repository of tutoring modules (such as GIFT's domain knowledge files) could significantly improve the ability and confidence of authors trying to select tutoring for an activity. These data could also be used for versioning that tracks, maintains, and prunes the set of recommended tutoring modules over time (an issue that is explored in Chapter 6). Finally, this work implies that multiple authoring interfaces are needed to support the research community versus instructors. With these shifts, GIFT could expand its user base and also increase the effectiveness of content over time. More generally, these are lessons that authoring tools for ITS and other learning technologies should follow to ensure that their systems are easier to author, effective for learners, and can be revised and maintained over time.

CHAPTER 6 Authoring Example-based Tutors for Procedural Tasks

Stephen B. Blessing¹, Vincent Aleven², Stephen B. Gilbert³, Neil T. Heffernan⁴,

Noboru Matsuda², Antonija Mitrovic⁵

¹ University of Tampa; ² Carnegie Mellon University; ³ Iowa State University;

⁴ Worcester Polytechnic Institute; ⁵ University of Canterbury

Introduction

Researchers who have worked on authoring systems for intelligent tutoring systems (ITSs) have examined how examples may form the basis for authoring. In this chapter, we describe several such systems, consider their commonalities and differences, and reflect on the merit of such an approach. It is not surprising perhaps that several tutor developers have explored how examples can be used in the authoring process. In a broader context, educators and researchers have long known the power of examples in learning new material. Students can gather much information by poring over a worked example, applying what they learn to novel problems. Often these worked examples prove more powerful than direct instruction in the domain. For example, Reed and Bolstad (1991) found that students learning solely by worked examples exhibited much greater learning than those learning instruction based on procedures. By extension then, since tutor authoring can be considered to be teaching a *tabula rasa* tutor, tutor authoring by use of examples may be as powerful as directly programming the instruction, while being easier to do.

Several researchers have considered how examples may assist programmers in a more general sense (e.g., Nardi, 1993; Lieberman, 2001). This approach, referred to as “programming by example” or “programming by demonstration,” generally involves the author programmer demonstrating the procedure in the context of a specific example and then the system abstracting the general rules of the procedure on the basis of machine learning or other artificial intelligence (AI) techniques. The balance in such systems is between its ease of use versus its expressivity. A system may be easy to use, but lack expressive power and thus generality. At the other extreme (e.g., a general-purpose programming language), a system can be very expressive and thus generalizes to new situations readily, but lacks ease of learning. Of course, as an author gets more used to a tool, regardless of initial complexity, the tool becomes easier. The balance between ease of use and expressivity lies with tutor authoring tools as much as it does in the more general case of programming by example.

Some researchers who build authoring systems for ITSs have leveraged this general approach, using examples as a major input method for the ITS. Five such systems are discussed here: Authoring Software Platform for Intelligent Resources in Education (ASPIRE), ASSISTments, Cognitive Tutor Authoring Tools (CTAT), SimStudent, and the Extensible Problem-Solving Tutor (xPST). All of these systems use examples in at least some important aspect of tutor creation. A main goal in using examples is to ease the authoring burden, to both speed up the authoring of ITSs and enable authoring for a wider variety of people. All five systems build tutors for procedural-type tasks, where each step of the task is reasonably well defined and student answers tend to be easily checked. The tutors built by these systems have been deployed in a wide variety of such tasks (e.g., math, chemistry, genetics, statistics, and manufacturing, to name a few). However, some of the systems can also tutor on non-procedural tasks (e.g., ASPIRE). The type of tutoring interaction mediated by these tutors is typically in the pattern of constraint-based and model-tracing tutors. That is, each student step is checked for correctness, with help and just-in-time messages available.

A short description of each of these five systems follows. After these discussions, the general implications for such an example-based method for tutor creation conclude the chapter.

The Authoring Systems

ASSISTments

ASSISTments is a web-based tutoring system started from work on CTAT (discussed below), and developed at both Carnegie Mellon University (CMU) and Worcester Polytechnic Institute (WPI). It is a platform, hosted at WPI, which allows sharing of content between teachers. The platform is domain neutral. ASSISTments gives students problems, and there are content libraries for many disparate subjects including mathematics, statistics, inquiry-based science, foreign language, and reading, but 90% of the content is in mathematics. Each item, or ASSISTment, consists of a main problem and the associated scaffolding questions, hints, and buggy messages.

Early work on this system (circa 2004) required programmers to build content, but soon this was untenable, so a graphical user interface (GUI)-based authoring tool was developed to enable other people, such as teachers and other researchers, to create content in quantity. Figure 1 shows the tutor and authoring screens for the same problem (Razzaq et al., 2009). Somewhere around 2011, the total amount of content created by non-WPI personnel began to outnumber that created by WPI personnel.

The figure displays the ASSISTments interface with several components:

- Content Developer View:** Shows the problem details (PRAEXE - Item 19 G-2003(Congruent triangles)) and a sidebar with scaffolding strategies like "Scaffolding strategy Scaffold".
- Student View:** Shows the main problem statement: "Triangles ABC and DEF are congruent. The perimeter of triangle ABC is 23 inches. What is the length of side DF in triangle DEF?" Below it, two triangles are shown: triangle ABC with sides labeled x, 8 inches, and 2x; and triangle DEF with all sides labeled x. A red box highlights the answer input field where '6' is entered.
- The Main Problem:** Labels the central problem area.
- 1st Scaffold:** Shows a scaffolded hint: "Which side of triangle ABC has the same length as side DF of triangle DEF?" Below it, a yellow box states: "Congruent triangles means triangles whose corresponding sides are equal in length."
- Hint:** Shows a hint: "Select one: AB, BC, AC". Below it, a red box states: "Side AB corresponds to side DE of triangle DEF, not DF. Try again, please." A link "Show hint 2 of 3" is visible.
- A Buggy Message:** Shows a message: "AC - You are almost right, but remember that DF is twice x." Below it, a red box states: "Side AB corresponds to side DE of triangle DEF, not DF. Try again, please." A link "Show hint 2 of 3" is visible.

Figure 1. ASSISTments interface.

This is possible because we created an authoring tool that makes it easy to build, test, and deploy items, as well as for teachers to get reports. We have a gentle slope for authors in that they can use our

QuickBuilder to just type in a set of questions and associated answers. In that sense, they have created a simple quiz, where the one hint given would just tell them the answer. For those that want to add further hints to the questions, that step is easy and is part of the QuickBuilder. If they want to create scaffolding questions or feedback messages for common wrong answers, they have to invoke the ASSISTment Builder, requiring a steeper learning curve. While there is a steeper learning curve, we have shown that going through the work of creating scaffolding questions can be very helpful for the lower knowledge students (Razzaq & Heffernan, 2009), but that does not mean that everyone creating content in ASSISTments needs to create both a scaffolding version and a hint version.

This gives teachers the opportunity to create problems specific to their school, for differentiated instruction, or to work with their textbook. All content created by any user can be viewed by, but not edited by, any other user that has the problem number. This makes sharing easy and prevents teachers from having to worry that their content could get “graffiti” on it.

We are exploring a new way of adding content with teachers in Maine. The teacher types in something like the following, “Do #7 from Page 327,” so the students have to open their textbook to page 327 to see the seventh question on that page (in doing it this way, the teachers are not violating the copyright of the publisher by duplicating the problem). Teachers can elect for students to receive correctness only feedback or additional tutoring on the homework. The content created around these texts is driven by the teachers and can be shared by anyone using that book. Inspired by Ostrow and Heffernan (2014) that showed video hint messages were more effective than a text version that used the same words, we funded seven teachers to make video hint messages, posted on YouTube or SchoolTube. We are just starting a study to examine the effectiveness of this.

The variabilization feature of the ASSISTments builder allows an author to design one problem and then have many problems created that assess the same skill. This was key to our getting our Skill Builders running. Skill Builders are problem sets that allow a student to keep doing problems until they reach the proficiency threshold, which by default is three correct in a row but can be set by the author. Any teacher that wants to change that simply makes their own copy and changes it.

Figure 2 shows the interface for a variabilized assistance. Authors can variabilize the hint messages, scaffolding questions, and feedback messages. Authors have to write tiny programs of interconnected variables, which do things like randomly changing the numbers used in the problems. Skill builders are much harder to create, and only a few teachers do this themselves, but WPI has created several hundred for topics from 4th to 10th grade mathematics. Well over half of the teachers use our skill builders.

The screenshot shows a software interface for creating math problems. At the top is a toolbar with various icons for font sizes, bold, italic, underline, etc. Below the toolbar is a question text area containing a math problem: "What is the length of the hypotenuse of the right triangle shown below, given the length of leg a to be %v{side1} feet and length of leg b to be %v{side2} feet." Below the text is a diagram of a right triangle with vertices at the top-left, top-right, and bottom. The left leg is labeled 'a', the right leg is labeled 'b', and the hypotenuse is labeled 'c'. A small square at the top vertex indicates it is a right angle. At the bottom of the interface, there is a red instruction: "Round your answer to the nearest tenth place."

Figure 2. This is a variabilized problem on the Pythagorean Theorem.

The authoring tool for ASSISTments has a gentle usability slope. Many teachers start using ASSISTments by first using content WPI created, but most of them soon use the extensibility of the tool to write their own questions. Most of these questions will be what we call “naked,” or the lacking of scaffolding hints, as that takes more time to create. We do have some authors that have used the tool to create large libraries of content. For instance, one teacher successfully made hundreds of Advanced Placement (AP) statistics questions with extensive hints.

CTAT

Examples are used extensively in CTAT, a widely used suite of authoring tools (Aleven, McLaren, Sewall & Koedinger, 2009; Aleven, Sewall, McLaren & Koedinger, 2006; Koedinger, Aleven, Heffernan, McLaren & Hockenberry, 2004). CTAT supports the development of tutors that provide individualized, step-by-step guidance during complex problem solving. These tutors provide ample assistance within a problem, such as feedback on the steps, next-step hints, and error feedback messages. They also support individualized problem selection to help each individual student achieve mastery of all targeted knowledge components. Therefore, these tutors support most of the tutoring behaviors identified by VanLehn (2006) as characteristic of ITSs. Over the years, many tutors have been built with CTAT in a very wide range of domains (Aleven et al., 2009; under review). Many of these tutors have been shown to be effective in helping students learn in actual classrooms.

CTAT supports the development of two kinds of tutors: example-tracing tutors, which use generalized examples of problem-solving behavior as their central representation of domain knowledge, and model-tracing tutors (or Cognitive Tutors), which use a rule-based cognitive model for this purpose (Aleven, 2010; Aleven, McLaren, Sewall & Koedinger, 2006). Example-tracing tutors are an innovation that originated with CTAT; this tutoring technology was developed as part of developing CTAT; cognitive tutors on the other hand have a long history that pre-dates CTAT (e.g., Aleven & Koedinger, 2007; Anderson, Corbett, Koedinger & Pelletier, 1995; Koedinger, Anderson, Hadley & Martk, 1997). These two types of tutors support the same set of tutoring behaviors. The main difference is that rule-based cognitive tutors are more practical when a problem can be solved in many different ways (Waalkens, Aleven & Taatgen, 2013). CTAT supports three different approaches to authoring (Figure 3). Example-tracing tutors are built with a variety of end-user programming techniques, including building an interface through drag-and-drop and then programming by demonstration within that interface, where the author’s actions are recorded as paths in a behavior graph (Figure 4; the behavior graph is on the right). Rule-based tutors on the other hand can be built in CTAT either through rule-based cognitive modeling, a form of AI programming (Aleven, 2010) or through programming by automated rule induction by a module called SimStudent, which is described in the next section.

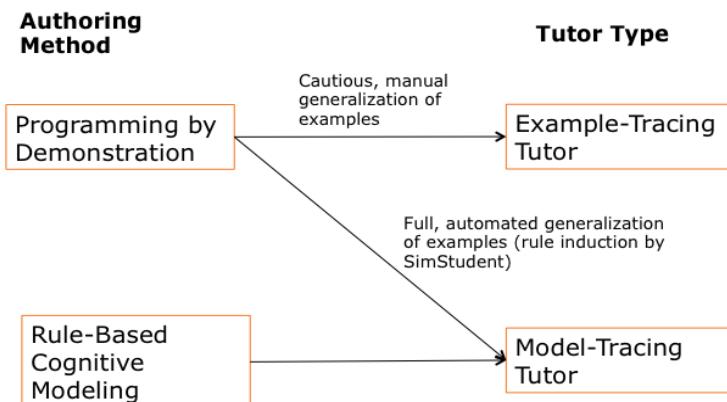


Figure 3. Tutor types and ways of authoring in CTAT

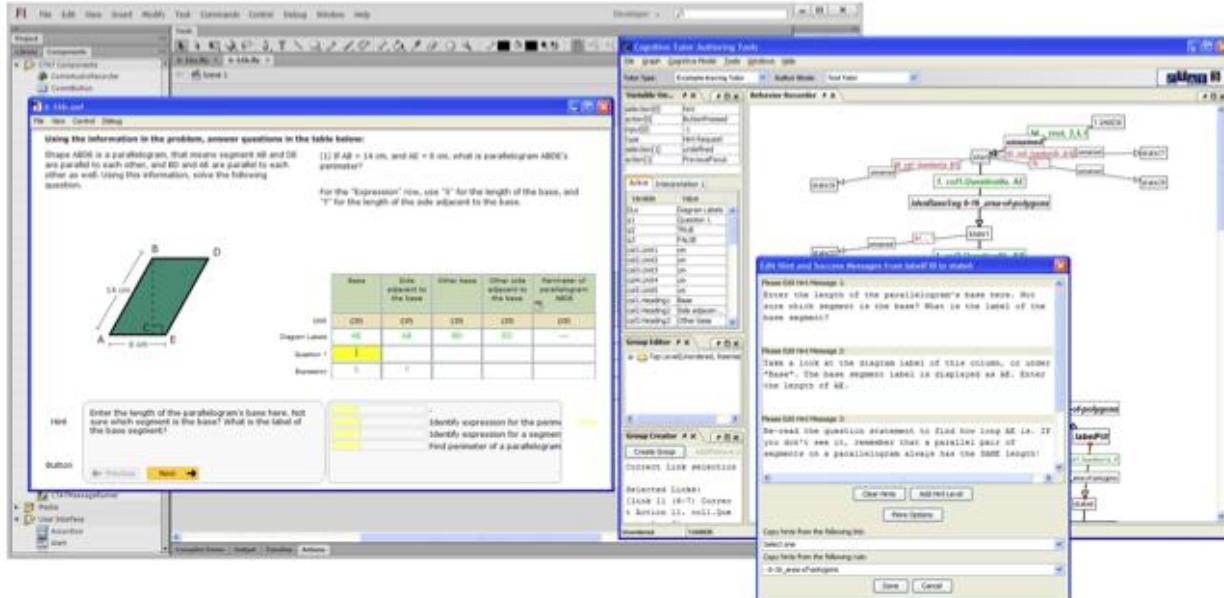


Figure 4. Author using CTAT (right) and Flash (left) to create an example-tracing tutor.

Examples figure prominently in each of these three authoring approaches. These examples take the form of behavior graphs, which capture correct and incorrect problem-solving behavior for the problems that the tutor will help students solve. A behavior graph may have multiple paths, each capturing a different way of solving the problem. Put differently, a behavior graph represents the solution space of a problem. Behavior graphs go at least as far back as Newell and Simon's (1972) classic book *Human Problem Solving*, a foundational work in cognitive science. An author can easily create behavior graphs using CTAT, by demonstrating how to solve problems in the tutor interface. A tool called the Behavior Recorder records the steps in a graph. CTAT also offers tools with which an author can *generalize* a behavior graph, expanding the range of problem-solving behavior that it represents.

Examples serve many different purposes in CTAT. In all three of CTAT's approaches to tutor authoring, examples (i.e., behavior graphs) function as a tool for cognitive task analysis. They help an author map out the solution space of the problems for which tutoring is to be provided, think about different ways a problem might be solved, and develop hypotheses about the particular knowledge components needed and how these components might transfer across steps. In addition, behavior graphs serve various separate functions in each of the authoring approaches. First, in example-tracing tutors generalized examples are the tutor's domain knowledge. The author generalizes the examples in various ways to indicate the range of student behaviors that the tutor will deem correct, so the tutor can be appropriately flexible in recognizing correct student behavior (Aleven, McLaren, Sewall & Koedinger, 2009). Also, in the common authoring scenario that many problems of the same type are needed, an author can turn a behavior graph into a template and create a table with specific values for each problem. Second, in building rule-based cognitive tutors by hand, the examples help in testing and debugging. They help navigate a problem's solution space (e.g., authors can jump to any problem-solving state captured in the graph, which is useful when developing a model from scratch), they serve as semi-automated test cases, and they can be used for regression testing (i.e., making sure that later changes do not introduce bugs). Lastly, in SimStudent, author-demonstrated examples are used to automatically induce production rules that capture the tutor's problem-solving behavior (more detail on this process can be found in the SimStudent section below).

As mentioned, example-tracing tutors use generalized examples (behavior graphs) to flexibly interpret student problem-solving behavior. The tutor checks whether the student follows a path in the graph. Once the student commits to a path, by executing one or more steps on that path, the example-tracer will insist that the student finishes that path, that is, that all subsequent actions are all on at least one path through the graph. Students are not allowed to backtrack and try an alternative problem-solving strategy within the given problem in order to keep them moving forward. Within this basic approach, CTAT's example tracer is very flexible in how it matches a student's problem-solving steps against a behavior graph. First, the example tracer can handle ambiguity regarding which path the student is on and when the steps that the student has entered so far are consistent with multiple paths in a graph. In such situations, the example tracer will maintain multiple alternative interpretations of student behavior until subsequent student steps rule out one or more interpretations. The example tracer also can deal with variations in the order of steps. That is, the student does not need to strictly follow the order in which the steps appear in the graph. An author can specify which parts of a behavior graph require a strict order and which steps can be done in any order. Even better, an author can create a hierarchy of nested groups of unordered and ordered steps. Further, steps can be marked as optional or repeatable. The example tracer can also deal with variations of the steps themselves. An author has a number of ways to specify a range of possibilities for a particular step, including range matches, wildcard matches, regular expressions, as well as an extensible formula language for specifying calculations and how a step depends on other steps. Thus, in CTAT example-tracing tutors, a behavior graph can stand for a wide range of behavior well beyond exactly the steps in the graph in exactly the order they appear in the graph. Authors have many tools that enable them to specify how far to generalize. When an author wants to make behavior graphs for many different but isomorphic problems, CTAT provides a "Mass Production" approach in which an author creates a behavior graph with variables for the problem-specific values and then, in Excel, creates a table with problem-specific values for a range of problems. They can then generate specific instances of the template in a merge step. This template-based process greatly facilitates the creation of a series of isomorphic problems, as are typically needed in tutor development.

Our experience over the years, both as developers of example-tracing tutors and consultants assisting others in developing example-tracing tutors, indicates that this type of tutor is useful and effective in a range of domains. It also indicates that the example-tracing technology implemented in CTAT routinely withstands the rigors of actual classroom use. Examples of example-tracing tutors recently built with CTAT and used in actual classrooms are Mathtutor (Aleven, McLaren & Sewall, 2009), the Genetics Tutor (Corbett, Kauffman, McLaren, Wagner & Jones, 2010), the Fractions Tutor (Rau, Aleven & Rummel, 2015; Rau, Aleven, Rummel & Pardos, 2014), a version of the Fractions Tutor for collaborative learning (Olsen, Belenky, Aleven & Rummel, 2014; Olsen, Belenky, Aleven, Rummel, Sewall & Ringenberg, 2014), a fractions tutor that provides grounded feedback (Stampfer & Koedinger, 2013), the Stoichiometry Tutor (McLaren, DeLeeuw & Mayer, 2011a; 2011b), AdaptErrEx (Adams et al., 2014; McLaren et al., 2012), an English article tutor (Wylie, Sheng, Mitamura & Koedinger, 2011), Lynnette, a tutor for equation solving (Long & Aleven, 2013; Waalkens et al., 2013), and a tutor for guided invention activities (Roll, Holmes, Day & Bonn, 2012). We have also seen, in courses, workshops, and summer schools that we have taught, that learning to build example-tracing tutors with CTAT can be done in a relatively short amount of time. Generally, it does not take more than a couple of hours to get started, a day to understand basic functionality, and a couple more days to grasp the full range of functionality that this tutoring technology offers. This is a much lower learning curve than that for learning to build cognitive tutors with CTAT. Authoring and debugging a rule-based cognitive models is a more complex task that requires AI programming. Example-tracing tutors on the other hand do not require any programming. In our past publication (Aleven, McLaren, Sewall & Koedinger, 2009), we estimated, based on data from projects in which example-tracing tutors were built and used in real educational settings (i.e., not just prototypes) that example-tracing tutors make tutor development 4–8 times more cost-effective: they can be developed faster and do not require expertise in AI programming. Echoing a theme that runs throughout the chapter, we emphasize that building a good tutor requires more than being

facile with authoring tools; for example, it also requires careful cognitive task analysis to understand student thinking and students' difficulties in the given task domain.

In sum, the CTAT experience indicates that the use of examples, in the form of behavior graphs that capture the solution space of a problem, is key to offering easy-to-learn, non-programmer options to ITS authoring. Thinking in terms of examples and concrete scenarios is helpful for authors. So is avoiding actual coding, made possible by the use of examples. The experience indicates also that the same representation of problem-solving examples, namely, behavior graphs, can serve many different purposes. This versatility derives from the fact that behavior graphs are a general representation of problem-solving processes. As such, they may be useful in a range of ITS authoring tools, not just CTAT, since many ITSs deal with complex problem-solving activities.

SimStudent

SimStudent is a machine-learning agent that inductively learns problem-solving skills (Li, Matsuda, Cohen & Koedinger, 2015; Matsuda, Cohen & Koedinger, 2005). At an implementation level, SimStudent acts as a pedagogical agent that can be interactively tutored. SimStudent is a realization of programming by demonstration (Cypher, 1993; Lau & Weld, 1998) in the form of inductive logic programming (Muggleton & de Raedt, 1994). SimStudent learns domain principles (i.e., how to solve problems) by specializing and generalizing positive and negative examples on how to apply, and not to apply, particular skills to solve problems.

At a theory level, SimStudent is a computational model of learning that explains both domain-general and domain-specific theories of learning. As for the domain-general theory of learning, SimStudent models two learning strategies: learning from examples and learning by doing (Matsuda, Cohen, Sewall, Lacerda & Koedinger, 2008). *Learning from examples* is a model of passive learning in which SimStudent is given a set of worked-out examples and it silently generalizes solution steps from these examples. There is no interaction between the “tutor” and SimStudent during learning from examples, except that tutor provides examples to SimStudent. *Learning by doing*, on the other hand, is a model of interactive, tutored-problem solving (i.e., cognitive tutoring) in which SimStudent is given a sequence of problems and asked to solve them. In this context, there must be a “tutor” (i.e., author) who provides tutoring scaffolding (i.e., feedback and hints) to SimStudent. That is, the “tutor” provides immediate flagged feedback (i.e., correct or incorrect) for each of the steps that SimStudent performs. SimStudent may get stuck in the middle of a solution and ask the “tutor” for help on what to do next. The “tutor” responds to SimStudent’s inquiry by demonstrating the exact next step.

As for the domain-specific theory of learning, SimStudent can be used as a tool for student modeling to advance a cognitive theory of learning skills to solve problems for a particular domain task. Using the SimStudent technology, researchers can conduct simulation studies with tightly controlled variables. For example, to understand why students make commonly observed errors when they learn how to solve algebraic linear equations, we conducted a simulation study. An example of a common error is to subtract 4 from both sides of $2x - 4 = 5$. We hypothesized that students learn skills incorrectly due to incorrect induction. We also hypothesized that incorrect induction might more likely occur when students carry out induction based on weak background knowledge that, by definition, is perceptually grounded and therefore lacks connection to domain principles. An example of such weak background knowledge is to perceive “3” in $5x + 3 = 7$ as *a last number* on the left-hand side of the equation, instead of perceiving ‘+3’ as *a last term*. To test these hypotheses, we controlled SimStudent’s background knowledge by replacing some of the background knowledge (e.g., the knowledge to recognize the last term) with weak perceptually grounded knowledge (e.g., the knowledge to recognize the last number). We trained two versions of SimStudent (one with normal background knowledge and the other one with weak

background knowledge) and compared their learning with students' learning. The result showed that only SimStudent with weak background knowledge made the same errors that students commonly make (Matsuda, Lee, Cohen & Koedinger, 2009).

So far, we have demonstrated that SimStudent can be used to advance educational studies for three major problems: (1) intelligent authoring, (2) student modeling, and (3) teachable agent. For intelligent authoring, SimStudent functions as an intelligent plug-in component for CTAT (Aleven, McLaren, Sewall & Koedinger, 2006; Aleven, McLaren, Sewall & Koedinger, 2009) that allows authors to create a cognitive model (i.e., a domain expert model) by tutoring SimStudent on how to solve problems. The intelligent authoring project was started as an extension of prior attempts (Jarvis, Nuzzo-Jones & Heffernan, 2004; Koedinger, Aleven & Heffernan, 2003; Koedinger, Aleven, Heffernan, McLaren & Hockenberry, 2004).

In the context of intelligent authoring, the author first creates a tutoring interface using CTAT, and then "tutors" SimStudent using the tutoring interface (Figure 5). There are two authoring strategies, *authoring by tutoring* and *authoring by demonstration*, and each corresponds to two learning strategies mentioned above, i.e., learning by doing and learning from worked-out examples, respectively. We have showed that when the quality of a cognitive model is measured as the accuracy of solution steps suggested by the cognitive model, authoring by tutoring generates a better cognitive model than authoring by demonstration (Matsuda, Cohen & Koedinger, 2015). It is only authoring by tutoring that provides negative examples, which by definition tell SimStudent when not to apply overly general productions, and negative examples have the significant role in inductively generating a better quality cognitive model.

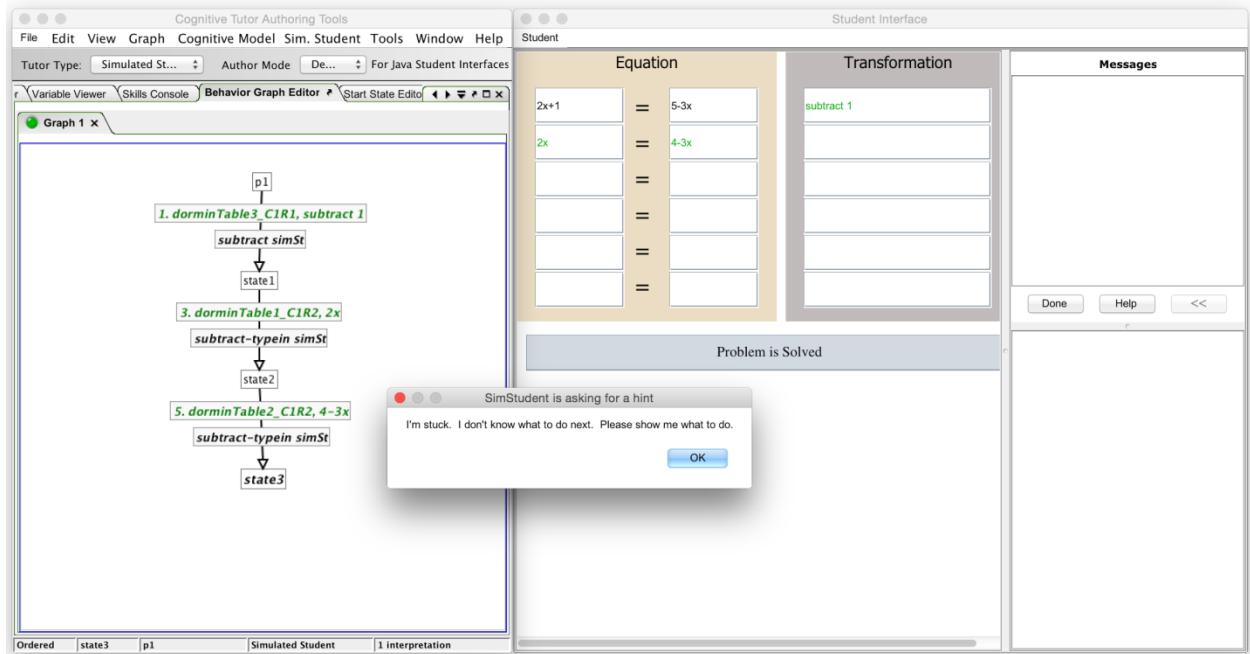


Figure 5. Authoring using SimStudent with the assistance of CTAT

SimStudent also functions as a teachable agent in an online learning environment in which students learn skills to solve problems by interactively teaching SimStudent. The online learning environment is called the Artificial Peer Learning environment Using SimStudent (APLUS). APLUS and a cognitive tutor share underlying technologies. In fact, APLUS consists of (1) the tutoring interface on which a student tutors

SimStudent; (2) a cognitive tutor in the form of the *meta-tutor* that provides scaffolding for the student on how to teach SimStudent and how to solve problems; and (3) a teachable agent (SimStudent), with its avatar representation. The combination of CTAT and SimStudent allows users to build APLUS for their own domains. In this context, SimStudent plays a dual role: (1) a tool to create a cognitive model for the embedded meta-tutor and (2) a teachable agent.

Examples, in the context of the interaction with SimStudent, are major input for SimStudent to induce a cognitive model. SimStudent learns procedural skills to solve target problems either from learning by doing or learning from worked-examples. SimStudent generalizes provided examples (both positive and negative) and generates a set of productions that each represents a procedural skill. The set of productions become a cognitive model that can be used for cognitive tutoring in the form of a cognitive tutor or a meta-tutor in APLUS.

An empirical study (Matsuda et al., 2015) showed that to make an expert model for an algebra cognitive tutor, it took a subject matter expert 86 minutes for *authoring by tutoring* SimStudent on 20 problems whereas *authoring by demonstration* with 20 problems took 238 minutes. A more recent study showed that authoring an algebra tutor in SimStudent is 2.5 times faster than example-tracing while maintaining equivalent final model quality (MacLellan, Koedinger & Matsuda, 2014). We are currently conducting a study to validate the quality of production rules. In the study, we actually use a SimStudent-generated cognitive model for an algebra cognitive tutor to model trace real student's solution steps. A preliminary result shows that after tutoring SimStudent on 37 problems, the model tracer correctly model traces 96% of steps that students correctly performed. At the same time, the "accuracy" of detecting a correct step (i.e., the ratio of the correct positive judgement, judging a step as correct, to all positive judgement) was 98%.

ASPIRE

The Intelligent Computer Tutoring Group (ICTG; <http://www.ictg.canterbury.ac.nz/>) has developed many successful constraint-based tutors in diverse instructional domains (Mitrovic, Martin & Suraweera, 2007; Mitrovic, 2012). Some early comparisons of constraint-based modeling (Ohlsson, 1994) to the model-tracing approach have shown that constraint-based tutors are less time-consuming to develop (Ohlsson & Mitrovic, 2007; Mitrovic, Koedinger & Martin, 2003), but yet require substantial expertise and effort. The estimate of time per constraint for Structured Query Language (SQL)-Tutor, the first and biggest constraint-based modeling (CBM) tutor developed (Mitrovic, 1998), was 1 hour per constraint, with the same person acting as the knowledge engineer, domain expert, and software developer. In order to support the development process, ICTG developed an authoring shell, the Web-Enabled Tutor Authoring System (WETAS; Martin & Mitrovic, 2002). Studies with novice ITS authors using WETAS had shown that the authoring time per constraint on average was 2 hours (Suraweera et al., 2009), but the authors still found writing constraints challenging.

ASPIRE (<http://aspire.cosc.canterbury.ac.nz/>) is a general authoring and deployment system for constraints-based tutors. It assists in the process of composing domain models for constraint-based tutors and automatically serves tutoring systems on the web. ASPIRE guides the author through building the domain model, automating some of the tasks involved, and seamlessly deploys the resulting domain model to produce a fully functional web-based ITS.

The authoring process in ASPIRE consists of eight phases. Initially, the author specifies general features of the chosen instructional domain, such as whether or not the task is procedural. For procedural tasks, the author describes the problem-solving steps. This is not a trivial activity, as the author needs to decide on

the approach to teaching the task. The author also needs to decide on how to structure the student interface and whether the steps will be presented on the same page or on multiple pages.

The author then develops the domain ontology, containing the concepts relevant to the instructional task. The purpose of the domain ontology is to focus the author on important domain concepts; ASPIRE does not require a complete ontology, but only those domain concepts students need to interact with in order to solve problems in the chosen area. The ontology specifies the hierarchical structure of the domain in terms of sub- and super-concepts. Each concept might have a number of properties and may be related to other domain concepts. The author can define restrictions on properties and relationships, such as the minimum and maximum, number of values, types of values, etc. The ontology editor does not offer a way of specifying restrictions on different properties attached to a given concept, such as the number of years of work experience should be less than the person's age. It also does not contain functionality to specify restrictions on properties from different concepts, such as the salary of the manager has to be higher than the salaries of employees for whom they are responsible. However, these restrictions are not an obstacle for generating the constraint set, as ASPIRE generates constraints not only from the ontology, but also from sample problems and their solutions. Figure 6 shows the domain ontology for the thermodynamics tutor, which is defined as a procedural task. In this tutor, the student needs to develop a diagram first and later compute unknowns using a set of formulas.

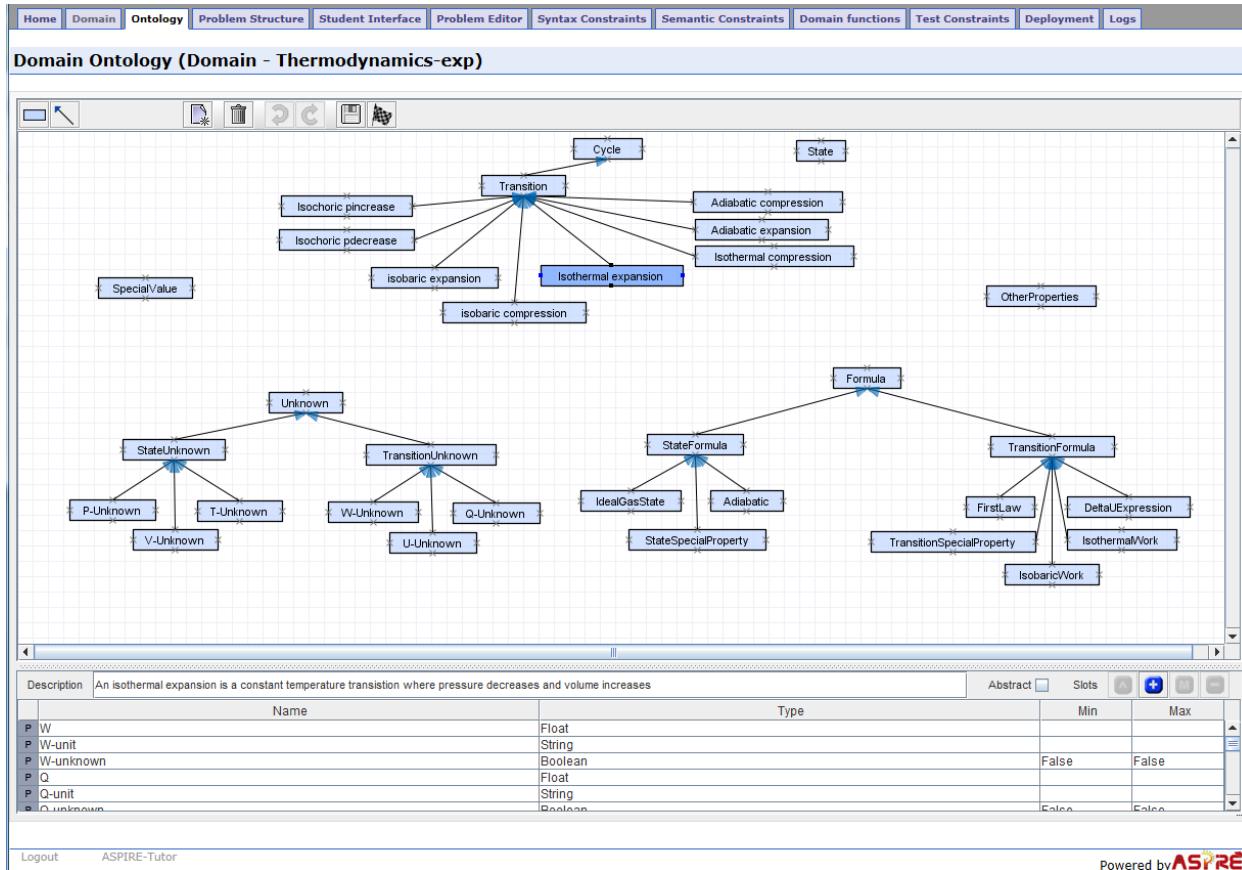


Figure 6: The ontology of Thermo-Tutor

In the third phase, the author defines the problem structure and the general structure of solutions, expressed in terms of concepts from the ontology. The author specifies the types of components to show on the student interface and the number of components (e.g., a component may be optional or can have

multiple instances). On the basis of the information provided by the author in the previous phases, ASPIRE then generates a default, text-based student interface, which can be replaced with a Java applet. ASPIRE also provides a remote procedure call interface, allowing for sophisticated student interfaces to be built, such as an Augmented Reality interface (Westerfield, Mitrovic & Billinghamurst, 2013). Figure 7 shows the Java applet allowing students to solve problems in Thermo-Tutor (Mitrovic et al., 2011).

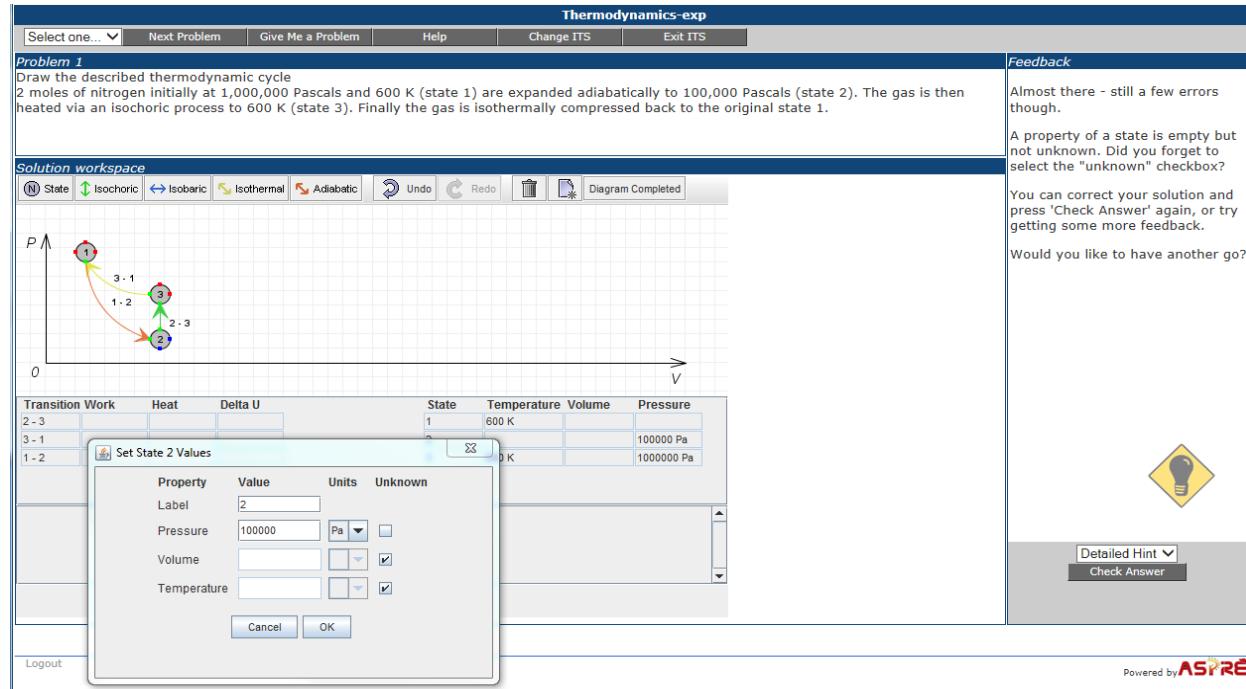


Figure 7: A screenshot from Thermo-Tutor showing the applet

In the fifth phase, the author adds sample problems and their correct solutions using the problem solution interface. ASPIRE does not require the author to specify incorrect solutions. The interface enforces that the solutions to adhere to the structure defined in the previous step. The author is encouraged to provide multiple solutions for each problem, demonstrating different ways of solving it. In domains where there are multiple solutions per problem, the author should enter all practicable alternative solutions. The solution editor reduces the amount of effort required to do this by allowing the author to transform a copy of the first solution into the desired alternative. This feature significantly reduces the author's workload because alternative solutions often have a high degree of similarity.

ASPIRE then generates syntax constraints by analyzing the ontology and the solution structure. The syntax constraint generation algorithm extracts all useful syntactic information from the ontology and translates it into constraints. Syntax constraints are generated by analyzing relationships between concepts and concept properties specified in the ontology (Suraweera, Mitrovic & Martin, 2010). An additional set of constraints is also generated for procedural tasks, which ensure the student performs the problem-solving steps in the correct order (also called *path constraints*).

Semantic constraints check that the student's solution has the desired meaning (i.e., it answers the question). Constraint-based tutors determine semantic correctness by comparing the student solution to a single correct solution to the problem; however, they are still capable of identifying alternative correct solutions because the constraints are encoded to check for equivalent ways of representing the same semantics (Ohlsson & Mitrovic, 2007; Mitrovic, 2012). ASPIRE generates semantic constraints by analyzing alternative correct solutions for the same problem supplied by the author. ASPIRE analyses the

similarities and differences between two solutions to the same problem. The process of generating constraints is iterated until all pairs of solutions are analyzed. Each new pair of solutions can lead to either generalizing or specializing previously generated constraints. If a newly analyzed pair of solutions violates a previously generated constraint, its satisfaction condition is generalized in order to satisfy the solutions, or the constraint's relevance condition is specialized for the constraint to be irrelevant for the solutions. A detailed discussion of the constraint-generation algorithms is available in (Suraweera, Mitrovic & Martin, 2010).

xPST

When an author uses the xPST system to create a model-tracing style tutor (e.g., Koedinger, Anderson, Hadley & Mark, 1997) for a learner, the author bases the instruction on a particular example. The example needs to already be in existence—xPST does not provide a way to create that example. Rather, that example comes from previously created content or is based on third-party software. This aspect of the system is contained in its name—problem-specific tutor. Very little generalization is done from the example. Broadly speaking, the instruction that the author creates is appropriate only for that one example. While this limits the ability for the instruction to be applied in multiple instances, it allows for a more streamlined and simplistic authoring process, opening up the possibility of authoring tutors to a wider variety of people, e.g., those who do not possess programming skills.

To quickly explain the first word in the xPST name, extensible, that ability comes from two different aspects. First, xPST can be extended in terms of the types of learner answers it can check. xPST's architecture compartmentalizes these “checktypes,” and it is easy for a programmer to add additional ones and make them available to xPST authors. Second, and more importantly, xPST can be extended in terms of the interfaces on which it can provide tutoring. Like other ITSs (such as seen in CTAT, or see Blessing, Gilbert, Ourada, and Ritter, 2009; Ritter & Koedinger, 1996), xPST's architecture makes a clear separation between the learner's interface and the tutoring engine. The architecture contains a TutorLink module that mediates the communication between these two parts of the system. The learner's interface can in theory be any existing piece of software, as long as a TutorLink module can translate the actions of the learner in the interface into what xPST understands, and then the module needs to communicate the tutoring feedback back to the learner's interface (e.g., a help message or an indication if an answer is right or wrong). More information concerning this type of communication can be found elsewhere (Gilbert, Blessing & Blankenship, 2009).

Allowing the learner interface to be existing software, given the proper TutorLink module, opens up many possibilities in terms of what to provide tutoring on and how that tutoring manifests itself. We have written TutorLink modules for Microsoft .NET programs, the Torque 3-D game engine, and the Firefox web browser. Regardless of the interface, the authoring interaction is similar: a specific scenario is created within the context of the interface and instruction on completing that scenario is authored in xPST. To explain how examples are used to create tutoring in xPST, we illustrate the process using the Firefox web browser as the interface. In this case, the TutorLink module operates as a Firefox plug-in. This allows any webpage to contain potentially tutorable content, where the student is provided with model-tracing style feedback. In one project, we had authors, which included non-programmer undergraduates, use a drag-and-drop form creation tool to easily create custom homework problems for a statistics tutor (Maass & Blessing, 2011). Countless webpages already exist that could be used for instruction. In another project, we used a webpage from the National Institutes of Health (NIH) to create activities involving DNA sequencing.

To provide a specific example, imagine an author wanted to create instruction on how to search using a popular article database, the American Psychological Association's (APA) PsychINFO, to find research

papers, so that students become better at information literacy. The webpage already exists, with all the widgets (the entry boxes, radio buttons, and pull-down menus) in place. The Firefox plug-in allows the author to write a problem scenario (e.g., to find a particular paper using those widgets) that will appear in a sidebar next to the already established page, and then the author writes instruction code that will ensure that the learner uses the page appropriately, providing help when needed, so that the learner finds the correct article. The author does their work on the xPST website (<http://xpst.vrac.iastate.edu>). This website provides a form to create a new problem, where the instruction to the existing webpage (in this case, <http://search.proquest.com/psychinfo/advanced/>), the sidebar's problem scenario, and the tutor "code" that contains the right answers and help messages can all be entered. While the code does have some of the trappings of traditional programming, those are kept to a minimum.

Figure 8 shows some of the code that would be used to create this PsychINFO tutor. This code in conjunction with the author-supplied scenario is in essence the example. The existing webpage provides the means by which the learner will work through the example (via the entry boxes and drop-down menus), and what is seen in Figure 8 is the information needed by xPST to provide tutoring. The code has three main sections: Mappings, Sequence, and Feedback. The Mappings map the interface widgets onto the names that the xPST tutor will use. The Firefox plug-in provides the names of the widgets for the author as the author begins to create the scenario. The Sequence is the allowed orderings for how the learner may progress through the problem. The syntax allows for required and optional parts, along with different kinds of branching. The Feedback section is where the author indicates the right answer for a widget, and the help and just-in-time messages that might be displayed for incorrect responses. Once authors have entered in enough code to see results, they can click the "Save and Run" button and immediately see the results of the xPST tutor. Figure 9 shows the tutor running the PsychINFO site with the code shown in Figure 8. This code is specific to this problem scenario, but could easily be copied and modified in order to create a different problem. In such a way an author could quickly create a short 5–6 problem homework set to provide practice to students concerning information literacy.

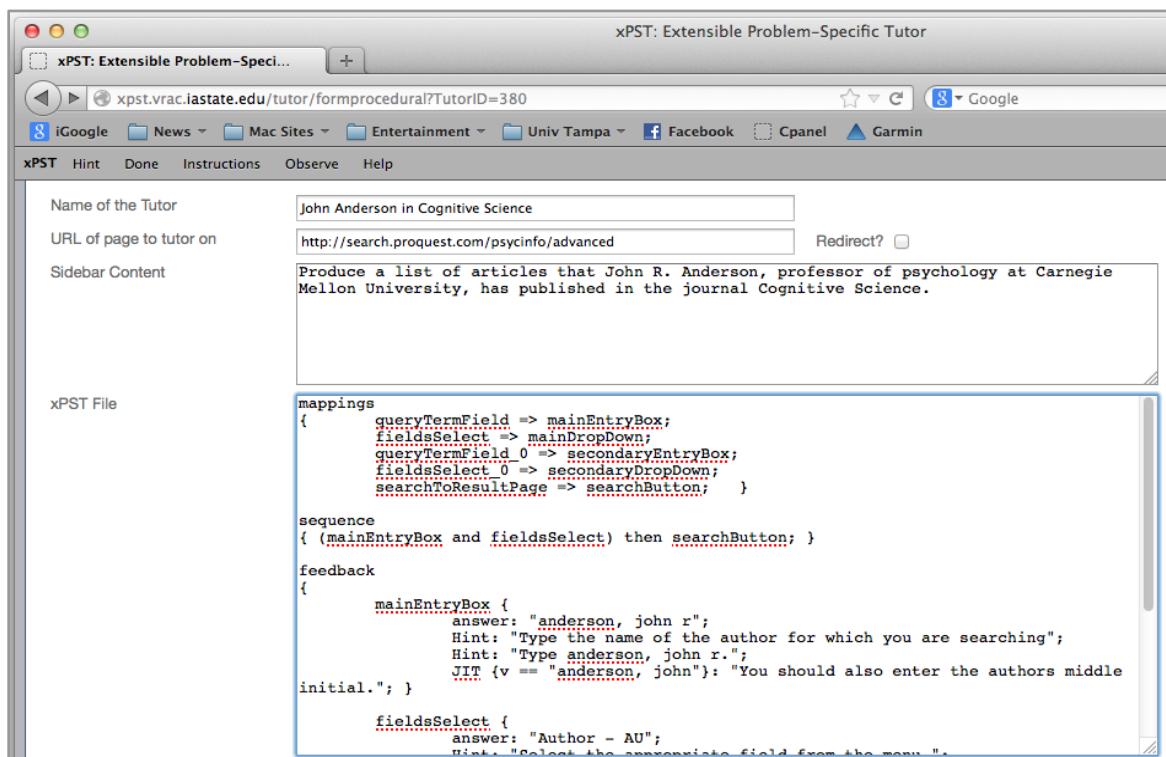


Figure 8. Authoring interface for xPST.

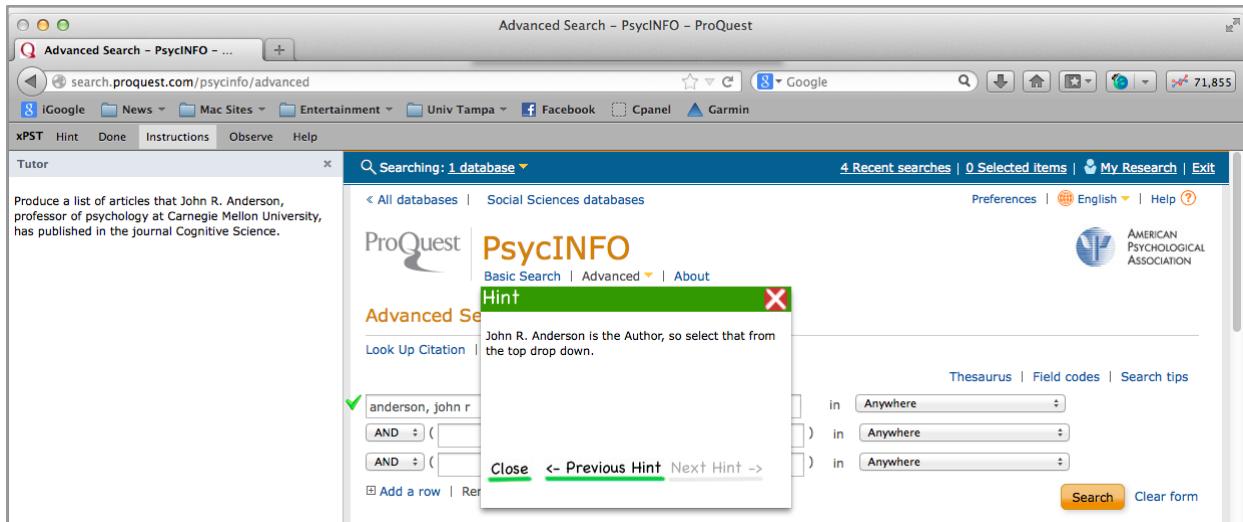


Figure 9. The example-based tutor running on the PsychInfo site.

We have examined the way non-programmers have learned to use xPST (e.g., Blessing, Devasani & Gilbert, 2011). Despite the text-entry method for instruction, non-programmers have successfully used xPST to create new tutors. In Blessing, Devasani, and Gilbert (2011), five such authors spent roughly 30 hours on average learning the system and developing 15 statistics problems apiece. Keeping in mind that all the problems had a similar feel to them, the endpoint was the ability to create one of the problems, which contained about 10 minutes of instruction, in under 45 minutes.

Conclusions

We start our conclusions by comparing the above systems on five dimensions: (1) their heritage, (2) practical concerns such as teacher reporting, (3) the authoring process, (4) how they generalize examples, and (5) their approach to cognitive task analysis. We finish by making recommendation to the Generalized Intelligent Framework for Tutoring (GIFT) architecture based on our observations.

Heritage

Four of these five systems (ASSISTments, CTAT, SimStudent, and xPST) share a common heritage, the ACT Tutors that John Anderson and his colleagues developed over the course of many years (Anderson, Corbett, Koedinger & Pelletier, 1995). The researchers created these tutors to fully test the ACT Theory of cognition, and they covered a few different domains, including several programming languages and many levels of mathematics. The most direct descendant of the ACT Tutors existing today are the commercial tutors produced by Carnegie Learning, Inc., which cover middle and high school math.

Despite this common heritage of the present systems, they were developed independently. Each of us felt that the authoring tools created to support the ACT Tutors (the Tutor Development Kit for the original set of tutors (Anderson & Pelletier, 1991) and the Cognitive Tutor Software Development Kit for Carnegie Learning's tutors (Blessing, Gilbert, Ourada & Ritter, 2009)), while powerful, were not approachable by non-programmers or non-cognitive scientists. We realized that in order for ITSs to be more prevalent, authoring needed to be easier. In our own labs, we developed separate systems that mimicked the behavior of the original ACT Tutors, because that had proved so successful, but without the programming

overhead that prior tools required. As seen in our descriptions above and our discussion here, these systems contain some similarities, but differ in important ways as well.

ASPIRE, the one system that does not have a connection to the ACT Tutors, originated with Ohlsson's work on a theory of learning from performance errors (Ohlsson, 1996). This led to the development of CBM (Mitrovic & Ohlsson, 1999), in which the tutor's knowledge is represented as a set of constraints, as opposed to the production-based representation of the ACT Tutors. In this way, the tutor's knowledge represents boundary points within which the solution lies. Having multiple systems that descend from multiple sources provides credence to the idea that the general technique of programming by demonstration and the use of examples is a useful and powerful one for the creation of ITSs.

Practical Concerns

There are scientific concerns as to what knowledge representations are most valuable to use to reflect how humans think (e.g., Ohlsson's constraints-based theory vs. Anderson's production rules). However, there are also practical concerns. For example, which tools prove easier to use might drive adoption, not necessarily those that produce the most learning. As another somewhat practical concern, some of the authoring methods discussed above may allow authors to more easily add complexity to their content over time. For instance, after assigning a homework question, a teacher may see that an unanticipated common wrong answer occurs, and the system needs to allow the teacher to write a feedback message that addresses that common wrong answer quickly.

While this chapter has focused on author tools for the content, an equally important element has to do with reporting. Some of these tools, such as ASSISTments and CTAT offer very robust ways to report student data. There is a possible tradeoff on the complexity and adaptability of the content, and the ways we report to instructors. We need easy ways that report information to the instructors and content creators. The reports to these classes of people should be focused differently than the types of reports to researchers. For instance, if a researcher has used ASSISTments' tools to create a randomized controlled experiment (see sites.google.com/site/neilheffernanscv/webinar for more information concerning this feature) embedded in a homework, perhaps comparing text hints versus video hints, the reports that the teachers receive should be different than the reports that the researchers receive.

The Authoring Process

The method by which authors create tutors in these systems varies along at least two different, though somewhat related, dimensions: (1) how the instruction is inputted and (2) how much of the process is automated. With regard to how the instruction is inputted, this varies from a method that is more traditional coding as in xPST, to a method that is more graphical in nature, such as CTAT's behavior graph. ASSISTment's QuickBuilder and ASSISTment Builder techniques seem to be a bit of a midpoint between those two methods of input. Devasani, Gilbert, and Blessing (2012) examined the trade-offs between these approaches with novice authors building tutors in both CTAT and xPST within two different domains, statistics and geometry. Relating their findings to Green and Petre's (1996) cognitive dimensions, they argued that the GUI approach has certain advantages, such as eliminating certain types of errors and the fact that visual programming allows for a more direct mapping. A more text-based approach has the advantages of flexibility in terms of how the authoring is completed and the ability to capture larger tutors that contain more intermediate states and solution paths more economically (what Green and Petre termed "diffuseness" and "terseness"). That flexibility may also translate into easier maintenance of those larger tutors.

The systems also differ in how much of the process is automated. This is also related to the amount of generalizability that the systems are able to perform, discussed below. In both ASSISTments and xPST, very little, if anything, is automated. ASPIRE and SimStudent have some degree of automation, in terms of how they induce constraints or productions. This automation eliminates or reduces greatly some of the steps that the author would otherwise have to do in order to input the instruction. CTAT is the middle system here, as it does have some mechanisms available to the author to more automatically created instruction (e.g., using Excel to more quickly create problem sets that all share similar instruction). As in any interface and systems design, these two dimensions play off each other in terms of what advantages they offer the author, between ease-of-use and generalizability.

Generalization of Examples

The discussed authoring technologies are diverse: they help authors create different kinds of domain models that can be used for adaptive tutoring. Some help authors create a collection of questions and answers with knowledge of feedback (ASSISTments, the example-tracing version of CTAT, and xPST), whereas others provide scaffolding to create the domain model either in the form of constraints (ASPIRE) or production rules (the model-tracing version of CTAT and SimStudent).

Some of the discussed approaches rely on the author's ability for programming while providing elaborated scaffolding to facilitate the programming process and ease the author's labor (ASSISTments, CTAT, xPST). In xPST, there is no generalization of examples at all. In CTAT, the author specifies the behavior graphs that includes both correct and incorrect steps, and also provides feedback on steps and hints. Furthermore, the author generalizes examples by adding variables and formulas that express how steps depend on each other or how a given vary, by relaxing ordering constraints, and by marking steps as optional or repeatable. In ASSISTments, some variabilization is possible. In those two cases, the authoring system does not generalize examples on its own; this task is left to the author.

On the other hand, ASPIRE and SimStudent deploy AI technologies to generate the domain model given appropriate background knowledge. ASPIRE, for example, generates constraints given the domain ontology developed by the author and example solutions. SimStudent uses the given primitive domain skills to generate a cognitive model from a set of positive and negative examples provided by the author. The difference between ASPIRE and SimStudent is not only in the formalism in which domain knowledge is represented, but also in the kind of examples they use. ASPIRE requires the author to specify only the alternative correct solutions for problems, without any feedback or further elaborations on them. SimStudent requires immediate feedback on steps (when inducing production rules in the learning by doing mode) or a set of positive and negative examples.

All five authoring systems discussed in this chapter share a common input for tutor authoring—example solutions. Different techniques are used for different purposes to generalize or specialize the given examples. It must be noted that all these five authoring systems share a fundamentally comparable instructional strategy for procedural tasks, step decomposition (i.e., force students to enter a solution one step at a time). ASPIRE differs from this requirement, as it can also support non-procedural tasks, in which the student can enter the whole solution at once. With the exception of ASPIRE, which provides on-demand feedback, the other authoring systems provide immediate (or semi-immediate) feedback on the correctness of the step performed, and just-in-time hint on what to do next.

Cognitive Task Analysis

Performing a cognitive task analysis (CTA) has been shown to be an effective means of producing quality instruction in a domain (Clark & Estes, 1996). CTA involves elucidating the cognitive structures that

underlie performance in a task. Another aspect of CTA is to describe the development of that knowledge from novice to expert performance. The more ITS authors (or any other designers of instruction) understand about how students learn in the given task domain, what the major hurdles, errors, and misconceptions are, and what prior knowledge students are likely to bring to bear, the better off they are. This holds for designing many, if not all, other forms of instruction, regardless of whether any technology is involved.

The space of cognitive task methods and methodologies is vast (Clark, Feldon, van Merriënboer, Yates & Early, 2007). Some of these techniques have been applied successfully in tutor development (Lovett, 1998; Means & Gott, 1988; Rau, Aleven, Rummel & Rohrbach, 2013). Two techniques that have proven to be particularly useful in ITS development, though not the only ones, are think-aloud protocols and a technique developed by Koedinger called difficulty factors assessment (DFA; Koedinger & Nathan, 2004). DFA is a way of creating a test (with multiple forms and a Latin-Square logic) designed to evaluate the impact on student performance of various hypothesized difficulty factors. Creating these tests is somewhat of an art form, but we may see more data-driven and perhaps crowd-based approaches in the future. Baker, Corbett, and Koedinger (2007) discussed how these two forms of cognitive task analysis can help, in combination with iterative tutor development and testing to detect and understand design flaws in a tutor and create a more effective tutor. Interestingly, in the area of ITS development, manual approaches to CTA are more and more being supplemented by automated or semi-automated approaches, especially in the service of building knowledge component models that accurately predict student learning (Aleven & Koedinger, 2013). CTA is important to ITS development, as it is for other forms of instructional design. The more instruction is designed with a good understanding of where the real learning difficulties lie, the more effective the instruction is going to be. ITSs are no exception. This point was illustrated in the work by Baker et al. (2007) on a tutor for middle school data analysis—CTA helped make a tutor more effective. Outside the realm of ITSs, this point was illustrated in the redesign of an online course for statistics, using CTA, where the redesigned course was dramatically more effective (Lovett, Myers & Thille, 2008).

Given the importance of CTA in instructional design, we should ask to what degree ITS authoring tools support any form of CTA and in what ways they are designed to take advantage of the results of CTA to help construct an effective tutor and perhaps make tutor development more efficient. For example, one function of the behavior graphs used in CTAT is as a CTA tool. The other authoring tools described here make use of CTA in various ways as the author creates a tutor. Although mostly implicit in their design, the authoring systems depend on authors having performed an adequate task decomposition in their initial interface construction, sometimes referred to as subgoal reification (Corbett & Anderson, 1995). Without the author having enabled the learner to make explicit their thought processes as they use the tutor, then attempts at assessing their current state of knowledge or addressing any deficiency will be greatly diminished. Therefore, before beginning the writing of any help or just-in-time messages, it is crucial to have the student's interface support the appropriate tasks needed to be performed by the student.

As mentioned, CTA is supported in CTAT, as the easily recorded behavior graphs. A behavior graph is a map of the solution space for a given problem for which the tutoring-system-being-built will provide tutoring. In other words, it simply represents ways in which the given problem can be solved. CTAT provides a tool, the Behavior Recorder, for creating them easily. Behavior graphs help in analyzing the knowledge needs, support thinking about transfer, and thereby guide the development of a cognitive model.

As a representation of the solution space of a problem, behavior graphs are not tied to any particular type of tutor and are likely be useful across a range of tutor authoring tools, especially those addressing tutoring for problems with a more complex solution space. For example, they may be helpful in tools for building constraint-based tutors. They may be less useful in the ASSISTments tool, given ASSISTments

strongly constrains the variability of the problems' solution space, with each problem essentially having one single-step path and multi-step path, the latter representing the scaffolded, version.

As the CTAT author creates a behavior graph, an xpST author begins to construct the task sequence and goal-nodes in xpST pseudo-code. In both cases, these authoring steps are a reflection of the tasks and knowledge components that the author is indicating as needed in order for a learner to do the task. ASPIRE has the author identify those tasks upfront, before the author creates the examples, based on the ontology that the author creates. SimStudent's induction of the task's rules depends on the representation being used, so the author's CTA is important in shaping what the learned rules will look like.

After the author has created the first version of the tutor and students have gone through its instruction, some of these systems have features that enable the authors to iterate the design of the tutor using student log files to inform a CTA and a redo of the tutor. ASSISTments, CTAT, and SimStudent all have robust ways for researchers and teachers to examine learner responses and adapt their tutor's instruction accordingly. ASSISTments produces a report showing learners' most common wrong answers, and also allows students to comment on the problems. SimStudent has a tool to validate its cognitive model by model-tracing through student log data to ensure correct functioning. Initial work has shown that this improves the quality of the model, though additional work will have to be performed to see how much it improves student learning.

Recommendations for GIFT

Having reviewed the challenges and benefits of example-based tutor authoring, we offer suggested features for GIFT so that it may also benefit from this approach. We begin with a brief summary of its architecture from the authoring perspective. GIFT is closer to ASPIRE than the other tools, in that its tutors can be viewed as a collection of states to be reached or constraints to be satisfied, without a particular procedural order to be followed. While sequencing can be achieved through conditions and subconditions, GIFT's core is designed around states. In particular, in GIFT's domain knowledge file (DKF) editor, typically used for authoring, there are tasks, which have concepts, which, in turn, have conditions, which, in turn, trigger feedback (Figure 10). The tasks are collections of states to be achieved. The concepts (with possible subconcepts) are analogous to learner skills used. Concepts are designed as learned if their conditions and subconditions are met. There is not a specific analogy to a procedural step that a learner might take, but a DKF condition is similar. If a step is taken, a condition is likely met. Note that the feedback assigned to be given when a condition is met is chosen from a menu of possible feedback items. Thus, a given feedback item can be reused easily by the author in multiple conditions.

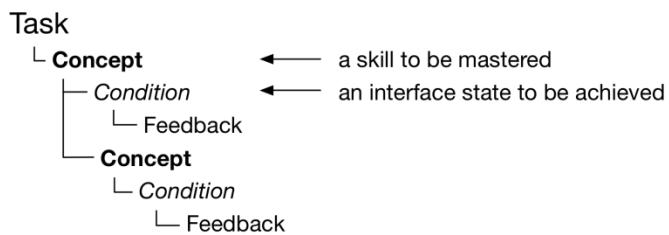


Figure 10: GIFT's DKF format, typically used for authoring

Conditions might be based on whether a certain time has passed in the simulation, or whether the learner has reached a specific location or state. At this early point within GIFT's development, there is not any way to combine conditions in its DKF authoring module, e.g., if the learner does X (Condition 1) while

also in location Y (Condition 2), then perform a particular action. However, it does have an additional authoring tool, SIMILE, that works more like a scripting engine in which authors write explicit if...then code, in which this is possible.

In GIFT there is not a natural way to represent a procedural solution path or branching at a decision point such as in CTAT or xPST. Software applications that manage the passage of time (e.g., video editing suites or medical systems monitoring patient data), aka “timeline-navigators” (Rubio, 2014), typically have a timeline and playhead metaphor as part of their user interface. An analogous interface is recommended for GIFT to indicate to the author the current status of the internal condition evaluations, though it would not likely map cleanly onto a linear timeline, since GIFT looks for active concepts and then evaluates their conditions. Whenever conditions are true, they generate feedback, which may accumulate across multiple conditions. While it is feasible with significant management of the conditions to create a sequence with branching points, the underlying architecture does not make this a natural task for an author. Also, GIFT does not differentiate between forms of feedback, such as hints, prompts, or buggy messages based on incorrect answers.

This state-based and less procedural approach makes GIFT much better adapted to tutors on simulations that enable multiple complex states, such as game engines. A 3D game engine scenario, with multiple live player entities and some game-based non-player characters, is difficult to frame as a procedural tutor and is better approached as a network of noteworthy states (Devasani, Gilbert, Shetty, Ramaswamy & Blessing, 2011; Gilbert, Devasani, Kodavali & Blessing, 2011; Sottilare & Gilbert, 2011). Game engines often have level editors that allow almost WYSIWYG editing and scripting by non-programmers. These could be an inspiration for GIFT. However, since GIFT is essentially an abstraction layer used to describe conditions and states within such a system, enabling the author to visualize the learner’s experience within the simulation while simultaneously understanding the current state of the tutor is a complex challenge for which there are not many common user interface precedents. Currently within GIFT, it is difficult to preview and debug the learner’s experience using the tutor or to easily encode a particular example into GIFT. The CTA of a tutoring experience (described above) must first be created separately and then be transformed to match GIFT’s state-based condition architecture. Once authored, this architecture also makes it difficult to conduct quality assurance testing. The condition-based tutor can be complex to test because the author must think through all possible combinations of states that might generate feedback.

In terms of example-based authoring, a given GIFT tutor is essentially one large example; there is no particular mechanism for generalization. However, GIFT is highly modular, so that elements of a given tutor such as the feedback items can be re-used in other tutors. The features of the aforementioned tutoring systems that promote generalization of rules and easy visualization of the learner’s experience via the authoring tool would be ones for GIFT to emulate.

References

- Adams, D. M., McLaren, B. M., Durkin, K., Mayer, R. E., Rittle-Johnson, B., Isotani, S. & Velsen, M. V. (2014). Using erroneous examples to improve mathematics learning with a web-based tutoring system. *Computers in Human Behavior*, 36, 401 - 411. doi:10.1016/j.chb.2014.03.053}
- Aleven, V. (2010). Rule-Based cognitive modeling for intelligent tutoring systems. In R. Nkambou, J. Bourdeau & R. Mizoguchi (Eds.), *Studies in Computational Intelligence: Vol. 308. Advances in intelligent tutoring systems* (pp. 33-62). Berlin, Heidelberg: Springer. doi:10.1007/978-3-642-14363-2_3
- Aleven, V. & Koedinger, K. R. (2013). Knowledge component approaches to learner modeling. In R. Sottilare, A. Graesser, X. Hu & H. Holden (Eds.), *Design recommendations for adaptive intelligent tutoring systems* (Vol. I, Learner Modeling, pp. 165-182). Orlando, FL: US Army Research Laboratory.

- Aleven, V., McLaren, B. M. & Sewall, J. (2009). Scaling up programming by demonstration for intelligent tutoring systems development: An open-access web site for middle school mathematics learning. *IEEE Transactions on Learning Technologies*, 2(2), 64-78
- Aleven, V., McLaren, B. M., Sewall, J. & Koedinger, K. R. (2006). The Cognitive Tutor Authoring Tools (CTAT): Preliminary evaluation of efficiency gains. In M. Ikeda, K. D. Ashley & T. W. Chan (Eds.), *Proceedings of the 8th International Conference on Intelligent Tutoring Systems* (pp. 61-70). Berlin: Springer Verlag.
- Aleven, V., McLaren, B. M., Sewall, J. & Koedinger, K. R. (2009). A New Paradigm for Intelligent Tutoring Systems: Example-Tracing Tutors. *International Journal of Artificial Intelligence in Education*, 19(2), 105-154.
- Aleven, V., McLaren, B. M., Sewall, J., van Velsen, M., Popescu, O., Demi, S. & Koedinger, K. R. (under review). Toward tutoring at scale: Reflections on “A new paradigm for intelligent tutoring systems: Example-tracing tutors.” Submitted to the *International Journal of Artificial Intelligence in Education*.
- Aleven, V., Sewall, J., McLaren, B. M. & Koedinger, K. R. (2006). Rapid authoring of intelligent tutors for real-world and experimental use. In Kinshuk, R. Koper, P. Kommers, P. Kirschner, D. G. Sampson & W. Didderen (Eds.), *Proceedings of the 6th IEEE international conference on advanced learning technologies (ICALT 2006)* (pp. 847-851). Los Alamitos, CA: IEEE Computer Society
- Anderson, J. R. & Pelletier, R. (1991). A development system for model-tracing tutors. In *Proceedings of the International Conference of the Learning Sciences*, 1-8. Evanston, IL.
- 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.
- Baker, R. S. J. d., Corbett, A. T. & Koedinger, K. R. (2007). The difficulty factors approach to the design of lessons in intelligent tutor curricula. *International Journal of Artificial Intelligence and Education*, 17(4), 341-369.
- Blessing, S. B., Devasani, S. & Gilbert, S. (2011). Evaluation of webxpst: A browser-based authoring tool for problem-specific tutors. In G. Biswas, S. Bull & J. Kay (Eds.), *Proceedings of the Fifteenth International Artificial Intelligence in Education Conference* (pp. 423-425), Auckland, NZ. Berlin, Germany: Springer.
- Blessing, S. B., Gilbert, S., Ourada, S. & Ritter, S. (2009). Authoring model-tracing cognitive tutors. *International Journal for Artificial Intelligence in Education*, 19, 189-210.
- Clark, R. E. & Estes, F. (1996). Cognitive task analysis. *International Journal of Educational Research*. 25(5). 403-417.
- Clark, R. E., Feldon, D., van Merriënboer, J., Yates, K. & Early, S. (2007). Cognitive task analysis. In J. M. Spector, M. D. Merrill, J. J. G. van Merriënboer & M. P. Driscoll (Eds.), *Handbook of research on educational communications and technology (3rd ed.)*. (pp. 577-93). Mahwah, NJ: Lawrence Erlbaum Associates.
- Corbett, A. T. & Anderson, J. R. (1995). Knowledge decomposition and subgoal reification in the ACT programming tutor. *Artificial Intelligence and Education*, 1995: The Proceedings of AI-ED 95. Charlottesville, VA: AACE.
- Corbett, A., Kauffman, L., McLaren, 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.
- Cypher, A. (Ed.). (1993). *Watch what I do: Programming by demonstration*. Cambridge, MA: MIT Press.
- Devasani, S., Gilbert, S. & Blessing, S. B. (2012). Evaluation of two intelligent tutoring system authoring tool paradigms: Graphical user interface-based and text-based. *Proceedings of the 21st Conference on Behavior Representation in Modeling and Simulation* (pp. 54-61), Amelia Island, FL.
- Devasani, S., Gilbert, S. B., Shetty, S., Ramaswamy, N. & Blessing, S. (2011). Authoring Intelligent Tutoring Systems for 3D Game Environments. *Presentation at the Authoring Simulation and Game-based Intelligent Tutoring Workshop at the Fifteenth Conference on Artificial Intelligence in Education*, Auckland.
- Gilbert, S. B., Blessing, S. B. & Blankenship, E. (2009). The accidental tutor: Overlaying an intelligent tutor on an existing user interface. In *CHI '09 Extended Abstracts on Human Factors in Computing Systems*.
- Gilbert, S., Devasani, S., Kodavali, S. & Blessing, S. B. (2011). Easy authoring of intelligent tutoring systems for synthetic environments. *Proceedings of the 20th Conference on Behavior Representation in Modeling and Simulation* (pp. 192-199), Sundance, UT.
- Green, T. R. G. & Petre, M. (1996). Usability analysis of visual programming environments: A ‘cognitive dimensions’ framework. *Journal of Visual Languages and Computing*, 7, 131- 174.
- Jarvis, M. P., Nuzzo-Jones, G. & Heffernan, N. T. (2004). Applying Machine Learning Techniques to Rule Generation in Intelligent Tutoring Systems. In J. C. Lester (Ed.), *Proceedings of the International Conference on Intelligent Tutoring Systems* (pp. 541-553). Heidelberg, Berlin: Springer.

- Koedinger, K. R. & Nathan, M. J. (2004). The real story behind story problems: Effects of representations on quantitative reasoning. *The Journal of the Learning Sciences*, 13(2), 129-164.
- Koedinger, K. R., Aleven, V. & Heffernan, N. (2003). Toward a rapid development environment for cognitive tutors. In U. Hoppe, F. Verdejo & J. Kay (Eds.), *Proceedings of the International Conference on Artificial Intelligence in Education* (pp. 455-457). Amsterdam: IOS Press
- Koedinger, K. R., Anderson, J. R., Hadley, W. H. & Mark, M. A. (1997). Intelligent tutoring goes to school in the big city. *International Journal of Artificial Intelligence in Education*, 8, 30-43.
- Koedinger, K. R., Aleven, V., Heffernan, N., McLaren, B. & Hockenberry, M. (2004). Opening the door to non-programmers: Authoring intelligent tutor behavior by demonstration. In J. C. Lester, R. M. Vicario & F. Paraguaçu (Eds.), *Proceedings of seventh international conference on intelligent tutoring systems, ITS 2004* (pp. 162-174). Berlin: Springer.
- Lau, T. A. & Weld, D. S. (1998). Programming by demonstration: An inductive learning formulation *Proceedings of the 4th international conference on Intelligent user interfaces* (pp. 145-152). New York, NY: ACM Press
- Li, N., Matsuda, N., Cohen, W. W. & Koedinger, K. R. (2015). Integrating Representation Learning and Skill Learning in a Human-Like Intelligent Agent. *Artificial Intelligence*, 219, 67-91.
- Lieberman, H. (2001). *Your wish is my command: Programming by example*. San Francisco, CA: Morgan Kaufmann.
- Long, Y. & Aleven, V. (2013). Supporting students' self-regulated learning with an open learner model in a linear equation tutor. In H. C. Lane, K. Yacef, J. Mostow & P. Pavlik (Eds.), *Proceedings of the 16th international conference on artificial intelligence in education (AIED 2013)* (pp. 249-258). Berlin: Springer
- Lovett, M. C. (1998). Cognitive task analysis in service of intelligent tutoring system design: a case study in statistics. In B. P. Goettl, H. M. Halff, C. L. Redfield & V. Shute (Eds.) *Intelligent Tutoring Systems, Proceedings of the Fourth International Conference* (pp. 234-243). Lecture Notes in Computer Science, 1452. Berlin: Springer-Verlag.
- Lovett, M., Meyer, O. & Thille, C. (2008). JIME-The open learning initiative: Measuring the effectiveness of the OLI statistics course in accelerating student learning. *Journal of Interactive Media in Education*, 2008(1).
- Maass, J. K. & Blessing, S. B. (April, 2011). Xstat: An intelligent homework helper for students. Poster presented at the 2011 Georgia Undergraduate Research in Psychology Conference, Kennesaw, GA.
- MacLellan, C., Koedinger, R. K. & Matsuda, N. (2014). Authoring tutors with SimStudent: An evaluation of efficiency and model quality. In S. Trausen-Matu & K. Boyer (Eds.), *Proceedings of the International Conference on Intelligent Tutoring Systems* (pp. 551-560). Switzerland: Springer.
- Martin, B. & Mitrovic, A. (2002). WETAS: a web-based authoring system for constraint-based ITS. In: P. de Bra, P. Brusilovsky and R. Conejo (eds) *Proc. 2nd Int. Conf on Adaptive Hypermedia and Adaptive Web-based Systems AH 2002*, Malaga, Spain, LCNS 2347, 543-546.
- Matsuda, N., Cohen, W. W. & Koedinger, K. R. (2005). Applying Programming by Demonstration in an Intelligent Authoring Tool for Cognitive Tutors *AAAI Workshop on Human Comprehensible Machine Learning (Technical Report WS-05-04)* (pp. 1-8). Menlo Park, CA: AAAI association.
- Matsuda, N., Cohen, W. W. & Koedinger, K. R. (in press). Teaching the Teacher: Tutoring SimStudent leads to more Effective Cognitive Tutor Authoring. *International Journal of Artificial Intelligence in Education*.
- Matsuda, N., Lee, A., Cohen, W. W. & Koedinger, K. R. (2009). A computational model of how learner errors arise from weak prior knowledge. In N. Taatgen & H. van Rijn (Eds.), *Proceedings of the Annual Conference of the Cognitive Science Society* (pp. 1288-1293). Austin, TX: Cognitive Science Society.
- Matsuda, N., Cohen, W. W., Sewall, J., Lacerda, G., & Koedinger, K. R. (2008). Why tutored problem solving may be better than example study: Theoretical implications from a simulated-student study. In B. P. Woolf, E. Aimeur, R. Nkambou & S. Lajoie (Eds.), *Proceedings of the International Conference on Intelligent Tutoring Systems* (pp. 111-121). Heidelberg, Berlin: Springer.
- McLaren, B. M., Adams, D., Durkin, K., Goguadze, G., Mayer, R. E., Rittle-Johnson, B., . . . Velsen, M. V. (2012). To err is human, to explain and correct is divine: A study of interactive erroneous examples with middle school math students. In A. Ravenscroft, S. Lindstaedt, C. Delgado Kloos & D. Hernández-Leo (Eds.), *21st century Learning for 21st Century Skills: 7th European Conference of Technology Enhanced Learning, EC-TEL 2012* (pp. 222-235). Berlin, Heidelberg: Springer. doi:10.1007/978-3-642-33263-0_18
- McLaren, B. M., DeLeeuw, K. E. & Mayer, R. E. (2011a). Polite web-based intelligent tutors: Can they improve learning in classrooms? *Computers & Education*, 56(3), 574-584.
- McLaren, B. M., DeLeeuw, K. E. & Mayer, R. E. (2011b). A politeness effect in learning with web-based intelligent tutors. *International Journal of Human Computer Studies*, 69(1-2), 70-79. doi:10.1016/j.ijhcs.2010.09.001

- Means, B. & Gott, S. (1988). Cognitive task analysis as a basis for tutor development: Articulating abstract knowledge representations. In J. Pstotka, L.D. Massey & S.A. Mutter (Eds.), *Intelligent tutoring systems: Lessons learned* (pp.35-57). Hillsdale, NJ: Lawrence Erlbaum Associates.
- Mitrovic, A. (1998). Experiences in implementing constraint-based modelling in SQL-tutor. In Goettl, B.P., Halff, H.M., Redfield, C.L. and Shute, V.J. (Eds.), *Proceedings of Intelligent Tutoring Systems*, 414-423.
- Mitrovic, A. (2012). Fifteen years of constraint-based tutors: What we have achieved and where we are going. *User Modeling and User-Adapted Interaction*, 22, 39-72.
- Mitrovic, A. & Ohlsson, S. (1999). Evaluation of a constraint-based tutor for a database language, *International Journal of Artificial Intelligence in Education*, 10, 238-256.
- Mitrovic, A., Koedinger, K. R. & Martin, B. (2003). A Comparative analysis of cognitive tutoring and constraint-based modeling. In: Brusilovsky, P., Corbett, A., and de Rosis, F. (Eds.) *Proceedings of User Modelling*, 313-322.
- Mitrovic, A., Martin, B. & Suraweera, P. (2007). Intelligent tutors for all: Constraint-based modeling methodology, systems and authoring. *IEEE Intelligent Systems*, 22, 38-45.
- Mitrovic, A., Williamson, C., Bebbington, A., Mathews, M., Suraweera, P., Martin, B., Thomson, D. & Holland, J. (2011). An Intelligent Tutoring System for Thermodynamics. *EDUCON 2011*, Amman, Jordan, 378-385.
- Muggleton, S. & de Raedt, L. (1994). Inductive logic programming: Theory and methods. *Journal of Logic Programming*, 19-20(Supplement 1), 629-679.
- Nardi, B. A. (1993). *A small matter of programming: Perspectives on end-user computing*. Boston, MA: MIT press.
- Newell, A. & Simon, H. A. (1972). *Human problem solving*. Englewood Cliffs, NJ: Prentice-Hall.
- Ohlsson, S. (1994). Constraint-based student modelling, in *Student modelling: The key to individualized knowledge-based instruction*, 167-189.
- Ohlsson, S. (1996). Learning from performance errors. *Psychological Review*, 103, 241-262.
- Ohlsson, S. & Mitrovic, A. (2007). Fidelity and efficiency of knowledge representations for intelligent tutoring systems. *Technology, Instruction, Cognition and Learning*, 5, 101-132.
- Olsen, J. K., Belenky, D. M., Aleven, V. & Rummel, N. (2014). Using an intelligent tutoring system to support collaborative as well as individual learning. In S. Trausan-Matu, K. E. Boyer, M. Crosby & K. Panourgia (Eds.), *Proceedings of the 12th International Conference on Intelligent Tutoring Systems, ITS 2014* (pp. 134-143). Berlin: Springer. doi:10.1007/978-3-319-07221-0_66
- Olsen, J. K., Belenky, D. M., Aleven, V., Rummel, N., Sewall, J. & Ringenberg, M. (2014). Authoring tools for collaborative intelligent tutoring system environments. In S. Trausan-Matu, K. E. Boyer, M. Crosby & K. Panourgia (Eds.), *Proceedings of the 12th International Conference on Intelligent Tutoring Systems, ITS 2014* (pp. 523-528). Berlin: Springer. doi:10.1007/978-3-319-07221-0_66
- Ostrow, K. & Heffernan, N. T. (2014). Testing the multimedia principle in the real world: a comparison of video vs. Text feedback in authentic middle school math assignments. In *Proceedings of the 7th international conference on educational data mining* (pp. 296-299).
- Rau, M. A., Aleven, V. & Rummel, N. (2015). Successful learning with multiple graphical representations and self-explanation prompts. *Journal of Educational Psychology*, 107(1), 30-46. doi:10.1037/a0037211
- Rau, M. A., Aleven, V., Rummel, N. & Pardos, Z. (2014). How should intelligent tutoring systems sequence multiple graphical representations of fractions? A multi-methods study. *International Journal of Artificial Intelligence in Education*, 24(2), 125-161.
- Rau, M. A., Aleven, V., Rummel, N. & Rohrbach, S. (2013). Why interactive learning environments can have it all: Resolving design conflicts between conflicting goals. In W. E. Mackay, S. Brewster & S. Bødker (Eds.), *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems (CHI 2013)* (pp. 109-118). ACM, New York.
- Razzaq, L. M. & Heffernan, N. T. (2009, July). To tutor or not to tutor: That is the question. In *AIED* (pp. 457-464).
- Razzaq, L., Patvarczki, J., Almeida, S. F., Vartak, M., Feng, M., Heffernan, N. T. & Koedinger, K. R. (2009). The assistance builder: Supporting the life cycle of tutoring system content creation. *Learning Technologies, IEEE Transactions on*, 2(2), 157-166.
- Reed, S. K. & Bolstad, C. A. (1991). Use of examples and procedures in problem solving. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 17, 753-766.
- Ritter, S. & Koedinger, K. R. (1996). An architecture for plug-in tutor agents. *International Journal of Artificial Intelligence in Education*, 7, 315-347.
- Roll, I., Holmes, N. G., Day, J. & Bonn, D. (2012). Evaluating metacognitive scaffolding in guided invention activities. *Instructional Science*, 40(4), 1-20. doi:10.1007/s11251-012-9208-7

- Rubio, E. (2014) Defining a software genre: Timeline navigators. (Unpublished Master's thesis). Iowa State University, Ames, IA.
- Sottilare, R. and Gilbert, S. B. (2011). Considerations for adaptive tutoring within serious games: authoring cognitive models and game interfaces. *Presentation at the Authoring Simulation and Game-based Intelligent Tutoring Workshop at the Fifteenth Conference on Artificial Intelligence in Education*, Auckland.
- Stampfer, E. & Koedinger, K. R. (2013). When seeing isn't believing: Influences of prior conceptions and misconceptions. In M. Knauff, M. Pauen, N. Sebanz & I. Wachsmuth (Eds.), *Proceedings of the 35th Annual Conference of the Cognitive Science Society* (pp. 916-919). Berlin, Heidelberg: Springer. doi:10.1007/978-3-642-39112-5_145
- Suraweera, P., Mitrovic, A. & Martin, B. (2010). Widening the knowledge acquisition bottleneck for constraint-based tutors. *International Journal of Artificial Intelligence in Education*, 20(2), 137-173.
- Suraweera, P., Mitrovic, A., Martin, B., Holland, J., Milik, N., Zakharov, K. & McGuigan, N. (2009). Ontologies for authoring instructional systems. D. Dicheva, R. Mizoguchi, J. Greer (eds.) *Semantic Web Technologies for e-Learning*. IOS Press, (pp. 77-95).
- VanLehn , K. (2006). The behavior of tutoring systems. *International Journal of Artificial Intelligence in Education*, 16(3), 227-265.
- Waalkens, M., Aleven, V. & Taatgen, N. (2013). Does supporting multiple student strategies lead to greater learning and motivation? Investigating a source of complexity in the architecture of intelligent tutoring systems. *Computers & Education*, 60(1), 159-171.
- Westerfield, G., Mitrovic, A. & Billinghurst, M. (2013). Intelligent augmented reality training for assembly tasks. In: H. C. Lane, K. Yacef, J. Mostow, O. Pavlik (Eds.): *Proceedings of the Sixteenth International Conference of Artificial Intelligence in Education*, LNAI 7926, pp. 542-551. Springer, Heidelberg.
- Wylie, R., Sheng, M., Mitamura, T. & Koedinger, K. (2011). Effects of adaptive prompted self-explanation on robust learning of second language grammar. In G. Biswas, S. Bull, J. Kay & A. Mitrovic (Eds.), *Proceedings of the 15th International Conference on Artificial Intelligence in Education, AIED 2011* (pp. 588-590). Springer Berlin Heidelberg. doi:10.1007/978-3-642-21869-9_110

CHAPTER 7 Supporting the WISE Design Process: Authoring Tools that Enable Insights into Technology-Enhanced Learning

Camillia Matuk¹, Marcia C. Linn², and Libby Gerard²

¹New York University; ²University of California, Berkeley

Introduction

Authoring environments not only provide tools to create supports for learning, they can also be opportunities to better understand the role of technology in learning. The key to achieving their dual purpose is to support users in reflective cycles of iterative, evidence-based refinement of learning materials. Doing so can enable users to ask and answer their own questions; encourage them to be more reflective of their instructional and design practices; and increase their awareness of the relationships between technology, learning, and instruction. Ultimately, this leads to improved materials that enhance learning.

This emphasis on supporting design is reflected in Murray's (2003) goals for contemporary digital authoring tools. These include lowering the cost of creating learning materials; involving users in the design of materials; supporting the representation of domain and pedagogical knowledge; facilitating the implementation of effective design principles; enabling rapid testing and refinement of new ideas; and producing materials that are reusable by multiple authors. Together, these goals encapsulate a vision of design that is more accessible, guided, and likely to flourish from the efforts of a community as opposed to those of an individual.

Authoring tools that support design are especially important for inquiry learning environments, which benefit from iterative refinement, customization, and a community of users. This chapter discusses four principles by which the Web-based Inquiry Science Environment (WISE) guides users' design processes. These include (1) providing design tools that are accessible by users with a range of abilities; (2) enabling users to build on the contributions of others; (3) making student data available as evidence to inform iterative refinement; and (4) allowing ways for users to appropriate the system to advance new goals. We end by discussing challenges and future directions for the design of similar authoring tools for inquiry-learning environments. Through examples drawn from the experiences of our network of users, we illustrate how WISE supports a process of design that also enables new understandings of technology-enhanced learning.

Related Research

For some time now, there has been a trend in the field of educational technology design toward thinking of teachers as more than just end-users, but rather as designers of curricula (e.g., Brown, 2009; Brown & Edelson, 2003; Cviko, McKenney & Voogt, 2014; Edelson, 2002). Indeed, the increasing availability of usable technologies means that authoring need no longer be a specialized task relegated only to developers (Dabbagh, 2001), but also one in which researchers and teachers of varying abilities can participate. Authoring moreover allows users to engage directly in design-based research (Murray 2003), which allows them to pose and answer their own questions about technology-enhanced learning; reflect upon their students' and their own teaching and design practices; and directly change the materials of their instruction.

The authoring of learning environments takes many shapes. It can be as minimal as duplicating existing materials and making a few textual edits. It can extend to reordering activities, adding features, and building curriculum and embedded technologies from scratch (Davis & Varma, 2008; Matuk, Linn & Eylon, under review). Such modifications by users, regardless of their extent, help ensure the materials' successful implementation and sustainability beyond the original designer's involvement (McLaughlin, 1976).

Authoring tools for inquiry learning environments are especially crucial. Although the benefits of inquiry learning are widely acknowledged (NRC, 1996, 2000), conducting inquiry in the classroom is challenging and benefits from adequate support (Evans, 2003; Settlage, 2003). Whereas several authoring tools feature all manner of advanced design capabilities, including the ability to create and customize curriculum materials through remixing, adapting, and sharing with other users (Dabbagh, 2001, Murray, 2003), few of these environments explicitly support inquiry learning (Donnelly et al., 2014). Below, we present an authoring environment that enables the principled design of science inquiry learning and instruction.

The Web-based Inquiry Science Environment

WISE (wise.berkeley.edu) is a free, open-source curriculum platform. Integrated tools allow users to author and customize units, manage student progress, and give feedback on students' work. The 20 freely available classroom-tested units—several of which are available in Spanish, Taiwanese, and Dutch, as well as English—cover challenging topics in the middle and high school science standards. These units have been refined through years of design-based research, guided by the Knowledge Integration framework (KI, Linn & Eylon, 2011), a pattern of instruction based in cognitive theories of how students learn. Units guided by KI engage students in a cycle of activities that includes eliciting their prior ideas, adding new normative ideas, distinguishing among and organizing those ideas, and reflecting upon and integrating them into a coherent explanation. WISE has a long history of improving students' science learning and an extensive user network of teachers and researchers (Linn & Eylon, 2011). As of Fall 2014, WISE had more than 10,000 registered teachers and 85,000 registered students worldwide (see wise.berkeley.edu/webapp/pages/statistics.html for live use statistics).

The units available on the WISE website have undergone cycles of review by experts in curriculum development, subject matter, and education research (Linn, Clark & Slotta, 2009). Concurrently, the design of the authoring tools has been iterated as we learn about users' goals and needs. Our observations of teachers implementing units in their classrooms, and our formal and informal conversations with teachers and researchers provide insights into user's authoring needs. Particularly as the usability of technology continues to shift authoring away from the developer and into the hands of the end-user, questions arise regarding how authoring tools might add value by not only supporting users' goals, but also guiding them to follow best practices in design and instruction.

Our work continually aims to balance the pedagogical practices we wish to promote among our users, with their actual observed practices. This can sometimes present tensions for designers, as teachers' decisions are by necessity not always driven by their pedagogical ideals. Indeed, when pressed for time or pressured to cover much content, teachers' instructional strategies can tend to emphasize content delivery rather than scaffold inquiry processes (Bell, 1998; Dabbagh, 2001; Murray, 1998); their decisions can tend to be driven by practical constraints rather than grounded in evidence from students' work (Boschman et al., 2014); and their professional insights can tend to remain static and isolated, and fail to benefit colleagues beyond their local circles.

At the same time, the system could evolve in beneficial ways if it could be constantly informed by and updated according to teachers' expertise.

We contend that authoring tools in support of users' research and design processes add most widespread value when their interaction is dialogic: That is, when use of the tools guides users' pedagogically sound actions, and when users' expertise and insights can be harnessed to refine the tools and extend their benefit to others. Our years of work from this perspective have resulted in the emergence and refinement of four guiding principles underlying WISE's authoring technologies:

- (1) Provide design tools that are accessible by users with a range of abilities.
- (2) Enable users to build on the contributions of others.
- (3) Make student data available as evidence to inform iterative refinement.
- (4) Allow ways for users to appropriate the system to advance new goals.

Discussion

Provide Design Tools that are Accessible to Users with a Range of Abilities

End-users have unique insights into the local needs of their classrooms and can thus design materials with greater relevance to learners than can developers. But not all users have the time nor the expertise to master complicated tools that might otherwise allow them to realize their ideas. Tools that lower the bar for users of all ability levels can make authoring accessible to a wide audience (Murray, 2003).

Authoring Tools for Customization

WISE makes design accessible to a range of users by providing tools that facilitate the creation and customization of materials without requiring programming skills. Units are organized as sequences of steps contained within activities (Figure 1). In building a unit, authors may iterate between creating and sequencing these nested containers in order to define the flow of tasks, and populating these with content. Users define individual step types from a drop-down menu, which include an array of question and response formats, such as multiple choice, open response, object sequences, drawings, concept diagrams, annotated images, data tables, and graphs. Through a what you see is what you get (WYSIWYG) interface, users can create and edit textual content, and embed rich multimedia from various sources, including web-based applications such as simulations, video, images, animations, and interactive multimedia. These customizations are displayed in real time within a preview mode, which allows users to test the appearance and functionality of their work from the student's point of view.

The screenshot displays the WISE Authoring Tool interface. At the top, a navigation bar includes links for Exit to Home, Open Project, Create Project, Copy Project, Save Project, Manage Art/Files, Edit Info, Preview Project, and EXTRAS. Below the bar, a project title "What makes a good cancer medicine? Ob" is shown with an ID of 12012, and a "Simple Mode Toggle" button. A toolbar below the title provides options for Number Steps By, Step Term, Step Term (Plural), Steps, Logging Level (set to High All Steps), and Hide Steps.

The main area is titled "Project Sequence (Active Activities & Steps)". It lists two activities:

- Activity 1: What is Cancer?** (7 Steps)
 - 1.1: Meet the scientist! (Icon: Introductory Page, Max Score: 0)
 - 1.2: Your ideas about cancer (Icon: Idea Basket, Max Score: 0)
 - 1.3: What is mitosis? (Icon: Introductory Page, Max Score: 0)
 - 1.4: Fast and slow dividers (Icon: Match & Sequence, Max Score: 10)
 - 1.5: Why do some cells divide fast and others slowly? (Icon: Reflection Note (pc), Max Score: 10)
 - 1.6: A definition of cancer (Icon: Survey 1, Max Score: 10)
 - 1.7: How fast do cancer cells divide? (Icon: Drawing, Max Score: 10)
- Activity 2: The Phases of Cell Division** (24 Steps)
 - 2.1: Interphase: When cells don't divide (Icon: Introductory Page, Max Score: 0)
 - 2.2: Observing cells through the microscope (Icon: Annotator, Max Score: 0)
 - 2.3: Putting the picture together (Icon: Drawing, Max Score: 10)
 - 2.4: When cells divide (Icon: Reflection Note (pc), Max Score: 10)
 - 2.5: Do you agree with your partner? (Icon: Reflection Note (pc), Max Score: 10)
 - 2.6: The phases of cell division: Scientists (Icon: Survey 1, Max Score: 0)

Below the sequence, there are settings for Editor View (Simple, Advanced, Refresh As I Type), Refresh Now, Edit Hints, and buttons for Save, Save & Close, and Close. On the left, configuration options include shuffle answers before next try (with radio buttons for shuffle choices or do NOT shuffle choices), feedback options (with radio buttons for has inline feedback or does NOT have inline feedback), and a field for hiding questions and answers after they are answered correctly. There is also a text input for the number of correct answers allowed (set to 1).

The right side shows a "CHALLENGE QUESTION" window titled "CHECK YOUR UNDERSTANDING". It features a cartoon character, a question about metaphase events, and a sidebar with a robot icon. The "ANSWERS" section lists four options for what happens in metaphase. At the bottom, it says "This is attempt #1".

Figure 1. These screenshots from the WISE authoring interface show the sequence of steps and activities, which can be created, copied, imported, and reordered (top); and the WYSIWYG editing interface that supports text and layout edits, automated feedback, and embedding of rich multimedia.

Becoming familiar with the tools to make these and more complex customizations requires new users less than 1 hour of training. The time to actually perform those customizations can vary greatly between both novice and expert authors, but depends mainly on authors' familiarity with the content, clarity of goals, and commitment to a design strategy for achieving those goals. For example, minor edits to text, embedded media, and page formatting, can take just a few minutes to perform and can even be done in the midst of students' work on the unit. More complex authoring tasks, however, such as designing activity sequences, creating new content, and integrating scaffolding tools, require careful alignment of

designs to some pedagogical framework. In these cases, it is an author's general experience having authored curricula that determines the time costs rather than having authored with particular WISE tools. Among the core WISE research and design team, these more elaborate designs go through cycles of review, feedback, and iteration by curriculum designers, content experts, educators, and technologists (Slotta & Linn, 2009). It may take just several hours to lay the foundation for a new curriculum unit, but it may take weeks, months, and years to continue to refine it.

The extent to which teachers use these tools to customize depends on various factors. These include practical considerations, as well as teachers' attitudes toward technology, assumptions about learning, and views on their roles as educators (Luehmann, 2002). Some teachers have independently learned to use the authoring environment to modify the content of given units for their particular classroom needs. One middle school teacher, for example, used the text editing tools to tailor the prompts in a grade 7 WISE unit about cell division. Knowing the specific comprehension difficulties of her mainly English-language learners, she elaborated on the instructions and incorporated hints, keywords, and sentence starters to guide her students' responses (Matuk, Linn & Eylon, 2015).

With the proper kind of support, these authoring tools allow teachers to effect powerful changes in their instruction. During summer professional development workshops, for instance, teachers worked in groups under the guidance of WISE researchers. Using the authoring tools they made customizations to units, which subsequently led to improved student learning (Gerard, Spitulnik & Linn, 2010). In this case, teachers benefited from having time to work with another teacher who taught the same unit during the prior school year and examine their students' work to inform customizations. Together, the teachers shared their classroom experiences with one another and identified places in the unit where students had difficulty. The teachers then examined their students' work on an embedded assessment in one of these challenging spots and students' work on a pre/posttest. Based on their classroom experiences and analysis of their students work, teacher negotiated changes for the unit. Importantly, researchers were present to provide technology support and insights on best practices in inquiry learning design.

Authoring Tools for Research

The ease with which units can be created and modified affords the rapid testing of ideas, and thus, their use as instruments for research. Researchers often use WISE's more complicated authoring functions to construct design experiments to investigate how students learn from technology-enhanced materials (Murray, 2003). For example, users can incorporate input to the WISE interface from hardware such as light, temperature, and motion probes. By checking boxes, users can specify navigation constraints dependent on students' responses. By selecting steps from a list, they can define nonlinear trajectories through a unit. Within the authoring interfaces of certain items, users can also compose conditional automated feedback directly beside students' possible responses. For other items, users can specify keywords that, if they were to appear in students' open responses, would trigger certain kinds of feedback to be delivered to students.

Using these capabilities, researchers have designed and implemented alternative versions of the same unit to investigate the value of different instructional approaches. They have explored the impacts of different kinds of automated feedback on how students revise their drawings (Rafferty, Gerard, McElhany & Linn, 2013), concept diagrams (Ryoo & Linn, 2014), graphs (Vitale, Lai & Linn, 2014), and open-ended responses (Liu, et al., 2014). They have compared ways of integrating new collaborative technologies into existing curriculum (Matuk & Linn, 2014), scaffolding students' understanding of visualizations (Chang, et al., 2008; Zhang & Linn, 2011), and supporting students' interpretations of visual evidence (Matuk & McElhaney, 2014).

The availability of several content-free scaffolding tools has allowed researchers to author units across subject matters in order to investigate their own questions about learning. The Idea Manager, for instance, a tool that helps students track and share ideas over the course of a unit, has been used to study the development of students' ideas about chemistry (McElhaney et al., 2013) and astronomy (Matuk & King Chen, 2011); and understand the value of exchanging ideas when studying the life sciences (Matuk & Linn, 2014; Wichmann et al., 2014). Likewise, the Image Annotator, a tool that allows students to label static and animated visuals, has been used to scaffold students' observations in chemistry, physics, and cell biology (Matuk & McElhaney, 2014). In another example, the concept diagramming tool, MySystem, has been used to study students' understanding of energy in physics (Swanson, 2010) and biology (Ryoo & Linn, 2010).

Thus, tools that facilitate creation and customization lower the bar for users of all abilities. They make it easy for teachers to adapt materials to their particular needs, and they enable researchers to rapidly build and test ideas in order to investigate questions about learning with technology-enhanced materials. In these manners, authoring tools allow the educational environment to become a platform for research as much as a platform for learning and instruction.

Enable Users to Build on the Contributions of Others

Another way that authoring is made more accessible is through the availability of existing resources. The ability to make use of an array of existing, pre-constructed artifacts offloads much of the workload from authoring, which allows teachers to focus on teaching and researchers to focus on research.

Indeed, while it is possible to author units from scratch, most users build upon existing, freely available materials. Authors can search for and clone any publicly available classroom-tested unit, any of their own privately owned units, and any unit directly shared with them by other users. They may then use these materials as templates for their own work, importing whole activities or individual steps, along with the existing resources contained within them. These resources include embedded multimedia, page layouts, investigation narratives, and assessment items. Tools for inserting, editing, and reordering allow users to easily remix various given materials for new purposes.

The ability to thus copy and modify units takes advantage of a community's contributions to make design more efficient (Recker et al., 2007). One middle school unit on global climate change, for example, has undergone multiple iterations by different generations of WISE researchers. As new users copied the unit, they made modifications to the embedded models and scaffolds: They added details to the content and even re-crafted the narrative to present the ideas from different angles. One version of the unit, for example, focuses on how the transfer of energy affects the Earth's temperature, while another examines the chemical reactions behind the greenhouse effect, and a third version introduces the notion of feedback loops as an explanation for climate change. It is also possible to merge elements from different units. This allows authors to quickly create entirely new activity sequences by combining existing tested material, and then modify these for coherence.

This ability to build on existing materials has moreover permitted systematic refinements to units on subsequent classroom implementations. Svhla and Linn (2012), for instance, made multiple iterations on their version of the Global Climate Change unit. Small adjustments made with each classroom implementation helped to clarify the visualizations, distinguish between concepts, and add structure to students' experimentation with a NetLogo simulation. Ultimately, their iterations produced a version of the unit that resulted in higher learning gains.

The ability to remix existing resources by copying and modifying has also allowed researchers new to WISE to quickly design and implement their own research projects. In a period of just several weeks, one

visiting researcher appropriated a middle school unit on photosynthesis as the context for a study on collaborative learning. Maintaining the unit's original simulations, she crafted a new inquiry narrative, integrated a collaborative tool to scaffold students' exchanging ideas with their peers, and analyzed students' learning based on existing assessment items (Wichmann, et al., 2014).

A strength of WISE is that it allows users to draw from the vast resources available online to compose coherent and personally relevant investigations (Linn, 2000; Linn & Hsi, 2000). By enabling users to draw on a user community resource of shared materials, WISE's authoring environment aids the refinement of existing designs, encourages the initiation of new research, and increases the variety of existing materials available for others' use.

Make Student Data Available as Evidence to Inform Iterative Refinement

Making student work accessible means it can be used as evidence for identifying refinements and customizations. Indeed, research finds that curriculum customizations informed by students' work result in greater learning gains than typical refinements that rely on teacher insights (Ruiz-Primo & Furtak, 2007). By making student evidence accessible in many formats and from various outlets, WISE enables researchers and teachers to readily use it to guide their revisions or customizations.

For example, most teachers and researchers make use of the grading interface's basic facilities for displaying class progress through a unit. These simple bar graphs show the percentage of students who have completed individual steps in the unit, as well as the percentage of the unit completed by individual students. With this information, users can make general pacing decisions that include adjusting allocated class time on subsequent implementations. Users may also obtain a snapshot view of students' ideas by browsing submitted responses, filtering these along different dimensions (class period, step in the unit), and viewing them by individual student or by step in the unit. Most teachers use the "grade by step" feature to see a range of responses to the same question at one time and use this information to customize their guidance, class instruction, and the unit accordingly. More experienced users may sort these responses according to various criteria, such as teacher-assigned or computer-automated score, whether the teacher had flagged or commented upon the response during grading, the number of revisions students made, and so forth. They may also view a table of the numbers of students currently working on any given step and the length of time spent there. Such information has been especially useful to those with previous experience implementing a given unit. By observing when critical masses of students either struggle or progress without the benefit of a challenge, teachers can identify where in the unit to adjust their face-to-face guidance, as well as how to modify the unit's embedded scaffolds. Each of the functions above allows users to more closely monitor students' progress and thinking at both the individual and group levels, thus informing appropriate modifications to the design of the materials. The *My Notes* tool available in the grading interface furthermore permits users to document private reflections that remain associated with the unit and useful as reference during subsequent implementations.

Researchers can conduct detailed analyses of students' interactions within the unit by exporting logged data in the form of a spreadsheet. McElhaney and Linn (2011) analyzed logs of students' interactions with a simulation of a car collision in a high school physics unit. By examining the number of students' trials, what variables they chose, and how they varied them, the researchers identified categories of students' approaches to experimentation and the necessary conditions for understanding its nature.

Similarly, Matuk and Linn (2014) used logs of students' uses of the Idea Manger to identify patterns in how middle school students shared ideas during a unit on cell division and the effects of these behaviors on their subsequent explanations of cancer treatment. Given ways to inspect and interpret students'

interactions, users can ensure their design refinements are grounded in evidence from students' work, and thus, avoid making unfounded design decisions.

Allow Users to Appropriate the System to Advance New Goals

Above, we discussed how WISE's authoring tools allow units to be adapted to individuals' local needs. However, there are also tools that allow users to tailor the platform itself for broader audiences and goals. For example, international researchers have employed WISE's translation tools to adapt versions of units from WISE's public library in Spanish, Taiwanese, and Dutch. These translated units have been featured in teacher professional development workshops and used by students and teachers in Europe, Asia, and South America (e.g., Rizzi et al., 2014). They have also served as platforms for users at other academic institutes to pursue research programs of their own (e.g., Raes, Schellens & de Wever, 2013).

New technologies can also be tested within WISE, given the ability to integrate third-party technologies. These can include virtual models from Molecular Workbench (Xie et al., 2011) and NetLogo (Wilensky, 1999), which are themselves free, open-source, and customizable. Users can also embed technologies of their own within WISE's existing step types.

This is how the Image Annotator was developed: as a basic working version of a tool—programmed in Actionscript—that allowed students to directly label existing static and animated graphics. Findings from subsequent classroom pilot tests (Matuk & Linn, 2013, Matuk & McElhaney, 2013) prompted users to request further features, which led WISE developers to create a dedicated Annotator step based on the initial prototype. The latest version features authorable elements, including user- rather than developer-defined images and label colors, prompts for students to elaborate written explanations of their labels, constraints on the number of labels required, and automated scoring and feedback dependent on students' responses. Thus, the result of WISE allowing the embedding of user-contributed technologies and ways to easily test them in classrooms led to the development of a new authorable tool usable by a wide audience of users both to support and research student learning.

WISE's open-source license has attracted an even broader base of users, who by their collective efforts, improve and enrich the system for others. Users have set up unique instances of WISE on their own servers and are tailoring it for new purposes. At Northwestern University, for example, the Center for Connected Learning (CCL) and Computer-Based Modeling, led by Uri Wilensky, has chosen WISE as a platform for delivering a curriculum of NetLogo/HTML5 simulations to teach complex systems. Their choice was based in their survey of contemporary learning management systems, which found WISE to be the most fully featured and capable as a platform to support the delivery and data-logging of their technologies (Wilensky, personal communication). Another research group led by Douglas Clark at Vanderbilt University has built a novel curriculum-integrated game engine within WISE called SURGE (www.surgeuniverse.com) and uses it to investigate how games can teach formal concepts of Newtonian mechanics (Clark et al., 2011). Meanwhile, Jennifer Chiu at the University of Virginia has re-skinned the WISE platform for a curriculum on engineering design (Chiu & Linn, 2011).

These examples illustrate the value of maintaining WISE on an open-source license. In doing so, improvements to the system evolve from the expert contributions of a distributed community of users (Kogut & Metiu, 2001), as uncoordinated developers can propose and select changes that optimize the system over time (Axelrod & Cohen, 2000).

Recommendations and Future Research

This chapter discussed four principles behind the design of WISE’s authoring environment that enable users to engage in design-related activities for teaching and research. We specifically described how tools that enable efficient creation and customization can help lower the bar for design by users of all abilities, and empower them to ask and answer their own questions about technology, learning, and instruction. We described how the ability to draw upon and remix shared, pre-constructed elements encourages refined curriculum designs and supports systematic design-based research. We discussed how making student data available for users to inspect and query can inform design revisions, as well as make visible new insights into student learning. Finally, we described how the open sourcing of WISE has encouraged other researchers to use it as a platform for new research programs. Ultimately, users’ contributions enhance the usefulness of the system for others.

The examples from WISE illustrated how features of authoring environments can support efficient testing and iteration of new learning tools, materials, and ideas. In doing so, users can be free to ask and answer their own questions, and to make direct modifications based on their observations—behaviors that encourage reflective educational practice, and contribute to research insights. Together, these outcomes can lead to powerful impacts on students’ learning.

Below, we highlight three remaining questions derived from this work and discuss opportunities for future development.

How Do We Design Tools that Guide Pedagogically Sound Design?

While contemporary authoring environments permit considerably more freedom to design outside set patterns and structures than did the earlier computer-based instruction systems (Dabbagh, 2001), they also permit designs that veer from tested pedagogical approaches. Indeed, a number of commercially available authoring environments support the creation of visually appealing materials. However, when these environments are not founded on pedagogically oriented design principles, they invite ineptly made, text-heavy drill-and-practice tasks and few inquiry-oriented activities (Bell, 1998; Dabbagh, 2001; Murray, 1998). This is especially true when teachers feel pressured to cover large amounts of content. A challenge for developers of authoring environments is thus to balance the tension between allowing users the freedom to design, while also providing the guidance needed to produce the most effective designs.

Our experiences suggest the most effective guidance to be in face-to-face support, whether this occurs in informal interactions among fellow teachers and researchers, or organized professional development. However, features within the authoring environment itself add value by offering timely, in-the-moment guidance during users’ independent work. In WISE, guidance is implicit in the pre-constructed resources available to authors, which reflect the underlying Knowledge Integration pedagogy. Existing classroom-tested units exemplify successful instructional patterns (e.g., predict-observe-explain, response-feedback-revision, faded scaffolds); and when cloned, these serve as templates upon which new authors can build. Integrated tools, such as the Idea Manager, are explicit in breaking down the process of eliciting, organizing, distinguishing, and reflecting upon ideas; and when integrated into a unit, they scaffold students through these steps. Contrary to the media primitives (e.g., buttons, menus, icons) that characterize other authoring environments (see Mulholland et al., 2011), these pedagogical primitives make transparent an underlying pedagogy with assumptions about how students learn and what makes effective instruction (Murray, 2003).

But how can we ensure that users are actually building upon successful instructional patterns and avoiding the lethal mutations that occur when customizations detract from the goals of the original design

(Haertel, cited in Brown and Campione, 1996)? The solution may be to explore ways to integrate guidance on effective pedagogical and instructional approaches into various stages of the design process (Dabbagh, Bannan-Ritland & Silc, 2000).

To aid in the planning stages, WISE might encourage users to thoughtfully approach their instruction by providing tools to conceptually map the flow of activities and associated resources (cf. learning activity management system (LAMS), CADMOS, *Learning Designer*, etc., cited in Conole, 2013). Learning tools and activity templates might be explicitly connected to outcomes, such that users might select patterns according to the learning goals they wish to target (e.g., incorporate the Image Annotator tool to develop students' observational skills; use a predict-observe-explain task to structure students' approaches to experimentation). "Running" the design would result in an evaluation of its predicted success and offer recommendations for activities, tools, and resources to optimize the design for given constraints (e.g., available class time, percentage of English language learners, etc.) and better align it with the goals of inquiry.

While authoring, shared artifacts might contain embedded guidance in the form of annotations. These could be contributed by designers, education researchers, and experienced educators; and would offer authors insights into design rationales and best practices for their use (cf. educative curriculum materials, Davis & Krajcik, 2005; Davis & Varma, 2008).

Finally, ready access to a database of instructional design principles would allow users guidance on-demand (e.g., Kali, 2006). Integrating these or similar solutions into the authoring process may help balance users' freedom to design with guidance for making pedagogically-sound design decisions.

How Can We Create Authoring Communities that Also Enable the System to Evolve?

As discussed, WISE authors benefit greatly from the ability to build upon others' contributions. Although WISE maintains a public database of classroom-tested units, there is currently no way for users to make their own creations publicly accessible except by directly sharing individual units with known users. Supporting an open marketplace of artifacts has the potential to allow individuals to build on one another's past successes on a large scale (see Morris & Heibert, 2011; Recker et al. 2007). However, unsupervised exchange also risks introducing contributions that are not aligned with practices known to be successful.

Whether and how to curate users' contributions is both a democratic issue, as well as a logistical one. Allowing users to freely exchange artifacts can foster the social interactions conducive to learning (Lave & Wenger, 1991; Lerner, Levy & Wilensky, 2010; Vygotsky, 1978; Wenger, 1998), and enrich the variety of contributions upon which others might draw. At the same time, an open marketplace of artifacts might decrease its perceived authority, as it would no longer be a resource of tested, theory-based materials. Yet, curation of such a repository would be costly to maintain. It would require the long-term commitment of a central individual or group of curators, as well as an effective system for passing down knowledge of the system to subsequent members.

One compromise is for a consortium of curators to maintain a subset of tested materials separate from a library for open exchange (cf. Lerner, Levy & Wilensky, 2010). This would offer users both the value of a trustworthy resource of materials alongside the value of social interactions and richness of community contributed artifacts. The impact of each for supporting high quality authoring needs to be explored.

How Can We Tap into the Vast Amounts of Logged Data to Support Authoring and Encourage Looking at Students' Ideas?

Logged data can be valuable for informing authoring decisions. WISE can track fine-grained data on students' interactions in WISE, from their responses and revisions within units, to the grades and feedback received across units and years. Some of this information is accessible through the teacher tools, which teachers use to inform their current and future instruction (e.g., when to give feedback on particular items, and to whom). Similarly, the authoring system might channel automated scores and other logged data to inform specific design decisions. For instance, information documented on past students' performance on particular items might inform authors of areas requiring revision. Archives of students' typical ideas on certain topics, and even records of the average time spent completing particular activities, might help authors see where more or less emphasis would benefit students' understanding.

These data, along with the annotations and templates contributed by other users, might fuel a recommender system to guide authors in following sound instructional design principles and tailoring their designs toward particular needs and goals (Dabbagh, 2000). A further question then becomes how to design a dashboard that displays these data in ways that make them accessible. What data are most appropriate and how are they best visualized to guide authors' design tasks?

In sum, supporting educators and researchers in designing learning environments, especially inquiry-based ones, can encourage more reflective practice, and ensure the long-term sustainability of materials. This goal is met when authoring tools follow principles that are sensitive to the design issues faced by researchers and educators.

Design Implications for GIFT

WISE shares many features in line with the design goals of the Generalized Intelligent Framework for Tutoring's (GIFT) authoring construct. Among others, these include tools that decrease the level of effort and skill necessary to author; allow integration of external media; facilitate rapid prototyping and testing; enable reuse and adaptation of materials; and rely on an open-source model of development of maintenance.

Two characteristics of WISE's efforts might inform the design of GIFT authoring tools. One is that WISE devotes many resources to creating and making available ready-to-use curriculum materials. Because the teachers that WISE targets have little time to devote to creating their own curriculum materials, let alone to learning to use authoring tools, the provision of existing materials encourages and sustains their engagement, and guides their practice. Units that address specific topics in middle and high school science curricula draw teachers to use WISE, and provide classroom-tested seed material upon which they can build when authoring their own customizations (Matuk, Linn & Eylon, 2015).

In seeking similar resources, users self-select as members of a community with specific shared goals. This allows WISE to build targeted support. Regularly scheduled gatherings bring WISE users together around mutual questions and challenges. These allow researchers and teachers to exchange insights about teaching, learning, and technology, and so better understand and appreciate one another's complementary roles.

This is related to a second characteristic of WISE, which is that through the concurrent nurturing of a community of users, teachers are both mentored in their use of the tools, as well as given voice to the shape those tools. In-class assistance and online support from WISE researchers, as well as regular organized professional development, serve to orient teachers who are new to WISE. They ensure smooth

and positive curriculum implementation experiences with the expectation that this initial guidance will build teachers' confidence to author their own customizations for subsequent implementations.

Teacher-researcher partnerships are moreover opportunities for teachers to contribute to designing and refining the tools of their own practice. Over the years, WISE has developed methods for eliciting teachers' insights, and a commitment to incorporating these into new iterations of its technologies (Matuk et al., 2015). An approach that thus privileges the voice of a user community ensures that the authoring environment is not prescriptive of researchers' theoretical ideals. Instead, these ideals continually evolve with practitioners' needs and goals, which, in turn, shape and drive the design of the tools.

In conclusion, WISE has found that its success lies beyond merely providing a comprehensive set of tools to meet the anticipated demands of its users. It also relies upon providing multiple channels of support and communication among its researchers and teachers. These ensure that the tools remain responsive to users' changing needs and are also used to their greatest value.

References

- Axelrod, R. and Cohen, M. (2000), Harnessing Complexity: Organizational Implications of a Scientific Frontier, New York, Free Press.
- Boschman, F., McKenney, S. & Voogt, J. (2014). Understanding decision making in teachers' curriculum design approaches. *Educational Technology Research and Development* 62(4), 393–416.
- Brown, A. L. & Campione, J. C. (1996). Psychological theory and the design of innovative learning environments: On procedures, principles, and systems. In R. Glaser (Ed.), *Innovations in learning: New environments for education* (pp. 289–325). Mahwah, NJ: Erlbaum.
- Brown, M. & Edelson, D. (2003). Teacher as design. (Design brief). LeTUS, Evanston, IL.
- Chiu, J. L. & Linn, M. C. (2011). Knowledge integration and WISE engineering. *Journal of Pre-College Engineering Education Research (J-PEER)*, 1(1), 2.
- Clark, D. B., Nelson, B., Chang, H., D'Angelo, C. M., Slack, K. & Martinez-Garza, M., (2011). Exploring Newtonian mechanics in a conceptually-integrated digital game: Comparison of learning and affective outcomes for students in Taiwan and the United States. *Computers and Education*, 57(3), 2178-2195.
- Conole, G. (2013). Tools and resources to Guide Practice. In H. Beetham & R. Sharpe (Eds.), *Rethinking Pedagogy for a Digital Age: Designing for 21st Century Learning* (pp. 78-101). New York: Routledge.
- Cviko, A., McKenney, S. & Voogt, J. (2014). Teacher roles in designing technology-rich learning activities for early literacy: A cross-case analysis. *Computers & Education*, 72, 68-79
- Dabbagh, N. H., Bannan-Ritland, B. & Silc, K. (2000). Pedagogy and Web-based course authoring tools: Issues and implications. *Web-based training*, 343-354.
- Davis, E. A. & Krajcik, J. S. (2005). Designing educative curriculum materials to promote teacher learning. *Educational researcher*, 34(3), 3-14.
- Davis, E. A. & Varma, K. (2008). Supporting teachers in productive adaptation. In Y. Kali, M. C., Linn & J. Roseman (Eds.), *Designing coherent science education: Implications for curriculum, instruction, and policy* (pp. 94-122). New York, NY: Teachers College Press.
- Donnelly, D. F., Linn, M. C. & Ludvigsen, S. (2014). Impacts and Characteristics of Computer-Based Science Inquiry Learning Environments for Precollege Students. *Review of Educational Research*. Retrieved from <http://rer.sagepub.com/cgi/doi/10.3102/0034654314546954>
- Evans, C. (2003, January). Challenges to successful science inquiry: Finding unifying themes in the multivariate nature of inquiry models. Paper presented at the annual meeting of the National Association for Research in Science Teaching, Philadelphia.
- Gerard, L. F., Spitulnik, M. & Linn, M. C. (2010). Teacher use of evidence to customize inquiry science instruction. *Journal of Research in Science Teaching*, 47(9), 1037-1063.
- Kali, Y. (2006). Collaborative knowledge building using the Design Principles Database. *International Journal of Computer-Supported Collaborative Learning*, 1(2), 187-201.

- Lerner, R., Levy, S.T. & Wilensky, U. (2010, August 10-14). *Encouraging Collaborative Constructionism: Principles Behind the Modeling Commons*. In J. Clayson & I. Kalas (Eds.), Proceedings of the Constructionism 2010 Conference. Paris, France.
- Luehmann, A. L. (2002). Understanding the appraisal and customization process of secondary science teachers. Paper presented at the annual meeting of the American Educational Research Association: New Orleans, LA.
- Linn, M. C. (2000). Designing the knowledge integration environment. *International Journal of Science Education*, 22(8), 781-796.
- Linn, M. C., Clark, D. and Slotta, J. D. (2003), WISE design for knowledge integration. *Sci. Ed.*, 87: 517–538. doi: 10.1002/sce.10086
- Linn, M. C. & Eylon, B. S. (2011). *Science learning and instruction: Taking advantage of technology to promote knowledge integration*. Routledge.
- Linn, M. C. & Hsi, S. (2000). *Computers, teachers, peers: Science learning partners*. Routledge.
- Liu, O. L., Brew, C., Blackmore, J., Gerard, L., Madhok, J. & Linn, M. C. (2014). Automated Scoring of Constructed-Response Science Items: Prospects and Obstacles. *Educational Measurement: Issues and Practice*, 33(2), 19-28.
- Matuk, C. F. & King Chen, J. (2011). The WISE Idea Manager: A Tool to Scaffold the Collaborative Construction of Evidence-Based Explanations from Dynamic Scientific Visualizations. In, *Proceedings of the 9th International Conference on Computer Supported Collaborative Learning CSCL2011: Connecting computer supported collaborative learning to policy and practice, July 4-8, 2011*. The University of Hong Kong, Hong Kong, China.
- Matuk, C. F. & Linn, M. C. (2013, April 27 - May 1). *Technology Integration to Scaffold and Assess Students Use of Visual Evidence In Science Inquiry*. Paper presented at the American Educational Research Association Meeting (AERA2013): Education and Poverty: Theory, Research, Policy and Praxis, San Francisco, CA, USA.
- Matuk, C. & Linn, M. C. (2014). Exploring a digital tool for exchanging ideas during science inquiry. In *ICLS'14: Proceedings of the 11th International Conference for the Learning Sciences*, Boulder: International Society of the Learning Sciences.
- Matuk, C., Gerard, L., Lim-Breitbart, J. & Linn, M. C. (2015, April 16-20). *Gathering Design Requirements During Participatory Design: Strategies for Teachers Designing Teacher Tools*. Paper presented at the American Educational Research Association Meeting, Chicago, IL, USA.
- Matuk, C., Linn, M. C. & Eylon, B.-S. (2015). Technology to support teachers using evidence from student work to customize technology-enhanced inquiry units. *Instructional Science*, 43(2), 229-257. DOI: 10.1007/s11251-014-9338-1
- Matuk, C. & McElhaney, K. (2014, April 3-7). Investigating a Digital Annotation Tool for Distinguishing Visual Evidence in Science Inquiry. Paper presented at the American Educational Research Association Meeting, Philadelphia, PA, USA.
- McLaughlin, M. W. (1976). Implementation as mutual adaptation: Change in classroom organization. *Teachers College Record*, 77: 339–351.
- Morris, A. K. & Hiebert, J. (2011). Creating Shared Instructional Products An Alternative Approach to Improving Teaching. *Educational Researcher*, 40(1), 5-14.
- Murray, T. (2003). An Overview of Intelligent Tutoring System Authoring Tools: Updated analysis of the state of the art. In *Authoring tools for advanced technology learning environments* (pp. 491-544). Springer Netherlands.
- Murray, T. (1998). Authoring knowledge-based tutors: Tools for content, instructional strategy, student model, and interface design. *The Journal of the Learning Sciences*, 7(1), 5-64.
- National Research Council. (1996). National science education standards. Washington, DC: National Academies Press.
- National Research Council. (2000). Inquiry and the national science education standards: A guide for teaching and learning. Washington, DC: National Academies Press.
- Kogut, B. & Metiu, A. (2001). Open-source software development and distributed innovation. *Oxford Review of Economic Policy*, 17(2), 248-264.
- Raes, A., Schellens, T. & De Wever, B. (2013). Web-based Collaborative Inquiry to Bridge Gaps in Secondary Science Education. *Journal of the Learning Sciences*, 23(3), 316-347.
- Rafferty, A. N., Gerard, L., McElhaney, K., Linn, M. C. (2013). Automating Guidance for Students' Chemistry Drawings. *Proceedings of Formative Feedback in Interactive Learning Environments* (AIED Workshop).

- Rizzi, Iribarren, C., Furman, M; Podestá, M.E & Luzuriaga, M. (2014, November 12-14) Diseño e implementación de la plataforma virtual de aprendizaje WISE en el aprendizaje de las Ciencias Naturales. Congreso Iberoamericano de Ciencia, Tecnología, Innovación Y Educación, Buenos Aires, Argentina.
- Ruiz-Primo, M. A. & Furtak, E. M. (2007). Exploring teachers' informal formative assessment practices and students' understanding in the context of scientific inquiry. *Journal of research in science teaching*, 44(1), 57-84.
- Ryoo, K. K. & Linn, M. C. (2010, June). Student progress in understanding energy concepts in photosynthesis using interactive visualizations. In *Proceedings of the 9th International Conference of the Learning Sciences-Volume 2* (pp. 480-481). International Society of the Learning Sciences.
- Ryoo, K. & Linn, M. C. (2014). Designing guidance for interpreting dynamic visualizations: Generating versus reading explanations. *Journal of Research in Science Teaching*, 51(2), 147-174.
- Settlage, J. (2003, January). Inquiry's allure and illusion: Why it remains just beyond our reach. Paper presented at the annual meeting of the National Association for Research in Science Teaching, Philadelphia.
- Slotta, J. D. & Linn, M. C. (2009). *WISE science: Web-based inquiry in the classroom*. Teachers College Press.
- Svihla, V. & Linn, M. C. (2012). A design-based approach to fostering understanding of global climate change. *International Journal of Science Education*, 34(5), 651-676.
- Swanson, H. (2010, June). Eliciting energy ideas in thermodynamics. In *Proceedings of the 9th International Conference of the Learning Sciences-Volume 2* (pp. 254-255). International Society of the Learning Sciences.
- Vitale, J., Lai, K. & Linn, M. C. (2014). Dynamic Visualization of Motion for Student-Generated Graphs. In *ICLS'14: Proceedings of the 11th International Conference for the Learning Sciences*, Boulder: International Society of the Learning Sciences.
- Vygotsky, L. S. (1978). *Mind in society: The development of higher psychological processes*. Cambridge, MA: Harvard University Press.
- Wichmann, A., Matuk, C., Sato, E., Gerard, L., Madhok, J. & Linn, M. C. (2014, August 18-20). Critiquing Peer-Generated Ideas during Inquiry Learning. The Biennial Meeting of the EARLI SIG20 Computer Supported Inquiry Learning, Malmö, Sweden.
- Wilensky, U. (1999). NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.
- Xie, C., Tinker, R., Tinker, B., Pallant, A., Damelin, D. & Berenfeld, B. (2011). Computational experiments for science education. *Science*, 332(6037), 1516-1517.
- Zhang, Z. H. & Linn, M. C. (2011). Can generating representations enhance learning with dynamic visualizations?. *Journal of research in science teaching*, 48(10), 1177-1198.

CHAPTER 8 Authoring Tools for Ill-defined Domains in Intelligent Tutoring Systems: Flexibility and Stealth Assessment

Matthew E. Jacobina, Erica L. Snow, Jianmin Dai, and Danielle S. McNamara
Arizona State University

Introduction

Intelligent tutoring systems (ITSs) provide customized instruction to students by modeling what students need to know and what they seem to know, and by providing adaptive feedback and problem sets based on performance within the system (e.g., Beal, Arroyo, Cohen & Woolf, 2010; Graesser et al., 2004). Building a successful ITS requires a great deal of time and expertise, which has inspired researchers to develop authoring tools to aid in their development (Ainsworth & Fleming, 2006; Blessing, 1997; Marchiori et al., 2012). Authoring tools have empowered instructors, researchers, and designers to create additional content and modify the ways in which a system responds to different performance and behaviors. One key goal of authoring tools is to facilitate these design objectives. Although authoring tools are developed for all stages of ITS development, we focus on tools (and the techniques that enable such tools) that are designed for researchers and instructors who, ultimately, are the ones who use the system. Ideally, for example, domain experts should be able to successfully modify a system for their particular domain even if they lack training as a computer programmer (Murray, 2003).

Notably, not all domains impose the same challenges to the creation and implementation of ITSs and their authoring tools. Developing a system to provide instruction on algebra is quite different from a system to teach aesthetic design. One common distinction made by developers of educational technologies is between *well-defined* and *ill-defined* learning problems and domains (Le, Loll & Pinkwart, 2013; Lynch, Ashley, Pinkwart & Aleven, 2009). Generally, problems in more well-defined domains have a limited number of solutions, and importantly, those solutions can be objectively predefined (e.g., $2 \times 2 = 4$). By contrast, problems in ill-defined domains often have multiple solutions and the accuracy or quality of those solutions can be subjective and on a continuous scale (e.g., the quality of an essay). As such, building an expert model or tracing students' progress through a series of problems is different for well-defined and ill-defined domains. In this chapter, we particularly focus on the needs of researchers and teachers in the ill-defined domains of reading comprehension and writing, and how those needs can begin to be addressed by authoring tools and data collection techniques.

The observations and recommendations in this chapter are based on two systems developed in our lab: the Interactive Strategy Training for Active Reading and Thinking-2 (iSTART-2) and Writing Pal (W-Pal). Through our discussion, we aim to extract key lessons we have garnered during our development process and use those lessons learned to provide suggestions for Generalized Intelligent Framework for Tutoring (GIFT) and other ITSs. Specifically, we first highlight the need for flexibility of content within our systems. Researchers require the ability to edit system features to test their effectiveness, and teachers need to add and edit content to better align with their courses. The system features that are made available to researchers and teachers should be selected to support these particular needs. Next, we highlight the potential benefits of collecting and analyzing behavioral data beyond what is required for traditional assessments. By conducting such analyses, researchers can learn about the processes underlying students' choices and performance. Importantly, these analyses are intended to eventually feed back into system flexibility, affording more appropriate and timely feedback for a broader range of content. Although our

recommendations are not limited to systems for ill-defined domains, they have emerged as particularly salient topics in our own work.

Related Research

In this section, we first provide a brief overview of each of our systems. We then discuss research related to scoring student responses using natural language processing (NLP) techniques and stealth assessments. Ultimately, these are the approaches we suggest here as having some potential to enhance flexibility and efficacy in tutoring systems, particularly in the context of authoring tools.

iSTART-2 and Writing Pal

The iSTART-2 system is a game-based tutoring system designed to improve high school students' reading comprehension by providing self-explanation and comprehension strategy instruction (Jackson & McNamara, 2013; McNamara, Levinstein & Boonthum, 2004; Snow, Allen, Jacovina & McNamara, 2015). Students using iSTART-2 complete a training phase before moving on to the practice phase. The training phase consists of a series of lesson videos that cover five self-explanation strategies and provide examples of their use. These lessons provide students with instruction on how to paraphrase texts in their own words, monitor their understanding of text information, predict what topics and information the text will next cover, bridge information with previous parts of the text, and elaborate on text information using prior knowledge. Each lesson video also includes a series of checkpoint questions that reinforce students' understanding of these strategies.

The practice phase includes a series of practice activities and customization options. From the practice menu, students can engage with several practice games, check their achievements earned during practice, or personalize the color of the system or the appearance of an on-screen avatar. The practice games fall into two categories: generative or identification practice. In generative practice, students read science texts and self-explain selected target sentences. They receive a score and (in certain activities) feedback on how to improve their self-explanations. In identification games, students read self-explanations that have ostensibly been written by other students, with the goal of identifying which of the five self-explanation strategies were used by the student. Our research indicates that when students receive self-explanation training in iSTART-2 (and earlier versions, iSTART and iSTART-ME), their self-explanation quality and comprehension improves when compared to receiving no self-explanation training (e.g., McNamara et al., 2004; McNamara, O'Reilly, Best & Ozuru, 2006; McNamara, O'Reilly, Rowe, Boonthum & Levinstein, 2007).

W-Pal is a game-based tutoring system designed to provide high school students with strategy lesson training, strategy practice, and holistic writing practice, specifically for prompt-based, argumentative essays (Allen, Crossley, Snow & McNamara, 2014; Roscoe & McNamara, 2013; Roscoe, Brandon, Snow & McNamara, 2013). The system includes eight modules that cover topics within prewriting (Freewriting and Planning), drafting (Introduction Building, Body Building, and Conclusion Building), and revising (Paraphrasing, Cohesion Building, and Revising). Each of these modules contains a series of lesson videos covering specific strategies that students are encouraged to use during the writing process. Examples of the strategies and checkpoint questions are included in these videos.

Students practice using strategies in a variety of practice games that focus students on individual components of writing (e.g., practicing conclusion paragraphs). Across games, students are given different tasks, such as generating text, answering multiple-choice questions, or organizing information by dragging and dropping. The game mechanics, such as points, levels, and bonus activities (e.g., a Sudoku-like game) are designed to enhance motivation and engagement. Students can also practice

writing essays and receive automatic formative feedback. Research from our lab indicates that students' writing strategy knowledge and writing proficiency improves over time while using the system (e.g., Allen, Crossley, et al., 2014; Crossley, Varner, Roscoe & McNamara, 2013).

Flexibility through Natural Language Processing

NLP lies at the core of both iSTART-2 and W-Pal. Within both, we have attempted to develop NLP algorithms that maintain a certain degree of flexibility within the systems, such that new content can be added to the systems (e.g., by the teachers) without having to recalculate the algorithms.

In iSTART-2, an NLP algorithm drives the self-explanation scoring using both latent semantic analysis (LSA; Landauer, McNamara, Dennis & Kintsch, 2007) and word-based measures to provide a score from 0 to 3. The algorithm is designed to assess the quality of the self-explanation in terms of how well students employed self-explanation strategies, not in terms of their content knowledge. That is, the comparisons made between the content of the text and students' self-explanations can detect similarities but not inaccuracies. One considerable advantage of not scoring quality of content knowledge (which is a very difficult task) is that any text can be entered into the system and used for practice. The algorithm assigns a low score when the self-explanation is short or contains irrelevant information and higher scores when the self-explanation incorporates information from earlier in the text and other relevant information. Scores from the iSTART algorithm have been shown to be similar to those of human raters (McNamara, Boonthum, Levinstein & Millis, 2007; Jackson, Guess & McNamara, 2010). Based on these scores and other factors (e.g., students' recent history of scores and the strategies they self-report using), students also receive feedback messages. When students consistently generate high quality self-explanations, they receive positive feedback. But when students receive lower scores, they might be encouraged to employ different self-explanation strategies, such as elaborating on what is in the text with what they already know. Instructors are able to add their own texts to the system that students can then self-explain using one of the system's generative practice activities. By adding their own texts, teachers can customize the content of iSTART-2 to more efficiently fit into their lesson plans. For example, a science teacher might input and assign several texts on photosynthesis; completing this training will then not only provide instruction on comprehension strategies for challenging science texts, but also help cover material within the teacher's curriculum.

NLP algorithms also drive the scoring and feedback in W-Pal. These algorithms are based on several linguistic properties of students' essays, ranging from simple measures such as the total number of words and paragraphs, to more sophisticated measures such as syntactic complexity and lexical specificity. Linguistic indices are calculated using both Coh-Metrix (McNamara & Graesser, 2012; McNamara, Graesser, McCarthy & Cai, 2014) and the Writing Analysis Tool (WAT; McNamara, Crossley & Roscoe, 2013). These algorithms provide students with both summative feedback on their essay (i.e., a holistic score on 6-point scale) as well as formative strategy feedback. The formative feedback provides students with actionable suggestions for how to improve the student's current and future essays. The feedback messages align with the strategy lesson videos provided within W-Pal. For example, if an essay contains very few words, feedback messages will likely focus on idea generation. Similar to iSTART-2, the algorithms in W-Pal are designed to be relatively generalizable; they are not tied to specific prompts. This allows teachers to add their own essay prompts into the system and create their own assignments for students.

Stealth Assessments

The ability for a system to provide *intelligent* feedback and recommendations to students is, in part, dependent on the quality of the student model and the data that drive that model. Performance measures

(e.g., accuracy) are clearly important, but sometimes not sufficient for a system to behave ideally—for example, they may not successfully detect when a student is sufficiently bored to consider quitting the system. Additional measures of student behavior and engagement may also be necessary. One way to covertly capture learning behaviors is through the use of stealth assessment (Shute, 2011; Shute, Ventura, Bauer & Zapata-Rivera, 2009). Stealth assessments are metrics designed to measure a specific variable that are discretely woven into a learning task rendering them *invisible* to the learner. This design allows these covert measures to assess designated constructs (e.g., engagement, cognitive skills, etc.) without disrupting students' flow during learning. Stealth assessments offer an alternative to traditional self-report or explicit construct measures. Indeed, one advantage of stealth assessments is that they do not rely on students' perceptions or memory of the learning task, but instead capture the targeted behavior in real time as it occurs during learning, thus eliminating the concern of a discrepancy between students' perceived behavior and observations of their actual behavior (McNamara, 2011). Stealth assessments can also save valuable time during an experiment or in a classroom. These measures do not have to be collected separately from the learning task and as such, and do not require extra instruction or time allocation that can take away from the teacher or ultimate learning task.

There are multiple ways that researchers can create and design stealth assessments (Shute, 2011). Relevant to this chapter is the use of online data (i.e., log data, language, and choice patterns) as proxies for learning behaviors. Online data have been used as a form of stealth assessment to measure a multitude of constructs, such as students' self-regulatory abilities (Hadwin et al., 2007), amount of exerted agency (Snow et al., 2015), and gaming behaviors (Baker, Corbett, Roll & Koedinger, 2008). For example, Hadwin and colleagues (2007) used log data from *gStudy* to examine variations and patterns in students' studying; *gStudy* is software that displays content to learners and tracks, for example, their annotating, searching, and help-seeking behaviors. The authors were particularly interested in examining how log data from *gStudy* could be used to profile students' self-regulatory abilities compared to traditional self-report measures. Results from this work revealed that students' studying habits could be captured by log data and patterns in these habits were predictive of self-regulation ability. Such promising results showcase the potential for stealth assessments to influence the behavior of an ITS. Critically, however, researchers must be careful (and often quite clever) in thinking about which interactions could relate to important student attributes or abilities, systems must be designed to record this information, and finally, the information needs to be usefully implemented into the system to make it more adaptive to individual students.

Discussion

Flexibility for Researchers and Teachers

Our ultimate goal is for iSTART-2 and W-Pal to be widely used by educators to enhance instruction of reading comprehension and writing. Specifically, one audience is high school teachers and their students. In order to optimize our systems, while simultaneously better understanding the processes involved with writing and reading, we also need our systems to be easily used by researchers who want to design experiments within our systems to answer research questions. Even research questions that are not directly designed to test the system will inevitably give some indication of system effectiveness and collect behavioral data from participants' use. For these reasons, we consider the usability of our system for researchers and teachers to be a high priority, thus requiring the development of authoring tools to meet their needs. Table 1 provides a summary of the features we discuss. We note that our estimates of the difficulty of use are based on anecdotal experiences rather than empirical data.

Table 1. Summary of the tools/features discussed in this chapter, including the targeted user for each.

Feature	System(s)	User(s)	Difficulty	Comments
Lesson/practice selection	iSTART-2 & W-Pal	Researchers*	Easy	Intuitive: Checkboxes mark available activities
Practice activity appearance groups	W-Pal	Researchers	Moderate	Requires an understanding of how different completion conditions (e.g., completing a game) will trigger the next appearance group
Performance thresholds	iSTART-2	Researchers	Moderate	Requires an understanding of iSTART-2 scoring
Essay feedback quantity/control	W-Pal	Researchers	Easy	Intuitive: Uses radio buttons
Essay self-assessments	W-Pal	Researchers	Easy	Intuitive: Uses radio buttons
New essay prompts	W-Pal	Teachers & Researchers	Easy	Intuitive: However, prompts must be for persuasive essays
New practice texts	iSTART-2	Teachers & Researchers	Advanced	Requires knowledge of how to tag appropriate target sentences

* Currently being considered for teachers

Note: Difficulty corresponds to the time required to use the feature competently, not masterfully. We estimate that “easy” features are immediately usable provided the user has a basic understanding of the system; “moderate” features require 1–2 hours to learn; “advanced” features require specialized knowledge through training/tutorials (~3–5 hours).

Before attempting to design a complete set of authoring tools for researchers and teachers, we built systems that delivered lesson content that covered targeted strategies and practice activities that provided actionable feedback—that is, more or less complete (essentially hard coded) systems that functioned on their own. As we tested the success of these systems, building certain tools for our research team was a practicality; we needed to toggle features on and off to test their relative effectiveness. Moreover, we wanted researchers without a programming background to be able to set these options for different students within the system. To meet this need, our programming team developed a web interface through which researchers can set the parameters for several options. As these selectable features accumulated in our researcher control panel, the systems became more flexible. Because student accounts are enrolled in “system classrooms,” each of which has its own settings, researchers can make comparisons between students in different classrooms.

When possible, settings and features are selectable through a live connection between the authoring tool and students’ interface. For example, researchers can select which lesson videos and practice activities are displayed to students. Figure 1 shows the iSTART-2 researcher control panel being used to disable certain practice games from appearing in students’ practice interface. Importantly, the layout of the authoring tool for this page matches the layout of the practice interface, with checkboxes indicating which games will be

available to students. In W-Pal, a more powerful (though more complex) tool is available that allows researchers to both select practice activities in each module as well as the order in which the activities are available to students. Figure 2 shows the tool that researchers use to define practice game “appearance groups” (i.e., one or more games that appear to students as part of a single group) as well as the conditions that must be met to advance to the next appearance group. In the depicted example from W-Pal’s Body Building module, a researcher has created two groups, each with one game. To advance from the first group to the second, a student must complete the game *Fix It: Bodies* three times. Researchers can also define a time requirement for how long students must play a game before advancing (leaving the time at zero yields no time requirements). Several other settings are available through this tool, such as the ability to control how students are transitioned from one group to the next once time has expired, or display pop-up messages to students after completing the appearance group. The appearance group editing tool is thus useful for controlling students’ practice experience and assessing the relative value of different games. After completing an appearance group that has been set, researchers see a visual representation for each group, which is similar to what will be displayed to students. The close alignment between what is seen by researchers and students renders it easy for researchers to confirm that settings are correct.

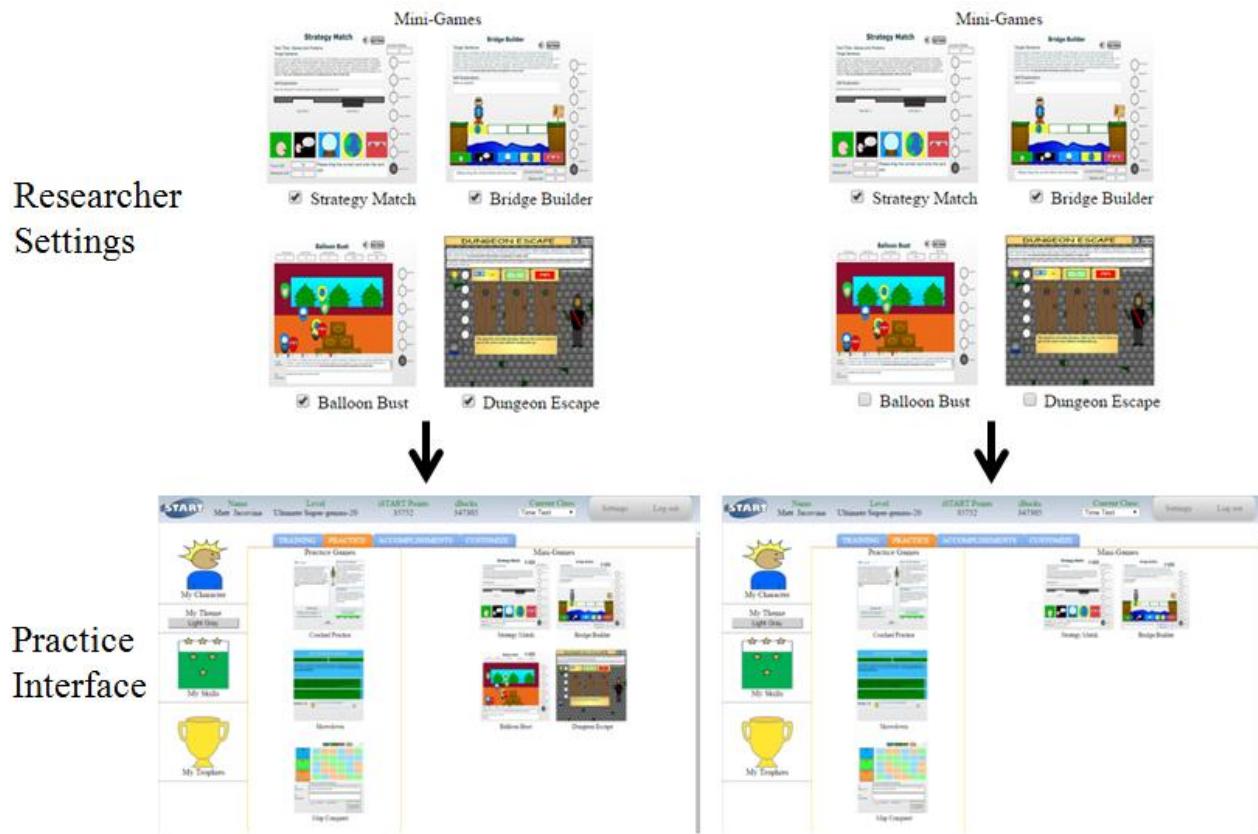


Figure 1. Research settings in iSTART-2 and the resulting practice interfaces that are visible to students.

Number of Groups: 2

Appearance Group 1 Games

- Fix It: Bodies
- RoBoCo
- Body Paragraph Editor
- Topic and Evidence Writer

Completion Conditions

Play time(minutes): 0

Games completed: 3

Completion Message

Transition to Next Appearance Group

- User closes game/lesson window
- Automatically close game/lesson window
- Automatically close game/lesson window after 60 seconds

Start Timer...

- After user clicks the game icon
- After user clicks the in-game Start Button

Single game options

- Require continuous play until completion of appearance group
- Game survey only at completion of appearance group

Appearance Group 2 Games

- Fix It: Bodies
- RoBoCo
- Body Paragraph Editor
- Topic and Evidence Writer

Completion Conditions

Play time(minutes): 0

Games completed: 1

Completion Message

Transition to Next Appearance Group

- User closes game/lesson window
- Automatically close game/lesson window
- Automatically close game/lesson window after 60 seconds

Start Timer...

- After user clicks the game icon
- After user clicks the in-game Start Button

Single game options

- Require continuous play until completion of appearance group
- Game survey only at completion of appearance group



Fix It: Bodies

Group 1 Play Time(minutes): 0
Group 1 Games Completed: 3



RoBoCo

Group 2 Play Time(minutes): 0
Group 2 Games Completed: 1

Figure 2. The appearance group editor in W-Pal and the resulting visual representation of the groups.

As examples of more specialized features, researchers in iSTART-2 can set pop-up messages to trigger when students do not meet a performance threshold in generative practice games, after which they are transitioned to a more rigorous practice activity. In Figure 3, a researcher has set the threshold to a score of 2.0 and applied that threshold to the games *Map Conquest* and *Showdown*. With this setting, whenever students' average self-explanation quality score is below 2.0 across those games, they receive a pop-up message that is defined in the editor. After closing the pop-up message, students are transitioned to *Coached Practice*, an activity that has fewer game features but provides additional feedback to students. By using this tool, researchers can assess the effects of alerting students of their poor performance and prescribing a specific practice activity as a means for improvement—additionally, the specific wording of the message and the stringency of the threshold can be easily manipulated. In W-Pal, researchers can also set options that change students' experiences while practicing; notably, during the process of writing essays and receiving feedback. These changes are made through the researcher control panel simply by toggling features on and off, or by selecting among options using radio buttons. For example, researchers can set whether students self-assess the quality of their essay after submitting it but before receiving feedback. Researchers also select whether students have control over the number of feedback messages that they receive about their essay, and the maximum number of messages that they can receive. By varying these features, researchers can study the optimal conditions for encouraging quality essay revisions following the delivery of the automated essay feedback.

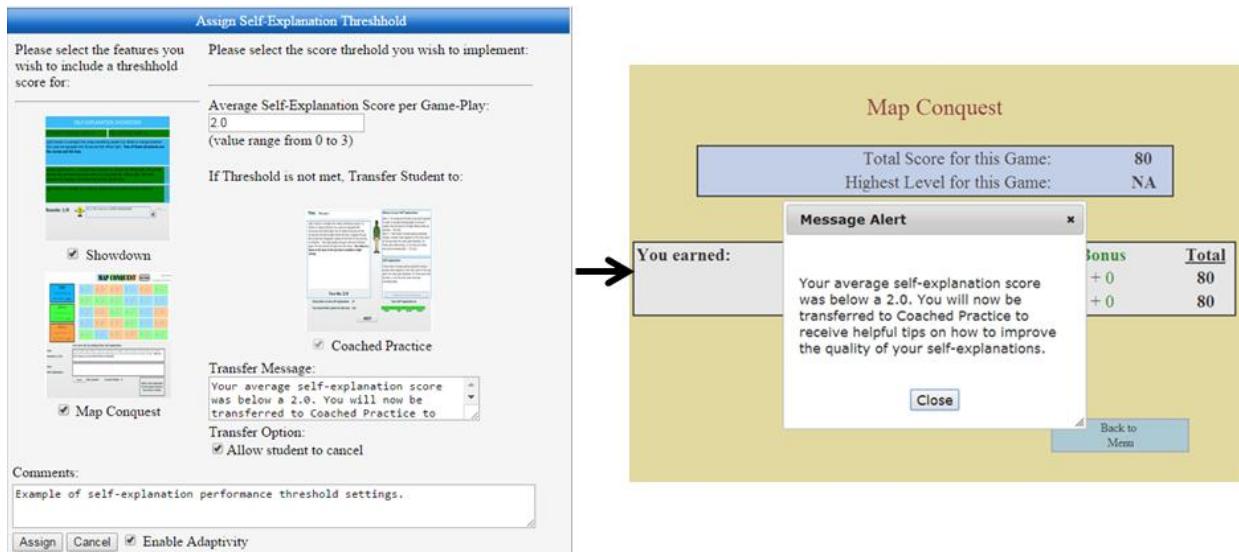


Figure 3. The self-explanation threshold editor and the pop-up message students receive after not meeting the performance threshold.

Generally, we consider the numerous options available to be a boon for researchers. We view our authoring tools as communicative of the features that are potentially important for students. By design, a researcher who is interested in reading comprehension or writing should be able to set up system classrooms with different settings and design an experiment using our researcher control panel, with minimal experience with the system and no programming knowledge. Selecting *which* system features to test and in which combination to design an interesting study, of course, requires expertise. However, some features are obviously more important than others, and we rely on researchers to make careful study design decisions whenever changing a feature from its default setting.

For teachers, the communicative function of our authoring tools is somewhat different. Options available through the teacher control panel may be considered as a means to customize the system to best match course content and classroom needs. When considering many of the features available to researchers, the goal of optimally setting system options is ambiguous. Disabling games might seem like a sensible decision if the teacher is under a tight time schedule and if the teacher fears that students will ignore the goal of learning to write in the context of games. However, our research indicates that eliminating the motivating features of games will likely decrease performance over the long term (e.g., Jackson & McNamara, 2013). Therefore, we currently do not include the ability to toggle games on and off within the teacher control panel. This may change in the future, of course, and teachers always retain their control over what they do and do not assign to students. Some features, meanwhile, are esoteric and clearly should be excluded from teachers' options (e.g., being able to disable certain uses of the word "game," which was included for a study in which we did not want to prime students to think of practice as gaming). Thus, for teachers, we have aimed to build authoring tools around their needs for adapting the system to their course. This has primarily centered on content creation.

A recent survey found that a majority of teachers prefer to modify the content of educational resources they obtain, and that they often share the resources they find with colleagues (Hassler, Hennessy, Knight & Connolly, 2014). Our experiences working with teachers match these findings, and we propose that flexibility of content is particularly important for ill-defined domains in which skills are often taught in the context of topics particular to individual classrooms. For example, the persuasive essay writing skills covered in W-Pal might normally be taught in the context of current events or topics raised by a book the

class is currently reading. Teachers may be unlikely to use systems in ill-defined domains that do not allow them to align practice (i.e., students' system use) with system content. Because of the NLP techniques we use to drive our scoring algorithms, however, our systems are able to meet this considerable challenge (see the previous section on NLP for more information). Figure 4 shows the interface teachers use to add new argumentative writing prompts into W-Pal. Though plain, this simple interface allows teachers to create new assignments in W-Pal that receive the same level and quality of feedback as the prompts built into it. Thus, pasting in an essay prompt and assigning it takes minutes and provides students, by default, a 25+ minute practice experience (longer if revisions are required). Similarly, iSTART-2 allows teachers to add texts to the system that can be self-explained in the practice activities. Although this process is somewhat complicated by the need to define target sentences (currently, we work directly with teachers wishing to add texts but will add tutorials in the future), it allows teachers, without an understanding of the algorithms driving feedback, to expand and customize system content. The NLP underpinnings in both systems are invisible to teachers, allowing them to focus on adding essay prompts and texts through the simple features available in the teacher control panel. Learning to adeptly tag target sentences takes many hours, but once mastered, teachers will be able to add texts in about 30 minutes, completing both the entry and tagging processes; practice with each text will last 10–30 minutes depending on its length.

Figure 4. Interface for adding a new essay prompt in W-Pal.

Stealth Assessments and their Representation in Authoring Tools

An ongoing goal for our systems is to better direct instruction and feedback to each student. Although our systems currently deliver feedback messages and make recommendations based on students' current and past performance, we plan to build richer student models that can respond with greater nuance. For ill-defined domains, in particular, constructing these models is a challenge that must be supported by copious data. As we discussed earlier, we are strong proponents of using stealth assessments to help obtain much information about students in a non-intrusive manner. In our own system designs, we allow students to make important choices that afford meaningful interaction patterns and generate responses that can be analyzed using NLP techniques. Essentially, our goal is to build systems that convey rich information about students through their normal interactions that go beyond what is directly being measured. This promotes meaningful data mining of system interactions. For example, in one study, we analyzed the

degree to which students were ordered or disordered in their interactions with iSTART-2, and found that more ordered interactions led to better performance (Snow et al., 2015). In another study, we found that when analyzing the narrativity of students' writing over a series of several essays, more successful writers were less rigid in their use of narrative elements (Allen, Snow & McNamara, under revision). In both studies, we were able to use stealth assessment to measure important student characteristics. In the future, we will attempt to leverage our ability to monitor these student attributes by using them to drive feedback messages.

The importance of stealth assessment, however, is not primarily what we wish to push forward (the virtues of stealth assessment have already been beautifully laid out in a past volume: Ventura, Shute & Small, 2014). Instead, we suggest that stealth assessments should become more prominent and easier for researchers (and eventually teachers) to use. The goals are to build better understandings of what stealth measures are capturing and, subsequently, drive better instruction for students. Eventually, we plan for our authoring tools to provide examples of the types of measures that are logged by the system and encourage researchers to consider those measures in conjunction with the other components of their studies. Over time, we intend this enhanced awareness of stealth measures to improve the understanding of tutoring systems for reading comprehension and writing. This goal could be particularly important for all ill-defined domains that already struggle for tractability in scoring and modeling. If a research group, for example, conducts a study examining impulsivity and writing performance, they could easily compare impulsivity scores with choice pattern measures, which our systems already measure—this could provide insight into impulsivity, choice patterns, and their interaction with writing performance. The authoring tools that researchers use when setting up their studies should make it apparent that these analyses are possible. When researchers are ready to run the analyses, tools should then provide these data to researchers in an understandable format. Again, we view it as an important goal of authoring tools to communicate system features that are pertinent to the needs of researchers.

Teachers, likewise, could benefit from an understanding of some of the stealth assessments that a system records. Although the system will ideally be using relevant information to guide instruction, keeping teachers apprised of their students' system performance can be helpful for letting the teacher know what is and is not working, and, of course, teachers can often intervene in ways that the system cannot (e.g., a teacher might assign different work to a student who is struggling with system content). By displaying certain stealth measures to teachers within their authoring interface, they will also develop a better understanding of how the system works. Although teachers do not need to understand the intricacies of how a system's "intelligence" works, it might also inspire observations about their students' in-person behavior as it connects to their system performance. Perhaps an oft-distracted student is particularly motivated by the game components of reading practice, and a teacher can leverage this information in other ways. Finally, by empowering teachers with knowledge of how a system works, they are better able to communicate their feedback and work with designers to improve its ability to function within classrooms. An obvious issue with displaying this information is that it may be counterproductive; instead of being enlightening, it may be overwhelming, confusing, and unhelpful. For our own systems, we have been cautious in adding too much information and have discussed with several teachers the pros and cons of adding specific pieces of information. Our approach to communicating with teachers about these issues varies by situation. Some teachers have a strong interest in educational technology and frequently provide feedback about their desired features and are excited to provide insights about the utility of more advanced features. In other situations, we ask teachers to fill out short online surveys that include free response questions asking about how we can improve and what they would like to see added. Based on information from teachers, we are planning new features, some of which will convey students' choice patterns.

Recommendations and Future Research

In this chapter, we discussed how stealth assessment techniques undergird our tutoring systems, iSTART-2 and W-Pal, which operate in the ill-defined domains of comprehension and writing. We specifically explore how techniques, such as NLP, can be used within the context of authoring tools and ill-defined domains in which student-generated responses must be scored and for which teachers (or researchers) may want to add their own content and prompts. Stealth assessments afford researchers the opportunity to examine and build more nuanced, complete models of student performance and behavior. Thus, for researchers and teachers, these techniques can help inform authoring tools, acting both as communicative devices to explain the impact of various features on learning and as means for content to be edited and added.

GIFT offers a platform to build powerful tutoring systems that can adapt to student needs. Its greatest strengths are currently most likely to be used by cognitive scientists and programmers who are already skilled developers. In order for the efficient advancement and proliferation of ITSs in ill-defined domains, however, we suggest that researchers and teachers must collaborate in system design, particularly to test and optimize system features. Across the brief time of our using GIFT, it has already made great strides in becoming easier for non-programmers to author; the example courses available through the GIFT package can easily be used as models and modified. The exemplified ability to use PowerPoint to present content—a familiar tool for many researchers and teachers—is an excellent means of affording educators opportunities to expand course content.

One avenue for expanding GIFT would be the addition of features that allow students to generate written responses and then receive feedback. Students often experience memory benefits when generating content, making generative activities educationally desirable (e.g., McNamara & Healy, 1995; Slamecka & Graf, 1978). To support these features, GIFT might consider incorporating simple NLP techniques (see Crossley, Allen & McNamara, 2014). NLP algorithms that rely on simple indices such as word counts and bags of words can go a long way in providing information about a student's responses. Such techniques can be effective for many purposes such as scoring responses to short answers, open-ended questions, and even, essays. As the framework evolves to more easily provide NLP output and use it to guide scoring and feedback, more sophisticated techniques can be developed and implemented (Allen, Snow, Crossley, Jackson & McNamara, 2014). An important goal for more advanced, flexible scoring and feedback algorithms will be to allow teachers to add their own question content.

Another consideration would be to provide easily understood methods of recording log data during system use. As we have discussed, the use of stealth assessments during tutoring affords the means to better understand students' use of the system and also collect information about the student without interruptions from surveys or additional assessments. Adding the ability to then implement these data—as well as linguistic data extracted from student-generated responses—into student models delivers a powerful tool for researchers. For teachers, displaying the most important and interpretable of these measures could also be useful to communicate nuances of student performance that might remain hidden when only traditional performance summaries are provided. Ultimately, the information provided by stealth assessments such as NLP techniques, can improve systems' ability to identify when students need assistance and what specific assistance would be most appropriate.

One exciting aspect about the GIFT project is its potential to empower both research and educational communities with the ability to build powerful ITSs. Because of the flexible and adaptable nature of the framework, a wide range of features can be built into systems that cover content in many domains. A particular hope is that these systems spread, inspiring researchers to test components of various systems and offering educators the opportunity to provide valuable feedback. Through such a network, combined

with the power of stealth assessment techniques such as NLP, even the challenges of ill-defined domains can be met successfully.

References

- Ainsworth, S. & Fleming, P. (2006). Evaluating authoring tools for teachers as instructional designers. *Computers in Human Behavior*, 22, 131-148.
- Allen, L. K., Crossley, S. A., Snow, E. L. & McNamara, D. S. (2014). Game-based writing strategy tutoring for second language learners: Game enjoyment as a key to engagement. *Language Learning and Technology*, 18, 124-150.
- Allen, L. K., Snow, E. L., Crossley, S. A., Jackson, G. T. & McNamara, D. S. (2014). Reading components and their relation to the writing process. *Topics in Cognitive Psychology*, 114, 663-691.
- Allen, L. K., Snow, E. L., and McNamara, D. S. (under revision). The narrative waltz: The role of flexible style on writing performance. Manuscript submitted to the *Journal of Educational Psychology*.
- Baker, R. S. J. D., Corbett, A. T., Roll, I. & Koedinger, K. R. (2008). Developing a generalizable detector of when students game the system. *User Modeling and User-Adapted Interaction*, 18, 287-314.
- Beal, C., Arroyo, I., Cohen, P. & Woolf, B. (2010). Evaluation of AnimalWatch: In intelligent tutoring system for arithmetic and fractions. *Journal of Interactive Online Learning*, 9, 64 -77.
- Blessing, S. B. (1997). A programming by demonstration authoring tool for model-tracing tutors. *International Journal of Artificial Intelligence in Education*, 8, 233-261.
- Crossley, S. A., Allen, L. K., Kyle, K. & McNamara, D. S. (2014). Analyzing discourse processing using a simple natural language processing tool. *Discourse Processes*, 51, 511-534.
- Crossley, S. A., Varner (Allen), L. K., Roscoe, R. D. & McNamara, D. S. (2013). Using automated cohesion indices as a measure of writing growth in intelligent tutoring systems and automated essay writing systems. In H. C. Lane, K. Yacef, J. Mostow & P. Pavlik (Eds.), *Proceedings of the 16th International Conference on Artificial Intelligence in Education (AIED)* (pp. 269-278). Heidelberg, Berlin: Springer
- Graesser, A., Lu, S., Jackson, G., Mitchell, H., Ventura, M., Olney, A. & Louwerse, M. (2004). AutoTutor: A tutor with dialogue in natural language. *Behavior Research Methods, Instruments & Computers*, 36, 180 –192.
- Hadwin, A. F., Nesbit, J. C., Jamieson-Noel, D., Code, J. & Winne, P. H. (2007). Examining trace data to explore self-regulated learning. *Metacognition and Learning*, 2, 107-124.
- Hassler, B., Hennessy, S., Knight, S. & Connolly, T. (2014). Developing an open resource bank for interactive teaching of STEM: Perspectives of school teachers and teacher educators. *Journal of Interactive Media in Education*.
- Jackson, G. T., Guess, R. H. & McNamara, D. S. (2010). Assessing cognitively complex strategy use in an untrained domain. *Topics in Cognitive Science*, 2, 127-137.
- Jackson, G. T. & McNamara, D. S. (2013). Motivation and performance in a game-based intelligent tutoring system. *Journal of Educational Psychology*, 105, 1036-1049.
- Landauer, T. K., McNamara, D. S., Dennis, S. & Kintsch, W. (Eds.). (2007). *Handbook of Latent Semantic Analysis*. Mahwah, NJ: Lawrence Erlbaum.
- Le, N. T., Loll, F. & Pinkwart, N. (2013). Operationalizing the continuum between well-defined and ill-defined problems for educational technology. *IEEE Transactions on Learning Technologies*, 6, 258-270.
- Lynch, C., Ashley, K. D., Pinkwart, N. & Aleven, V. (2009). Concepts, structures, and goals: Redefining ill-definedness. *International Journal of Artificial Intelligence in Education*, 19, 253-266.
- Marchiori, E. J., Torrente, J., del Blanco, Á., Moreno-Ger, P., Sancho, P. & Fernández-Manjón, B. (2012). A narrative metaphor to facilitate educational game authoring. *Computers & Education*, 58, 590-599.
- McNamara, D. S. (2011). Measuring deep, reflective comprehension and learning strategies: Challenges and successes. *Metacognition and Learning*, 3, 1-11
- McNamara, D. S., Boonthum, C., Levinstein, I. B. & Millis, K. (2007). Evaluating self-explanations in iSTART: Comparing word-based and LSA algorithms. In T. Landauer, D. S. McNamara, S. Dennis & W. Kintsch (Eds.), *Handbook of latent semantic analysis* (pp. 227–241). Mahwah, NJ: Erlbaum.
- McNamara, D. S., Crossley, S. A. & Roscoe, R. D. (2013). Natural language processing in an intelligent writing strategy tutoring system. *Behavior Research Methods*, 45, 499-515.

- McNamara, D. S. & Graesser, A. C. (2012). Coh-Metrix: An automated tool for theoretical and applied natural language processing. In P. M. McCarthy & C. Boonthum (Eds.), *Applied natural language processing and content analysis: Identification, investigation, and resolution* (pp. 188-205). Hershey, PA: IGI Global.
- McNamara, D. S., Graesser, A. C., McCarthy, P. & Cai, Z. (2014). Automated evaluation of text and discourse with Coh-Metrix. Cambridge: Cambridge University Press.
- McNamara, D. S. & Healy, A. F. (1995). A generation advantage for multiplication skill and nonword vocabulary acquisition. In A. F. Healy & L. E. Bourne, Jr. (Eds.), *Learning and memory of knowledge and skills* (pp. 132-169). Thousand Oaks, CA: Sage.
- McNamara, D. S., Levinstein, I. B. & Boonthum, C. (2004). iSTART: Interactive strategy trainer for active reading and thinking. *Behavioral Research Methods, Instruments & Computers*, 36, 222-233.
- McNamara, D. S., O'Reilly, T., Best, R. & Ozuru, Y. (2006). Improving adolescent students' reading comprehension with iSTART. *Journal of Educational Computing Research*, 34, 147-171.
- McNamara, D. S., O'Reilly, T., Rowe, M., Boonthum, C. & Levinstein, I. B. (2007). iSTART: A web-based tutor that teaches self-explanation and metacognitive reading strategies. In D. S. McNamara (Ed.), *Reading comprehension strategies: Theories, interventions, and technologies* (pp. 397-421). Mahwah, NJ: Erlbaum.
- Murray, T. (2003). An overview of intelligent tutoring system authoring tools: Updated analysis of the state of the art. In T. Murray, S. Blessing & S. Ainsworth (Eds.), *Authoring tools for advanced technology learning environments* (pp. 491-544). Dordrecht, Netherlands: Kluwer Academic Publishers.
- Roscoe R. D., Brandon, R. D., Snow, E. L. & McNamara, D. S. (2013). Game-based writing strategy practice with the Writing Pal. In K. Pytash & R. Ferdig (Eds.), *Exploring technology for writing and writing instruction*. (pp. 1-20). Hershey, PA: IGI Global.
- Roscoe, R. D. & McNamara, D. S. (2013). Writing pal: Feasibility of an intelligent writing strategy tutor in the high school classroom. *Journal of Educational Psychology*, 105, 1010-1025.
- Shute, V. J. (2011). Stealth assessment in computer-based games to support learning. *Computer games and instruction*, 55, 503-524.
- Shute, V. J., Ventura, M., Bauer, M. & Zapata-Rivera, D. (2009). Melding the power of serious games and embedded assessment to monitor and foster learning. In U. Ritterfield, M. Cody & P. Vorderer (Eds.), *Serious games: Mechanisms and effects* (pp. 295-321). New York, NY: Routledge.
- Slamecka, N. J. & Graf, P. (1978). The generation effect: Delineation of a phenomenon. *Journal of Experimental Psychology: Human Learning and Memory*, 4, 592-604.
- Snow, E. L., Allen, L. K., Jacovina, M. E. & McNamara, D. S. (2015). Does agency matter?: Exploring the impact of controlled behaviors within a game-based environment. *Computers & Education*, 26, 378-392.
- Ventura, M., Shute, V. & Small, M. (2014). Assessing persistence in educational games. In R. Sottilare, A. Graesser, X. Hu, and B. Goldberg (Eds.), *Design recommendations for intelligent tutoring systems: Volume 2 Instructional management* (pp. 93-101). Orlando, FL: U.S. Army Research Laboratory.

CHAPTER 9 Design Considerations for Collaborative Authoring in Intelligent Tutoring Systems

Charlie Ragusa

Dignitas Technologies, LLC

Introduction

Use of eLearning systems has grown dramatically in recent years, driven by demand from government, educational institutions, and corporations. Technological advancements have facilitated this growth, including software as a service (SaaS), cloud computing, and an increasing variety of delivery platforms (e.g., mobiles, tablets, internet-of-things). As Internet access and mobile device usage increase, the next generation is accustomed to the concept of interactive media for everything from informal information gathering to formal training.

In comparison to the broader eLearning community, intelligent tutoring systems (ITSs) are still primarily limited to a research and development context. A key enabler to the widespread adoption of ITSs will be the existence of robust and easy-to-use authoring tools (Murray, 2003). ITS development has special challenges compared to a general eLearning system, and development of domain independent ITSs even more so. Though certainly not trivial, the basics of authoring in many non-ITS eLearning systems are straightforward, typically involving support for authoring of non-interactive content (e.g., text, pictures, videos) and simple assessments (e.g., multiple choice). Learner assessment often takes the form of quizzes or exams, while content is frequently a link to existing media or an attached document. All too often this results in little more than a migration of offline content such as text books and lecture notes to an online environment, with the presentation of data enhanced through limited multimedia.

Authoring for an ITS is more demanding because the system is interactive: the difference is analogous to creating a playable video game instead of a movie. Content and knowledge assessment remain essential, but ITS-enabled courses require representations of domain knowledge, learner models, expert models, pedagogical models, conditional and non-linear flow through the material, and various meta-data. For ITSs equipped with physiological sensors, authoring is needed to adapt to the learner's affective state.

Due to these complexities, for non-trivial domains, the knowledge and skills required to author effective instruction often do not reside in a single individual. The best outcome is achieved by collaboration among some combination of instructional designers, subject matter experts, psychologists, traditional educators, and software engineers (Nye, Rahman, Yang, Hays, Cai, Graesser & Hu, 2014). This chapter examines the challenges related to collaborative authoring in general and as they pertain to the Generalized Intelligent Framework for Tutoring (GIFT; Sotilare, Brawner, Goldberg & Holden, 2012). Topics include roles and responsibilities, workflow, and software architecture considerations.

As an intelligent tutoring framework, GIFT is unique in that it is open source and domain independent, includes a sensor framework, and is designed to integrate with external training applications. These characteristics, along with the author's familiarity with GIFT, make it well suited for a discussion on collaborative ITS authoring. Consequently, discussion from this point forward is very GIFT centric. Of course, many of the ideas should be applicable to collaborative authoring in general.

Related Research

While the literature is replete with publications on eLearning and ITSs, relatively little has been published on the topic of *collaborative* authoring for ITSs. Early research on collaborative authoring typically addressed collaborative authoring of documents. More recently, collaboratively writing documents is pervasive and most readers should have some experience with collaborative authoring in a variety of formats such as the following:

- Documents shared via email
- Shared network drives within an organization
- Shared documents on cloud-based drives such as Microsoft OneDrive and Dropbox
- Wiki page authoring, e.g., Wikipedia
- Document workflow tools, such as Microsoft SharePoint
- Google Documents
- Microsoft OneNote and Word
- Content Management Systems
- WebDAV (Whitehead Jr. & Wiggins, 1998)
- Version control systems such as Subversion, Git, or Mercurial

Research on collaborative writing continues; however, only some of this work is relevant for eLearning. The eLearning industry has published *The Ultimate List of Cloud-Based Authoring Tools*, which lists over 50 cloud-based eLearning authoring tools (Pappas, 2013). Several tools offer support for collaborative authoring and some even support branching and interactivity, implying a rudimentary level of intelligent tutoring. There are a few published reviews of these tools, in some cases comparing many tools (Elkins, 2013), and in other cases, providing more in depth comparison of just two (Tao, 2015). This set of tools offers some insights for collaborative authoring. First, each tool tends to focus on either web developers or instructors, and less commonly both. Second, most tools allow authors to create content in ways that is familiar to them (e.g., translating their PowerPoint slides into an interactive web page). Finally, most tools focus in building specific learning resources that can be embedded as HTML pages.

Another relevant collaborative authoring environment is Stanford University's WebProtégé, a free open-source collaborative ontology development environment for the web (Tudorache, Nyulas, & Noy, 2013). WebProtégé is particularly interesting because, in addition to being a cloud-based collaborative authoring environment, it embodies many of the concepts described in this chapter including history and revision management, built-in discussion support, and interoperability with a desktop version of the Protégé authoring tool. Much like GIFT, it is a highly technical editor that outputs extensible markup language (XML) (among other formats). Also, WebProtégé is constructed using the Google Web Toolkit (GWT)

the same platform used to construct GIFT's web based authoring tools. Assuming the continued use of GWT by the GIFT team, approaches and techniques used by WebProtégé may be directly transferrable to future GIFT collaborative authoring tools.

Discussion

The current suite of GIFT authoring tools is largely desktop applications (Hoffman & Ragusa, 2014). The tools allow flexible configuration of the system, but are aimed toward software developers rather than content experts. Moreover, they were not specifically designed with collaboration in mind. However, each incremental improvement to the GIFT framework can update these tools, since they are generated automatically from the XML schemas of GIFT’s configuration files. Additional coding is necessary only when it is required to implement specialized functions, such as creation of custom dialogues or additional validation beyond the schema. While these tools do not formally support collaboration, the usual cumbersome collaboration methods for collaboration are possible: emailing authored files, shared drives, or a revision control system (e.g., Subversion). The latter approach has the advantage of versioning, graceful merging of edits, and conflict resolution for when two authors edit the same part of a file.

Though most GIFT tools are desktop applications, a few are web-based. The GIFT survey authoring system was designed as a web application and recent GIFT releases have introduced web-based tools for authoring courses and for domain knowledge files. These new tools are a step toward reaching content experts, but would be more powerful with explicit support for collaboration.

General Considerations

Independent of issues related to collaboration, any new authoring tools should adhere to best practices for user interface design (Stone, Jarrett, Woodroffe & Minocha, 2005), such as the following:

- Intuitive interfaces that do not surprise users with unusual behavior
- Availability of context sensitive help
- Aesthetics
- Input validation
- User-friendly error messages
- Undo/Redo
- Preview capability

These considerations are not discussed in detail, but are noted here for completeness.

Terminology and Authoring Granularity

Currently, GIFT supports authoring and runtime execution at the granularity of a single learner session which it calls a GIFT “course.” There is no minimum or maximum time associated with a course, but the working assumption is that a course will be completed in a single learner session, whether it be 5 minutes or 2 hours or more. Given a single granularity, this is the obvious choice, however, independent of collaborative authoring concerns, GIFT should expand its capability to support a wider range of granularities and would be well served by modifying its nomenclature to match current norms. One suggestion would be to rename the current course construct to “lesson” and repurpose the term “course” to describe a series of related lessons.

A further refinement would be to add an optional intermediate level of granularity that could be used to define “sections” or “modules” within a course. The precise terminology is perhaps less important than the support for the hierarchical construct. Despite this suggestion, unless otherwise noted, “course” will be used throughout the remainder of this chapter to reflect a GIFT course as currently implemented by GIFT. Collaborative authoring considerations for course/module/lesson hierarchies is left to a future discussion.

Authoring in the Cloud

GIFT supports both web-based content delivery as well as desktop/fat-client operation. Regardless of the runtime environment, GIFT authoring can and should be managed as a cloud-based web application. Cloud deployment is an ideal environment for collaborative authoring (Schneider, 2012). Beyond the obvious benefits to collaborative authoring of concurrent access by multiple users, cloud infrastructure typically includes support for several key elements of a collaborative system such as accessibility, storage, versioning, and scalability. For simple courses that require no other client resources beyond a web browser, content can remain in the cloud and be fetched by the browser as needed. On a desktop runtime environment, the course and any resources needed locally can be downloaded and cached as necessary. For the remainder of this chapter, a cloud-based authoring system is assumed.

Given the assumption of a cloud-based authoring environment, GIFT must move all core authoring functions to the cloud. Essential functions of the authoring system (ignoring collaboration, for the moment) include the following:

- Authoring, uploading, and management of content
- Authoring, uploading, and management of GIFT configuration elements/files
- Authoring and management of surveys¹
- Publishing authored courses (i.e., making them available for use)

The objective is for authored courses and all required resources to be served from the cloud, and fetched or downloaded as needed. Courses requiring only a browser and internet connection can be delivered on demand to the browser from the cloud. Courses using sensors or third-party desktop applications will be downloaded and cached by the local GIFT runtime, where the user or local administrator will bear some responsibility for downloading and installing the necessary desktop applications.

It should be noted that some changes suggested here will require changes to the GIFT runtime environment. As much as possible and practical, existing third-party software (e.g., Java Web Start) that can be used without license fees or proprietary encumbrances should be leveraged to handle the low-level details, including security related issues. From the author’s standpoint, the goal is a seamless and straightforward system. The same cloud application responsible for authoring could then be leveraged for tools such as report generation.

Resource Management, Projects, and Tool Integration

A typical GIFT course references multiple resources including some combination of the following:

¹ GIFT uses “survey” as a catch-all term for form-based quizzes, assessments, exams, as well as traditional surveys (e.g., psychological, biographical, and satisfaction, etc.).

- Content (HTML, PDF, PowerPoint, etc.)
- Core GIFT XML configuration files: Course, Domain Knowledge, Meta-Data
- Surveys¹
- 3rd Party Training Applications including application-specific scenario and configuration files, such as 3D training simulation data.
- Secondary XML configuration files: Learner and Sensor Configuration

Content and XML configurations currently exist as files. Surveys are managed using a relational database. To date, third-party training applications have been desktop applications installed on the user workstation that are not directly managed by GIFT. Existing GIFT best practices are to organize content and primary XML configuration files inside a common subfolder of a designated domain folder for the GIFT installation. A domain knowledge folder is required, but organization beyond that is not enforced. Rather than being configured on a per-course basis, secondary XML configuration files have been managed as part of the GIFT installation.

To facilitate collaboration, GIFT will need to create a *project* construct to serve as the overarching logical container for all the resources related to a specific effort. The project is analogous to the best-practice idea of locating related resources in a common folder, but is more flexible. Resources used by multiple projects can be stored in a single location and simply referenced by projects as needed. The project construct also serves to manage the collaboration settings for the project, including the user names of the collaborators, their roles, and access control specifications. This paradigm has parallels to collaborative editing tools of compiled documents (such as LaTeX, e.g., www.overleaf.com) or code projects (e.g., Cloud9, c9.io).

GIFT currently uses a distinct editor for each major authoring task. This is true of both the desktop authoring tools as well as the browser-based authoring tools. The project construct also serves to unify the tools so that the user experiences the tool suite as a single unified tool with multiple integrated functions. With the project construct as a framework, two collaboration functions are essential:

- Project creation
- Collaborator management

Project creation means the creation of a new project within the system. Collaborator management is the infrastructure used to manage collaborators and their roles, permissions, and workflow.

Types of Collaboration

Collaboration can take multiple forms. The most basic forms of collaborative authoring include in-person reviews where a document is shown on a shared screen and a group reviews and/or edits together. Another simple collaborative authoring technique is sharing documents via email or a shared document repository for multiple authors to contribute to or review. In the following sections, more advanced collaboration modes and related issues are discussed.

Concurrent Editing

In a concurrent editing environment, multiple authors can edit a shared document in real time. Edits made by one author appear immediately in the views of the other authors. Well-known commercial applications supporting concurrent authoring include recent versions of certain Microsoft Office applications, Microsoft OneNote, Etherpad, and documents in Google Drive. Aside from a few variations, these applications all work similarly in that they are cloud-based, require sharing of the document with other collaborators, and allow updates and edits to be seen by other collaborators in real time (if shared with those collaborators).

Concurrent editing has the obvious advantage of allowing real-time collaboration between two or more remote authors, which closely mimics working together side-by-side at a single workstation or whiteboard, especially when paired with an additional voice or chat communication channel to discuss ideas. This is especially useful for authoring where ideas are not fully developed, and require discussion, negotiation, and agreement by the authors.

Roles

In the context of intelligent tutoring, collaborative authoring implies a team of two or more individuals working together to create an intelligent tutor. In some cases, the team members may be peers, in which case the team may exist for no other reason than to divide the workload or support peer reviews. However, a more likely scenario is that the team consists of individuals with differing skills and backgrounds that are brought together to leverage their complementary talents. Thus, before considering the nature of role-based collaboration, we first define some common roles of potential collaborators.

Key authoring roles include the following:

Instructional System Designer This is a person with experience and/or formal training in the design and construction of instructional systems. A person in this role is well founded in learning theory and the application of current technology to the learning process.

Subject Matter Expert Within the context of a given authoring project this is the person with advanced domain knowledge in the area to be trained. The expertise could be from advanced education in the area, life experience, or both.

Course Facilitator This is the person(s) that will be responsible for delivering the training to the end-user (learners). They could be an actual instructor in a blended learning environment or simply a training coordinator.

Supporting roles include the following:

Educational Psychologist This is a person with an expertise in the science of learning from both a cognitive and behavioral perspective.

Software Engineer The existence of this role reflects the idea that certain ITS capabilities require expertise in programming, formal logic, or other specialized skills. Thus, the software engineer's role is to manage and/or implement any lower-level system requirements or configuration items that are either not handled by the authoring tool's user interface or require strong technical expertise.

Experimenter Given that GIFT and other ITSs are often used as research tools, experiments are an important part of the ecosystem. This role involves implementing an experimental design and collecting the correct types and quantities of data to satisfy the objectives of an experiment.

Reviewer This is a role that exists to capture and approve learners' completion or results, with responsibility for review and approval. An example would be a training compliance officer within a corporate environment. This role may overlap with other roles, particularly the course facilitator.

Administrator This is a system-level role. Users with administrative privileges have the ability to configure authoring tool and application-wide settings, perhaps including adding and/or approving new users to the site and assigning roles.

It's worth noting that the composition of authoring teams is likely to vary widely from one organization to another and even from one project to another within an organization. In many cases, a single individual may support multiple roles, and in other cases, multiple individuals may share the same role.

Additionally, though the set of roles described above may be sufficient for many authoring environments, the system should not limit users to the roles in this set. Rather the system should support the arbitrary creation of new roles via assignment of access levels and privileges.

Role-Based Access Control

Controlling access to project resources based on role is valuable for collaborative ITS authoring. It is a ubiquitous concept in multi-user information technology (IT) systems. Collaborators are assigned one or more roles on a per-project basis, and their access to resources is constrained by their least restrictive role. Allowing "read" and/or "write" privileges for each role may be sufficient for most projects, although "create" and "delete" for management roles may also be required. Such constraints serve to declutter views and minimize unwanted and potentially costly erroneous operations.

It is worth considering the granularity at which privileges can be set, as too fine a granularity can be overwhelming for those setting privileges, but too coarse a granularity may leave gaps where a user has too few or too many privileges. Fine granularity gives the administrator the most control, but coarse granularity is easier to implement. In places where fine granular control is appropriate, the burden should be minimized by cascading changes on nested resources and resource elements.

Both the organizational level and the project level of a collaborative authoring system should allow role-based access control. Roles and permissions established at the organizational level would become defaults for any new project, but could be customized by the project as needed, simplifying initial setup.

Role-Based Interface Customization

In light of the roles previously described, it is clear that different collaborators may need to interact with the authoring system in substantially different ways. Some roles have completely non-intersecting skills and experience, and may author different parts of the course. All portions of the course under development frequently require input and/or review by more than one user. Displaying content in a form that is natural to the author or reviewer should be considered a best practice.

Multiple viewers/editors can be built in to the authoring system to provide an intuitive interaction for a user based upon the role(s). The working assumption is that users with a given role will have similar

expectations and technical abilities. For example, many software engineers may prefer editing content as raw XML, whereas subject matter experts may prefer a graphical drag-and-drop interface.

At a minimum, interfaces must have two modes: one that allows editing and one that is simply for review where edits are not permitted. For this functionality, the interface would remain effectively the same. This level of interface customization may be sufficient for some portions of the authoring system, while others would benefit from fully separate views of the data. There is a trade-off in terms of effort required to implement additional interfaces and a pay-off in terms of usability of those interfaces. Accordingly, analysis and input from potential users in each of the target roles must drive the decisions to implement each additional interface (i.e., build to meet demand).

Workflow

Role-based access controls constrain who and what can be edited, while a workflow typically (though not always) imposes constraints based on timing, sequencing, and roles. Enterprise document management systems offer examples of formal document workflows, such as Microsoft SharePoint. GIFT authoring is currently unconstrained by workflow. Courses can be authored in a top-down or bottom-up fashion, and any and all aspects of a GIFT course can be edited at any time. If workflow is desired, it must be agreed upon and managed by the collaborators themselves.

Given the extreme flexibility and generalized nature of GIFT, *low-level* authoring is unlikely to ever be constrained by workflow. Nevertheless, implementing *support* for workflow for high-level authoring could have several advantages for collaborators, including division of labor, support for review/approval processes, assignments based on expertise, or enforcement of authoring best practices.

A system of note in this regard is EasyGenerator (www.easygenerator.com), a commercial cloud-based adaptive system, which supports both collaborative authoring and built-in workflow. In EasyGenerator, authoring is performed using a didactic approach. Authors first enter learning objectives based on course goals. After goals and objectives are established, authors enter questions used to evaluate student learning of goals. Finally, learning content is added/authored. Content can be added separately or it can be tied directly to a question.

For GIFT, workflow support could be created at one of three different levels. The first and simplest level would be to provide built-in support for one or more pre-defined workflow templates, analogous to the EasyGenerator approach. The second level would integrate a workflow engine into the authoring system and provide a means to upload (or choose from previously uploaded) workflow configurations created outside of the authoring system. The third, and most sophisticated, approach builds on the second but includes support for creating the workflow definition within the authoring tool itself.

Before developing any workflow, it is essential to solicit input from the user community. This is especially true for the first level, given that user-configurable workflow would not be supported. For the second and third approaches, a key step is to identify a suitable workflow engine. One promising option in this regard is jBPM (<http://www.jbpm.org/>), an open-source business process management (BPM) suite, which includes, among several features and tools, an extensible pure Java workflow engine supporting the Business Process Modeling Notation specification (www.omg.org/spec/BPMN/2.0/). Of course, jBPM is just one of many open-source workflow engines that might be applied to this purpose (for more examples, see java-source.net).

Inline Support for Collaborator Communication

Support for inline communication is an appreciated feature in many collaborative environments. This functionality is primarily provided by two modes of communication in current technology. The first allows real-time conversations/discussions between collaborators with a global real-time chat capability. Applications such as Google Chat provide this capability and are widely available for no cost. For many use cases, this may be sufficient; however, there is some advantage to having the capability built in to the collaborative authoring system. With a built-in capability, a record of the conversation could be saved as part of the course “project” and then referenced in the future. Also, because the current state of the course is readily available on their screen, collaborators are able to more easily reference the material they are discussing.

The second mode of communication is per-element annotations that can be associated with various aspects of the course. This functionality is seen on the review tab in Microsoft Office products, which allow “comments” on specific parts of a document. Such a feature enables asynchronous communication between authors concerning specific aspects. For example, reviewers could use it to note confusion or mark something needing improvements during the review process.

Should the GIFT team decide to implement support for comments, decisions must be made as to the appropriate level of granularity. In the case of Microsoft Word, comments can be inserted/attached to something as small as a single character. However, as a practical matter for GIFT, it may be best to keep the comments fairly coarse to avoid introducing unnecessary complexity to the authoring tools. There are also issues about the portability of comments across multiple authoring interfaces for the same data (e.g., raw XML vs. a form-based tool).

Social Networking

Collaborative authoring is social by its very nature. However, beyond the obvious, it is uncertain exactly what role social networking should play. It may be that social networking in the larger sense has more of a role in the end-user/learner experience than in the authoring process itself. In this case, the authoring system would clearly require support for configuration of the social networking aspects of the runtime environment, perhaps on a per-course basis, and could then provide a view into data generated via the social interactions as a means to inform ongoing course development. Furthermore, it is not uncommon for instructors to engage with learners in a social learning context, so a mechanism to support this may also be required. Given the assumption of a cloud architecture, it is easy to envision the instructor’s interaction being mediated through the authoring system itself, blurring the distinction between the authoring system and the runtime system.

On the other hand, in the event that GIFT (or any other ITS) is deployed as large-scale SaaS platform, there may be a role for social networking in authoring. One can imagine, for example, the authoring system allowing authors from different organizations sharing resources, ideas, etc. Of course, this is impractical for commercial enterprises based on proprietary intellectual property but fits well with various open education initiatives. In general, this area has a wide variety of areas for investigation and requires significant further research.

Version Control and Course Publication

Version control of documents is essential in a collaborative authoring system. The idea is to protect work progress against inadvertent changes and deletion by saving revisions of the work as it progresses. In the

event that unwanted changes or deletions are made, the system provides a mechanism to roll back to an earlier revision.

GIFT currently manages course configuration and content at the file level, while surveys are managed as entries in a relational database. The first step for revision management would be to manage revisions at these same levels. Concurrent editing requires a more sophisticated approach than file-level management. One approach would be to abandon the notion of files and store configuration and content items as objects in a database. In this way, revisions can be tracked at more granular levels. This approach also supports other ideas described in this document such as access constraints, workflow, and comments. For versioning surveys, database schema changes would be required.

Currently GIFT course authoring and publishing are decoupled. Thus, after authoring, a second explicit step, using the GIFT export tool, must be taken by the author to export an authored course—a process which packages up one or more GIFT courses, including copies of required resources, in a form suitable for distribution. After receiving the distribution, the recipient of the exported course must explicitly import the course into a GIFT instance.

Once the GIFT authoring tool and GIFT content both reside in the cloud, the distinction between a course that is under development (i.e., being authored) and one that's ready for use will be blurred. Courses in-progress may reside in the same repository and (depending upon the implementation) may actually reference some of the same shared files. The act of publishing a course then becomes an operation that provides visibility and access to a particular revision(s) of a set of course resources, rather than the physical act of copying files. Additional refinements to the course would be saved as later (non-published) revisions, that can be published if desired.

Course Resource Metadata

A potentially valuable feature for the authoring system would be support for metadata tagging of course resources. Such a capability is probably best categorized as a *like-to-have* feature more than a *must-have*, but is certainly worthy of consideration. Such a scheme would be useful for capturing and managing documentation of rationales for key decisions, references for content acquired from third parties and other data relevant to the authoring process. Having such data stored and available alongside the corresponding resource could be useful to authors in the same way that inline code comments are useful to computer programmers. The true value of such metadata is often fully appreciated (either by their presence or their absence) only when the content is revisited or modified at some point in the future, particularly by a new author.

Managing metadata at the file level could be done as a sub-element of the project construct and/or as part of a shared content repository. Approaches for finer grained management vary depending on the resource. For example, metadata for objects in a database (e.g., surveys) are probably best handled by extending the database schema appropriately. Given that GIFT already supports metadata tagging of content for pedagogical purposes, there may be some opportunity for synergy or reuse.

Usability Metrics

Any new authoring system should include support for capturing usability metrics. The objective is to log user interactions with the authoring system and then, once a sufficient dataset is collected, perform an analysis on the data to better understand how the system is used. Lessons learned from the analysis can be applied to improve the application's user experience in forthcoming releases.

At a minimum, the application should be instrumented to capture the following time-stamped data:

- User navigation to the functional areas of the application
- User access to the help system
- Usage errors (e.g., errors caught by input validation)
- Server response times

Finer-grained instrumentation could include detailed logging of user interactions (e.g., mouse clicks) with widgets contained in the different functional areas. Also, although the value of the help system can often be inferred from surrounding user interactions, it may be worthwhile to directly ask users of the help system if the provided help was satisfactory via a simple checkbox conveniently and unobtrusively located within the help display.

Details about data analysis must be left for another discussion; however, it is worth noting that users must be tracked individually, rather than collectively, and the analysis should not be viewed as a static data set but rather should track how user behavior changes over time. Doing so enable inferences to be made about collaboration as well as how individuals and teams increase in their proficiency over time.

Integration with Third-Party Authoring Tools

GIFT currently has some level of integration with four systems: AutoTutor (Graesser, Chipman, Haynes & Olney, 2005; Nye, 2013), the Student Information Models for Intelligent Learning Environments (SIMILE) Workbench (Goldberg & Cannon-Bowers, 2013), Tools for Rapid Development of Expert Models (TRADEM) (Brown, Martin, Ray, & Robson, 2014), and RapidMiner (Hoffman & Klinkenberg, 2013). None of these authoring tools are integrated seamlessly, but the design of GIFT is meant to support authoring of ITSs via external (third-party) applications that deliver content and user experiences within the context of a GIFT course. Indeed, the current version of GIFT includes sample courses that use AutoTutor, Virtual Battlefield Simulator 2 (VBS2), Tactical Combat Casualty Care Simulation (TC3Sim; Sotomayor, 2010), PowerPoint, and others. In each case, scenarios and/or content was developed using the training application's respective authoring capabilities.

As a matter of practicality, it is not feasible for GIFT authoring tools to integrate with more than a small subset of possible third-party authoring tools, although these integrations are beneficial. Integration with third-party tools means that each new release of either system incurs a significant burden of ongoing testing and maintenance. In addition, many third-party authoring tools exist only as proprietary desktop applications and very few expose the requisite functionality via an application programming interface (API), making cloud-based integration difficult if not impossible. Hence, as a general rule, external authoring tools will not/should not be integrated, but rather should remain as independent tools, the output of which is used as input to the GIFT authoring system.

A compelling use case driven by either a unique technical capability and/or substantial user demand could motivate an exception to this rule. Of course, the extent to which such integration can be made seamless would vary based upon technical feasibility. One of the four systems currently integrated with GIFT, AutoTutor is an ITS unto itself and offers the most promise in terms of authoring integration.

AutoTutor's compelling capability is that it provides the ability to engage learners in two-way dialogue driven by computational linguistics and semantic analysis. Additionally, the AutoTutor runtime has been integrated with GIFT for some time now. More recently the AutoTutor Script Authoring Tool has been

released as a web application (Nye, Graesser, Hu & Cai, 2014), and thus is well suited for integration with any web or cloud based authoring system for GIFT.

There will likely be increased interest and opportunities for integration with third-party authoring systems as the GIFT authoring capability matures and its popularity grows. Each potential integration partner system must be considered based on its merit and weighed against competing opportunities. An alternative approach might be to make a common plug-in API for third-party authoring tools, though this might be complex to implement in a cloud-based environment. Since GIFT is an open-source project, this sort of specialized integration may best be left to third-parties with a vested interest in the success of the respective authoring system.

Mobile

Without question, GIFT should support mobile learning; however, mobile authoring seems less of a priority. Desirable though it may be, there are simply too many competing priorities. The work of creating and maintaining platform specific (iOS, Android, Windows, etc.) apps as well as addressing the concern for minimizing bandwidth seems like an unnecessary burden at this time.

That said, it would be wise for ongoing ITS authoring development to proceed with a mobile future in mind. At the very least, developers should be acquainted with mobile best practices so as to architect the system in such a way as to facilitate migration in the future. Until such time, mobile considerations may best be limited to designing web pages to render effectively on mobile devices.

Scalability and Cloud Architecture Considerations

Earlier we made the assumption that any collaborative ITS authoring system would be best constructed as a cloud-based web application. However, the discussion thus far, save for a brief mention of the advantages of deploying within the cloud, has relied very little on cloud technology per se. In fact, the only real assumption has been that an authoring tool would be Internet accessible and support multiple concurrent users. For small-scale use, a traditional web application would be sufficient. In theory, the entire system could reside on a single host, perhaps augmented by a second host for database operations.

For enterprise-level deployments, more sophisticated architectures are required to take full advantage of the cloud, especially in the area of scalability. Perhaps the greatest scalability challenge would arise from offering GIFT (inclusive of the authoring system) as a SaaS platform. In such a case, there would simply be a single GIFT presence in the cloud, which would scale to meet demand as new organizations and their users came on board.

To gracefully support this level of scalability, GIFT must be architected for and implemented on a cloud infrastructure, either platform as a service (PaaS) or infrastructure as a service (IaaS). While a detailed discussion on the implications of these choices is beyond the scope of this chapter (see Mell & Grance, 2011 for an overview), PaaS would allow developers to start at a higher level of abstraction and thereby accelerate development. The trade-off, of course, is that PaaS ties the application to the chosen platform, reducing, and perhaps even, eliminating, any hopes for portability. This may be irrelevant for a commercial enterprise, but may be of some concern for an open-source project such as GIFT. Conversely, the choice of IaaS will tend to maintain a higher level of portability at the expense of development time and long-term maintenance expense.

A particularly interesting IaaS option is OpenStack (www.openstack.org), an open-source IaaS platform. Ignoring the relative merits of OpenStack vs. other IaaS options, OpenStack has the unique advantage of

being available both through commercial OpenStack cloud service providers, while also being deployable to organization-owned hardware for an internally owned and operated cloud.

Lastly, regulatory and compliance requirements, such as International Traffic in Arms Regulations (ITAR), must be considered for certain applications by US government agencies and contractors. Amazon Web Services, for example, offers Amazon Web Services (AWS) GovCloud (2015) to address this concern. In general, service providers have been expanding to fill these types of spaces, with specialized support for government needs and also Health Insurance Portability and Accountability Act (HIPAA) privacy regulations.

The IaaS/PaaS choice is just the first of several architectural considerations, where getting the architecture right is the fundamental design that will determine scalability. The bottom line is that development of any cloud-based authoring system must be preceded by a thorough analysis of cloud architectures, in light of current and anticipated system requirements.

Recommendations and Future Research

Future success of advanced ITSs will depend on the availability of collaborative authoring tools. Any effort to develop the next generation of such authoring tools should be preceded by a thorough analysis including the following:

- Detailed examination of the design considerations as outlined here,
- Review of analogous tools such as WebProtégé and EasyGenerator, and
- Input from the user community to identify design considerations and priorities.

Once objectives and priorities for the authoring tool are established they must also be put into the larger context of schedule and budget for the ITS as a whole. Tradeoffs will have to be made between advancing the capabilities of the ITS itself and advancing the authoring system.

Given the rapid pace of development of GIFT (and presumably other ITSs) authoring tool design should plan for change. To the greatest extent practical, the authoring system should be built with appropriate abstractions, perhaps as a framework, so that authoring for new ITS capabilities can be added with minimal changes to the system as a whole.

Finally, while authoring tools are currently mainly used to support research on ITSs and their capabilities, a sophisticated collaborative authoring environment could offer a testbed for research on the psychology of collaboration. Even in the short term, quantitative research studying the performance and efficiency of the ITS authoring systems is an important direction. As such, moving forward, identification and analysis of a common set of usability metrics is probably an important step forward.

References

- Amazon Web Services. (2015, Mar 20). Retrieved from AWS GovCloud (US) Region - Government Cloud Computing: <http://aws.amazon.com/govcloud-us/>
- Brown, D., Martin, E., Ray, F. & Robson, R. (2014). Using GIFT as an Adaptation Engine for a Dialogue-Based Tutor. *Proceedings of the Second Annual GIFT Users Symposium (GIFT Sym2)*, (pp. 163-174).
- Easy Generator. (2015, Mar 20). Retrieved from www.easygenerator.com

- Elkins, D. (2013, January 24). *E-Learning Authoring Tool Comparison*. Retrieved from E-Learning Uncovered : <http://elearninguncovered.com/2013/01/e-learning-authoring-tool-comparison/>
- Goldberg, B. & Cannon-Bowers, J. (2013). Experimentation with the Generalized Intelligent Framework for Tutoring (GIFT): A Testbed Use Case. *AIED 2013 Workshops Proceedings Volume 7*, (pp. 27-36).
- 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.
- Hoffman, M. & Ragusa, C. (2014). Unwrapping GIFT: A Primer on Authoring Tools for the Generalized Intelligent Framework for Tutoring. *Generalized Intelligent Framework for Tutoring (GIFT) Users Symposium (GIFTSym2)*, (pp. 11-24).
- Hofmann, M. & Klinkenberg, R. (2013). *Rapidminer: Data Mining Use Cases and Business Analytics Applications*. Chapman & Hall/CRC.
- Mell, P. & Grance, T. (2011). *The NIST Definition of Cloud Computing (800-145)*. National Institute of Standards and Technology (NIST).
- Murray, T. (2003). An Overview of Intelligent Tutoring System Authoring Tools: Updated analysis of the state of the art. In T. Murray, S. Blessing & S. Ainsworth, *Authoring tools for advanced technology learning environments* (pp. 491-544). Springer.
- Nye, B. D. (2013). Integrating GIFT and AutoTutor with Sharable Knowledge Objects (SKO). *AIED 2013 Workshop on GIFT*, (pp. 54-61).
- Nye, B. D., Graesser, A. C., Hu, X. & Cai, Z. (2014). AutoTutor in the cloud: A service-oriented paradigm for an interoperable natural-language ITS. *Journal of Advanced Distributed Learning Technology*, 2(6), 49-63.
- Nye, B. D., Rahman, M. F., Yang, M., Hays, P., Cai, Z., Graesser, A. & Hu, X. (2014). A tutoring page markup suite for integrating shareable knowledge objects (SKO) with HTML. *Intelligent Tutoring Systems (ITS) 2014 Workshop on Authoring Tools*.
- Open Source Workflow Engines in Java*. (2015, Mar 20). Retrieved from Java-Source.net: java-source.net/open-source/workflow-engines
- Pappas, C. (2013, March 12). *The Ultimate List of Cloud-Based Authoring Tools*. Retrieved from eLearning Industry: <http://elearningindustry.com/the-ultimate-list-of-cloud-based-authoring-tools>
- Schneider, P. (2012, June 18). *Content Authoring Tools: Cloud-Based or Desktop?* Retrieved from Learning Solutions Magazine: <http://www.learningsolutionsmag.com/articles/952/content-authoring-tools-cloud-based-or-desktop>
- Sotomayor, T. M. (2010). Teaching tactical combat casualty care using the TC3 sim gamebased simulation: a study to measure training effectiveness. *Studies in health technology and informatics.*, 154, 176-179.
- Sotilare, R. A., Goldberg, B. S., Brawner, K. W. & Holden, H. K. (2012). A modular framework to support the authoring and assessment of adaptive computer-based tutoring systems (CBTS). *Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC) 2012.*, Paper No. 12017, pp. 1-13.
- Tao, T. (2015, Feb 28). *Articulate vs. Captivate: The complete series*. Retrieved from Fredrickson Communications: fredcomm.com/articles/detail/articulate_vs_captivate_comparing_popular_rapid_elearning_development_tools
- Tudorache, T., Nyulas, C. & Noy, N. F. (2013). WebProtégé: A collaborative ontology editor and knowledge acquisition tool for the web. *Semantic Web*, 4(1), 89-99. Retrieved from WebProtege - Protege Wiki: <http://protegewiki.stanford.edu/wiki/WebProtege>
- Whitehead Jr., E. J. & Wiggins, M. (1998). WebDAV: IEFT standard for collaborative authoring on the Web. *Internet Computing, IEEE*, 2(5), 34-40.

Chapter 10 Authoring for the Product Lifecycle

Steve Ritter
Carnegie Learning

Introduction

Intelligent tutoring systems (ITSs) and other adaptive learning environments have been developed and tested for many years, and there is substantial evidence that they can contribute to significantly better student outcomes (Van Lehn, 2011; Pane et al., 2014). However, such systems have found limited use in schools and training programs. In part, this reflects a mismatch between the traditional educational environment, which holds time fixed and aims to teach students as much as possible within that time, and adaptive systems (and other mastery environments), which aim to allow students to define levels of student mastery and then provide enough instruction to allow students to reach that level of competency, however long it takes.

Within the authoring tool community, there is another theory about the relatively slow adoption of adaptive learning environments: they are too expensive to produce. In the classic volume on such authoring tools (Murray, Blessing and Ainsworth, 2003), there are two stated reasons for developing authoring tools: “to reduce development cost, and to allow practicing educators to become more involved in their creation.” (p. iv). In both of these goals, we have primarily focused on the creation of the instructional systems (c.f. Blessing, 2003; Razzaq and Heffernan, 2010; Aleven, et al., 2006). Some systems have focused on reuse of existing systems (Ainsworth, et al., 2003; Ritter and Koedinger, 1996), but even these take the creation of a new system from existing parts to be their goal.

It is important that authoring tools for intelligent tutoring systems focus on being able to create new systems quickly and on making authoring accessible to teachers and content experts who are not sophisticated programmers. But if ITSs are to become widespread and in regular use, they need to also focus on features that allow these systems to be maintained and improved over time. One of the primary advantages of ITSs is that they allow us to collect detailed data on student learning, which can help us improve the educational outcomes of the systems themselves. We call this focus on continual improvement “authoring for the product lifecycle.”

The Far-Outer Loop

VanLehn (2006) describes tutoring systems as containing an inner loop and an outer loop. The inner loop relates to the tutor’s behavior at each step of a complex task; the outer loop is responsible for choosing tasks for a student. In fact, the inner and outer loop description applies more generally to adaptive systems. Within adaptive systems, inner-loop behavior is responsible for guiding students through a task, including providing hints and feedback for the student, diagnosing errors and adapting to different methods of problem-solving that the student might employ. The outer loop helps the system adapt to the student by assessing the student’s level of knowledge at a higher level. The outer loop sets appropriate pacing for the student (for example, by assessing mastery and allowing or recommending that the student progress to the next topic when master is obtained) and picks appropriate tasks for the student to complete (typically aiming to select tasks that emphasize skills that are within the student’s zone of proximal development). In this way, tutoring systems adapt both within-task and across tasks.

Product-lifecycle authoring introduces an additional form of adaptation, taking place in what could be referred to as the far-outer loop. This loop encompasses changes to the tutoring system at a timescale larger than the task level. These changes represent improvements to the system itself that are made based on data collected from prior users of the system. The goal of a tool focused on authoring for the lifecycle is to enable rapid and relatively inexpensive responses to data collected from the system, so that future students using the system will have an improved experience. When considering such a system, we need to consider the possible changes to the system that can result from this data collection and some models for how to implement changes to the tutoring system itself.

Types of Data-Based Changes

Our first consideration for product-lifecycle authoring is the type of changes that we might make to systems based on these data. We consider four types of changes: those affecting system parameters, those focused on instructional design changes, those addressing content, and those affecting the ability of the system to be personalized for different types of students. These types of changes differ in the extent to which they require extensive changes to the system and the extent to which they employ human judgment (and thus cannot be easily automated).

Parameter Changes

A common type of change to ITSs is to adjust the parameters that control how the system reacts to students. For example, model-tracing tutors typically assess student knowledge with respect to discrete skills, also known as knowledge components. The system's task is to assess each student's mastery of each knowledge component. Many tutors perform this task through Bayesian knowledge tracing, which employs four parameters for each knowledge component (Corbett and Anderson, 1995). Two of these parameters represent estimates of knowledge: the probability that the student has mastered the knowledge component prior to instruction and the probability that the knowledge component will be mastered, given an opportunity (this parameter is basically controlling the ease of learning the knowledge component). Since knowledge is not perfectly reflected in performance, Bayesian knowledge tracing also uses two performance parameters: the probability that the student will guess the correct answer (i.e., answer correctly without having mastered the underlying knowledge) and the probability that the student will "slip" (i.e., answer incorrectly, even though the student does possess the requisite knowledge). Since each of the four parameters is considered a probability, each can vary between 0 and 1, although there are various reasons why particular areas of this four-dimensional problem space may not be used (Beck and Chang, 2007; Ritter et al., 2009).

Since settings of these parameters control task selection and mastery determination, they are essential components in implementing the outer loop of a tutoring system. Although the settings of these parameters is crucial to proper behavior of the system, the parameters are typically based on the intuitions of the developers in the initial release of a tutoring system. Once data are collected from students, best-fitting values for these parameters can be found (Cen et al., 2007; Gonzalez-Brenes et al., 2014; Khajah et al., 2014), and Cen et al. (2007) demonstrated that modifying the system to use the discovered parameters can produce better outcomes.

Beyond fitting Bayesian parameters within knowledge components is the question of whether the task is being modeled with the correct set of knowledge components. Decomposing a task into the knowledge components that best explain learning is also typically done based on intuition (informed by cognitive task analysis). Here, too, there is a need for empirical refinement. Koedinger et al. (2012) demonstrate that models found through data-fitting provide significant improvements over the initial intuition-driven

model. Thus, authoring systems that manage changes to such parameters over time can provide significant benefit to a widely used system.

Design Changes

Adjusting knowledge tracing parameters can correct inefficiencies in the way that the tutoring system navigates the outer loop. If a deficiency in the system involves the inner loop (the nature of the task itself), changes may require fundamental changes to the task model itself. Dickison et al. (2010) found that parameter adjustments made on the basis of previously collected data correctly modeled a new student cohort, except in the case of an instructional unit that had undergone design changes. Since design changes can negate the validity of changes to knowledge tracing parameters, it is essential that authors wishing to improve a system be able to understand whether improvements can be achieved through parameter changes or if they require design changes. In a system maintained for any length of time, there are always a long list of potential design changes to be made. Some are driven by customer requests; others by technical changes. If changes are to be made on the basis of the potential for improvements in the instructional effectiveness of the system, then a lifecycle authoring system needs to provide guidance to authors that can help prioritize these improvements and predict their likely impact.

While it is difficult to provide general guidance on identifying design errors, Carnegie Learning's experience suggests a few heuristics that could be helpful in prioritizing design changes. Internally, we use an "attention metric," which combines several indicators of educational ineffectiveness, to which we (as authors) must direct our attention. The most important relates to "wheel spinning" (Beck and Gong, 2013), the case where students fail to master a skill in what is considered a reasonable amount of time. A pure mastery learning system will continue to try and instruct such a student, even if no progress is being made. In our tutors, we terminate instruction on this topic after some period of time and notify a teacher that the student has failed to master the topic. Instructional topics that produce a large number of such notifications are strong candidates for redesign. In fact, parameter fitting on such units may be counterproductive. If a particular unit is not producing improvements in performance, then fitting parameters based on the data might lead to a near-zero probability of mastering the skill on an opportunity, which would result in such a system wanting to present even more ineffective instruction to students.

Another factor we have found useful in our "attention metric" concerns the way that teacher treat units of instruction. Teachers have control over inclusion of units of instruction within a curriculum, and units that are often excluded are good candidates for scrutiny. Similarly, teachers can manually skip students past particular problems, and the record of the frequency of this kind of behavior can indicate that those problems are perceived to be confusing or otherwise ineffective.

Content Changes

One form of task change within a tutoring system involves changes to the content presented within a task, rather than the basic structure of the task. Such content changes could be driven by user feedback to the authors (e.g., ratings of helpfulness or enjoyment of particular activities), or the desire to allow end-users to customize their system (Heffernan & Heffernan, 2014), increase the number of task contexts, or increase the variety of contexts.

Depending on the sophistication of the task model and architecture of the overall system, content authoring might employ general tools that can easily be used by non-programmers, or they might employ special-purpose tools, whether for programmers or not (Ritter et al., 1998). In a lifecycle authoring tool, the particular concern for content is in managing the data about particular pieces of content. Such a

system needs to track what elements of content are being used and (if available), which receive high or low ratings from users. A particular concern related to lifecycle content authoring is the issue of problem morphs. Tutoring systems assume that problems that are modeled with the same knowledge components and delivered with the same task model are educationally equivalent. A lifecycle authoring system needs to provide tools to determine whether this assumption is justified. If some problems prove to be unexpectedly difficult or easy, then either the task model or the knowledge component model will need to be adjusted.

Personalization

A particularly compelling type of change to a tutoring system is one that personalizes the system such that different students receive different educational experiences. Such personalization changes would be warranted, for example, if data showed aptitude-treatment interactions: that a particular tutoring approach worked well for students with certain characteristics but that a different approach worked best with students possessing different characteristics. Many people have strong intuitions that aptitude-treatment interactions are pervasive in education and, particularly, believe that learning styles reflecting a preferred presentation mode (such as verbal vs. visual) reflect such interactions, but the evidence for this is weak (Pashler, et al., 2009). More modest forms of personalization do seem to be effective (Ritter et al., 2014). An important consideration for lifecycle authoring tools would be identifying potential opportunities for treating individuals or classes of individuals differently within a tutoring system. Yudelson et al. (2014) describe a technique for identifying whether a tutoring system should treat subclasses of students differently for the purpose of knowledge tracing.

Models for Applying Changes

The previous section focused on the types of changes that we might want to make to a tutoring system, based on data collected from that system. Another dimension to be considered in a lifecycle authoring system is the model for approving and applying such changes to the system. We consider three types of models. In the “manual” model, the data are analyzed and reviewed by authors before changes are made to the system. In the “automated” model, the data are collected and analyzed by a stored set of procedures, and the changes to the system are automatically applied. The “crowdsourced” model combines aspects of the human judgment applied in the manual model and the programmed changes used in the automated model. In this case, changes contributed by users can be automatically incorporated into the system, making users authors. But the model might also have a publishing model where a central authority approves changes or where users (or particular categories of users) approve changes or control who has access to the changes. A key issue within each of these models is determining when our confidence in the data currently collected justifies making changes to be seen for future users. Given the concerns of personalization and, in some cases, uncertainty about how user populations may change over time, this is a difficult statistical issue that has not received enough attention.

The Manual Change Model

The manual change model is a model of iterative change with humans (typically learning scientists) in the loop. In this model, instruction is often instrumented to provide feedback about what elements of instruction are most effective. Sometimes, A/B tests (randomized field experiments) are employed, providing data directly relevant to future improvements; other times, more naturalistic data collection is involved.

The Open Learning Initiative (oli.cmu.edu) courses are good examples of how manual iterative refinement can produce more effective courseware (Thille, 2008). These courses collect extensive data, from embedded tutors, manipulatives and other embedded activities that provide extensive information about the effectiveness of various aspects of individual courses. In many cases, it can be relatively easy to identify areas for improvement in a course, but there is often a large space of potential design improvements available to remedy the flaws. Instead of relying solely on in-house expertise, the OLI project aims to develop a community of practice, sharing results on elements of the courses and soliciting ideas for improvement.

Improvements to Carnegie Learning's geometry tutor represent another example of the manual change model (Butcher and Aleven, 2008; Hausmann and Vuong, 2012). Over a period of several years, iterative improvements in the design of a tutor teaching reasoning about angles in a geometric diagram were conducted, focused on more closely following research on self-explanation and on the contiguity principle (Clark and Mayer, 2011). The process involved a series of lab-based and small field design experiments, which eventually led to large implementations and field evaluations. Results showed that students were able to reach mastery in less time and with the need to complete fewer problems in the improved version of the tutor.

The Automated Change Model

The manual change model is quite flexible, potentially leading to a wide variety of changes, but it is labor-intensive and can take years of effort to produce improvements. The automated change model may be more limited in its scope, but automated changes are able to be applied much more quickly.

The basic idea behind the automated change model is that one can pre-specify a design space of potential approaches to instruction (or parameterizations of approaches). The system is then able to explore the design space, collecting data on what approaches work best.

Liu et al. (2014a) used Learning Factors Analysis (LFA; Cen, et al., 2007) to automatically discover knowledge component models that best explained previously collected data from cognitive tutors. In this process, the author specifies a set of knowledge components that might potentially represent relevant learning factors. For example, in modeling the ability for students to take the area of geometric figures, the orientation of a triangle (base parallel to the ground or not) may or may not cause difficulties for some students. These skills become parameters for potential use in a predictive model. LFA is also able to "merge" knowledge components to produce new parameters. For example, the LFA model discovered that computing area "backwards" (calculating one of the linear measures, given the area) was a difficulty factor for circles but not for rectangles. This parameter results from merging the potential knowledge components related to particular shapes and to working backwards. In almost all cases, the models found by LFA were superior to those developed by experienced developers, even after years of manual refinement. While changes resulting from LFA have not yet been automatically applied to the tutoring system, it would be straightforward to create a system that did automatically apply the results of such an analysis. At this point, automatic refinement of this kind requires enough confidence in the technique. It is likely that such confidence would result from continued demonstrations that such changes not only produce improvements in model fits but that applying such improvements produce real-world improvements. Some such randomized field trials are currently underway.

One approach that is inherently fully automated is the use of multi-armed bandit procedures (Liu et al., 2014b). As with LFA, the multi-armed bandit approach starts with a specification of a design space for the application. The approach typically works well with large spaces that can be parameterized. The approach performs a search of the design space in the field, presenting different variants of the

educational system to different students. Designs (defined by sets of parameters) that work best (by whatever metric is able to be used in the field) become probabilistically favored in selection for new students. Eventually, the system converges on a design that works best for users. One important consideration in this type of system include balancing the need for exploration of the design space and the desire to exploit the parameters representing the most effective variant of the system. Implementations of this kind of system must also be in contexts where it is reasonable to measure effectiveness quickly and reliably.

One concern with automated change models, particularly in commercial systems, is maintaining some control and knowledge over the changes made. If we are to rely on automated changes, we need to be very certain that the changes made will result in better performance, not just for typical students but across the whole range of students using the system.

The Crowdsourced Change Model

Adaptation to different student populations is a strength of the crowdsourced change model. In this model, users (or some subset of users) are able to contribute to the improvement of the system, either by creating new content or providing feedback on existing system content and features. Key to the crowdsourced change model is the ability to create a community in which users feel rewarded for their contributions. Variants of this model may be similar to the manual change model (in the case where suggested changes are centrally curated) or the automated change model (in the case where user-generated content is automatically provided to other users).

Razzaq et al. (2009) provide an example of a crowdsourced content authoring system. The ASSISTment Builder is a content authoring system allowing end-users (particularly teachers) to extend ASSISTment by writing new content. Their goal was to provide a system that is simple to use but also provides some flexibility in allowing advanced users to variabilize content, enabling users to create a large quantity of items. This new content can be immediately provided to other users, resulting in something like an automated improvement model. The system also provides a feedback mechanism for users, which provides a basis for manual improvements in the system. Users are able to point out errors or contribute suggestions for improvement in particular items.

Aleahmad et al. (2009) similarly describe an open content authoring system, grounded in creating a web-based authoring community. A particular goal of this system was to encourage a wide variety of items, which could enable the resulting system to better personalize content to address particular student interests. The system also contemplated a rating and curation system that would allow the community to vet content before it was presented to students.

The Lifecycle Authoring System and Implications for the Generalized Intelligent Framework for Tutoring (GIFT)

If ITSs and other adaptive learning systems are to achieve wide adoption, they will need to be built with the expectation that they can change over time. Lifecycle authoring systems allow these systems to capitalize on one of their most important advantages: their ability to collect and make sense of data that can result in improvement to the systems themselves. The design goals of the Generalized Intelligent Framework for Tutoring (GIFT) architecture include consideration of the use of data to improve instructional effectiveness (Sottilare & Holden, 2013), but much work remains to be done in identifying commonalities in the way this may be done across different tutoring systems and formalizing these commonalities into standard approaches to system improvement.

While different lifecycle authoring systems will take different approaches, all systems need to consider two basic dimensions: the types of changes that they support and the model for applying such changes. We would not expect a single system to be designed to support all types of changes and all models. Some change models seem particularly appropriate for particular types of changes. For example, automated change models seem particularly suited to parameter changes, since they require a description of the search space. Crowdsourcing seems particularly suited to content changes, under the assumption that content creation is a natural domain for end-users, particularly, users who are teachers. Design changes, on the other hand, may require input from programmers, instructional designers and domain experts, leading to the likelihood that such changes will be produced with a manual change process. Advance planning for the types of changes expected to be made in adaptive systems and incorporation of appropriate models for improvement will allow advanced adaptive instructional systems to become more mainstream, leading to better educational outcomes.

References

- Ainsworth, S., Major, N., Grimshaw, S. K., Hayes, M., Underwood, J. D., Williams, B. & Wood, D. J. (2003). REDEEM: Simple Intelligent Tutoring Systems From Usable Tools. In T. Murray & S. Blessing & S. Ainsworth (Eds.) *Tools for Advanced Technology Learning Environments*. (pp. 205- 232). Amsterdam: Kluwer Academic Publishers.
- Aleahmad, T., Aleven, V. & Kraut, R. (2009). Creating a corpus of targeted learning resources with a web-based open authoring tool, *IEEE Transactions on Learning Technologies*, 2(1), 3-9.
- Aleven, V., McLaren, B.M., Sewall, J., Koedinger, K.R.: The cognitive tutor authoring tools (CTAT): Preliminary evaluation of efficiency gains. In Ikeda, M., Ashley, K.D., Tak-Wai, C., eds.: International Conference on Intelligent Tutoring Systems, Springer (2006) 61–70
- Beck, J.E. and Gong, Y. (2013). Wheel-Spinning: Students Who Fail to Master a Skill. In *Proceedings of the 16th International Conference on Artificial Intelligence in Education*. Memphis, TN. pp. 431-440.
- Beck, J. E. and Chang, K. M. (2007). Identifyability: A fundamental problem of student modeling. *Proceedings of the 11th International Conference on User Modeling*, pp. 137-146.
- Blessing, S.B. (2003) A Programming by Demonstration Authoring Tool for Model-Tracing Tutors. In Murray, T., Blessing, S.B. & Ainsworth, S. (Ed.), *Authoring Tools for Advanced Technology Learning Environments: Toward Cost-Effective Adaptive, Interactive and Intelligent Educational Software*. (pp. 93-119). Boston, MA: Kluwer Academic Publishers
- Butcher, K. & Aleven, V. (2008). Diagram interaction during intelligent tutoring in geometry: Support for knowledge retention and deep transfer. In C. Schunn (Ed.) *Proceedings of the Annual Meeting of the Cognitive Science Society*, CogSci 2008. New York, NY: Lawrence Earlbaum.
- Cen, H., Koedinger, K.R., Junker, B. (2007). Is Over Practice Necessary? Improving Learning Efficiency with the Cognitive Tutor using Educational Data Mining. In Lucken, R., Koedinger, K. R. and Greer, J. (Eds). *Proceedings of the 13th International Conference on Artificial Intelligence in Education*, pp. 511- 518.
- Clark, R. C. & Mayer, R. E. (2011). E-Learning and the Science of Instruction: Proven Guidelines for Consumers and Designers of Multimedia Learning (3rd ed.). San Francisco, CA: John Wiley & Sons.
- Corbett, A.T., Anderson, J.R. (1995). Knowledge Tracing: Modeling the Acquisition of Procedural Knowledge. *User Modeling and User-Adapted Interaction*, 4, 253-278.
- Dickison, D., Ritter, S., Nixon, T., Harris, T.K., Towle, B., Murray, R.C. and Hausmann, R.G.M.: Predicting the Effects of Skill Model Changes on Student Progress. *Intelligent Tutoring Systems* 2010: 300-302.
- Koedinger, K. R., McLaughlin, E. A. & Stamper, J. C. (2012). Automated cognitive model improvement. Yacef, K., Zaiane, O., Hershkovitz, H., Yudelson, M., and Stamper, J. (eds.) *Proceedings of the 5th International Conference on Educational Data Mining*, pp. 17-24. Chania, Greece.
- Dickison, D., Ritter, S., Nixon, T., Harris, T.K., Towle, B., Murray, R.C., Hausmann, R.G.M. (2010). Predicting the Effects of Skill Model Changes on Student Progress. In *Intelligent Tutoring Systems* (2), pp. 300-302.
- González-Brenes, J.P., Huang, Y., Brusilovsky, P. (2014). General Features in Knowledge Tracing: Applications to Multiple Subskills, Temporal Item Response Theory, and Expert Knowledge. *The 7th International Conference on Educational Data Mining (EDM 2014)*. London, England

- Hausmann, R.G.M. & Vuong, A. (2012) Testing the Split Attention Effect on Learning in a Natural Educational Setting Using an Intelligent Tutoring System for Geometry. In N. Miyake, D. Peebles & R. P. Cooper (Eds.), Proceedings of the 34th Annual Conference of the Cognitive Science Society. (pp. 438-443). Austin, TX: Cognitive Science Society.
- Heffernan, N. & Heffernan, C. (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*.
- Khajah, M., Wing, R. M., Lindsey, R. V. & Mozer, M. C. (2014) Incorporating latent factors into knowledge tracing to predict individual differences in learning. In J. Stamper, Z. Pardos, M. Mavrikis & B. M. McLaren (Eds.), Proceedings of the 7th International Conference on Educational Data Mining (pp. 99-106).
- Koedinger, K.R., Stamper, J.C., McLaughlin, E.A. & Nixon, T. (2013). Using data-driven discovery of better student models to improve student learning. In H.C. Lane, K. Yacef, J. Mostow & P. Pavlik (Eds.), Proceedings of the 16th International Conference on Artificial Intelligence in Education, pp. 421-430.
- Liu, R., Koedinger, K. R. & McLaughlin, E. (2014a). Interpreting model discovery and testing generalization to a new dataset. Proceedings of the 6th International Conference on Educational Data Mining, London, UK.
- Liu, Y., Mandel, T., Brunskill, E. and Popovic, Z. (2014b). Trading Off Scientific Knowledge and User Learning with Multi-Armed Bandits. Proceedings of the 6th International Conference on Educational Data Mining, London, UK.
- Murray, T., Blessing, S. & Ainsworth, S. (Eds.) (2003). Authoring Tools for Advanced Technology Learning Environments. Kluwer Academic/Springer Pub.: Netherlands.
- Pashler, H., McDaniel, M., Rohrer, D. & Bjork, R. (2009). Learning styles: Concepts and evidence. *Psychological Science in the Public Interest*, 9, 105-119.
- Razzaq, L., Parvarczki, J., Almeida, S.F., Vartak, M., Feng, M., Heffernan, N.T. and Koedinger, K. (2009). The ASSISTment builder: Supporting the Life-cycle of ITS Content Creation. IEEE Transactions on Learning Technologies Special Issue on Real-World Applications of Intelligent Tutoring Systems. 2(2) 157-166.
- Razzaq, L. & Heffernan, N. (2010). Open content Authoring Tools. In Nkambou, Bourdeau & Mizoguchi (Eds.) Advanced in Intelligent Tutoring Systems. pp 425-439. Berlin: Springer Verlag.
- Ritter, S., Anderson, J., Cytrynowicz, M., and Medvedeva, O. (1998) Authoring Content in the PAT Algebra Tutor. *Journal of Interactive Media in Education*, 98 (9)
- Ritter, S., Harris, T. H., Nixon, T., Dickison, D., Murray, R. C. and Towle, B. (2009). Reducing the knowledge tracing space. In Barnes, T., Desmarais, M., Romero, C. & Ventura, S. (Eds.) Educational Data Mining 2009: 2nd International Conference on Educational Data Mining, Proceedings
- Ritter, S. and Koedinger, K. R. (1996). An architecture for plug-in tutor agents. *Journal of Artificial Intelligence in Education*, 7, 315-347.
- Ritter, S., Sinatra, A. M. and Fancsali, S. E. (2014). "Personalized Content in Intelligent Tutoring Systems." In Design Recommendations for Intelligent Tutoring Systems, vol. 2, pp. 71-78. Army Research Laboratory.
- Sottilare, R. A. and Holden, H. K. (2013). Motivations for a Generalized Intelligent Framework for Tutoring (GIFT) for authoring, instruction and analysis. In *AIED 2013 Workshop Proceedings*, Volume 7, 1-9.
- Thille, C. (2008). Creating open learning as a community based research activity. In Iiyoshi, T. & Kumar, V. (Ed.), *Opening Up Education: The Collective Advancement of Education through Open Technology, Open Content, and Open Knowledge*. Cambridge, MA. MIT Press.
- VanLehn, K. (2006). The behavior of tutoring systems. *International Journal of Artificial Intelligence and Education*, 16, 227–265.
- Yudelson, M.V., Fancsali, S.E., Ritter, S., Berman, S.R., Nixon, T. and Joshi, A. (2014). Better data beats big data. Proceedings of the 7th International Conference on Educational Data Mining

SECTION III

AUTHORING AGENT-BASED TUTORS

Keith Brawner, Ed.

CHAPTER 11 Authoring Agent-based Tutors

Keith W Brawner
US Army Research Laboratory

Introduction

The purpose of this introductory chapter is not to raise questions or present recommendations, but to attempt minor summary of the conversations among the literature. Many of these written works have revolved around the ideas of identification of authors, establishment of roles, reductions in complexity, and automation. The artificial intelligence (AI) community has long desired to “democratize AI” through the use of tools to encode knowledge in expert systems, neural networks, and other items. These efforts have fallen somewhat short: AI for problem solving purposes remains in the hands of engineers, scientists, and programmers. The field of intelligent tutoring systems (ITSs) has made somewhat better progress but has started looking to the agent-based AI world for solutions, making this section timely and relevant. Before addressing the topic of agents and their authoring, it is helpful to refresh the mental model of tool software lifecycle.

The Birth of a Tool

Given that necessity is the mother of invention, it is no surprise that the ITS tools, thus far, have been primarily crafted toward a single system, template, use case, or user category. The creation of a tool is frequently the last portion of the development of a system, relegated to the end of a project along with user manuals, training materials, and long-term supporting logistics. The reason for this is simple: tooling occurs after machining.

There has been little research into ITS tools as a factor of two things. The first is byproduct of being late in the developmental cycle, while the second is the multi-faceted nature of the tools. ITS authoring tools naturally involve pedagogical strategy, learning knowledge modeling and assessment, content creation and supplementation, and other items. Each of these items calls for a somewhat unique solution, and in this section, I call for a science of the pedagogical authoring process, as it relates to pedagogical agent creation (Shaffer, Ruis & Graesser, 2015).

The Life and Growth of a Tool

The life of a tool is naturally closely tied to the life of the system. At the time of system creation, there is usually no tool to speed the process of development. Developers must handcraft each of the system parts, edit configuration files by hand, and test various configurations for speed and effectiveness. After a workable solution is found, the work of creating the ITS components can be offloaded to a knowledgeable user with the appropriate background. This usually occurs with a simplistic tool, such as an extensible markup language (XML) editor, interface specifier, or simple application programming interface (API) specification. Projects such as the Generalized Intelligent Framework for Tutoring (GIFT) *AT editing tools, and the AutoTutor Script Authoring Tool (ASAT) have reached these stages by XML editing (Brawner & Sinatra, 2014) and script authoring (Nye, Hu, Graesser & Cai, 2014), respectively.

Assuming that the system survives long enough to be well used or profitable, such a knowledgeable user usually has enough of a programming background to automate part of the process, program a workflow, or otherwise decompose the authoring task into pieces. This allows the task to be performed by less knowledgeable users such as undergraduate students or interns. Projects for authoring conversational agent-based tutors such AutoTutor have recently reached this stage (Nye et al., 2015).

The Death of a Tool

Assuming the system survives long enough to reach a modicum of success, the authoring task is decomposed into component parts and performed by more junior personnel. Such a task is time consuming but uncomplicated, and automation techniques begin to become attractive time-saving items. Projects such as SimStudent (MacLellan, Koedinger & Matsuda, 2014) allow mostly automated authoring through a process of knowledge demonstration. With SimStudent, such knowledge demonstration can be performed by any user with knowledge of the domain of instruction, but relies upon the extensive architecture underpinning a simulation of the environment, measurement of actions, and tutoring system interoperability. Each one of these items could potentially have a tool to aid a category of user in assembling a system, if the situation is complicated enough to warrant it.

ITS Complicated

One of the themes that repeats itself through the ITS literature conversation is the simple fact that ITSs are complicated. The word “complicated” is used as a proxy for expensive, time-consuming, difficult to understand, and other themes. The construction of an ITS currently involves personnel with knowledge of instructional design, learner modeling, a specific domain, sensors/interfaces, machine learning interpretation of data streams, and the ability to create a student environment that is able to provide these assessments and feedbacks. Another author in this section describes the process as requiring “deep and broad knowledge to manage these constraints, accommodate tradeoffs, and negotiate incompatibilities” (Shaffer et al., 2015).

One of the goals of the GIFT project is to simplify this expertise required through the creation of interoperable “modules,” with each of them tasked with the functions above (Learner, Pedagogical, Sensor, Domain, etc.). In this manner, the hope and plan is to create a module (or module plug-in) only once, allow it to interoperate, and to extensively reuse it. Such a module could be a plan for instruction, such as the Engine for Management of Adaptive Pedagogy (Goldberg et al., 2012), or a machine learning process for interpreting sensor data from game environments and the Microsoft Kinect (Baker, DeFalco, Ocumpaugh & Paquette). This type of solution, however, raises a new problem: that of generalization.

Generalization

The problem of generalization is a discussion that permeates all conversations where GIFT is involved. The first book in the Design Recommendations for Intelligent Tutoring series attempted to summarize the problem of generalizable models of student performance (Sottilare, Graesser, Hu & Holden, 2013), while subsequent books have addressed domain-general models for instruction (Sottilare, Graesser, Hu & Goldberg, 2014), and future books intend to address the topics of assessment and teams. In each meeting to discuss each problem, the question of “how can X be done without explicit and complete knowledge of Y?” is raised, where X and Y relate to any of the other modules.

ITSs, as a category, are intended to cut across all categories of training. ITS authoring, as a category, cuts across all categories of modules and components. The unique challenges presented are how to construct tools that generalize to the general purpose of the ITS. In this regard, the authors of this section present solutions and recommendations for how this may be accomplished for agents (Cohn, Olde, Bolton, Schmorrow & Freeman, 2015), in complex domains of instruction (Shaffer et al., 2015), during assessing conversations (Zapata-Rivera, Jackson & Katz, 2015), and with pedagogical and authoring soundness (Lester, Mott, Rowe & Taylor, 2015).

Agents

One of the reoccurring themes is that, as part of the natural process of replacing a human tutor with a computer tutor, the computer tutor should be presented in the form of an agent. Agent-based software technology has struggled with many of the same problems that ITS generalization has: domain-general behaviors, user behavior recognition, user intention recognition, response planning, management of specific domain knowledge, etc. (Allen et al., 2000).

The chapters in this section are especially relevant to the agent replacement conversation. The nature of computer teaching agents is that they can teach more complex domain information, involve deeper knowledge elicitation (Rus, Stefanescu, Niraula & Graesser, 2014), and generally improve learning (Graesser, VanLehn, Rosé, Jordan & Harter, 2001). The task of creating such learning agents is difficult and worthy of discussion in this chapter. The chapters within this section provide timely and relevant descriptions of authoring tools for agent-based tutors and include descriptions of existing tools and methods that uniquely support agent-based tutors; emerging technologies for agent-based tutors; and recommendations for how GIFT should be enhanced to make authoring of agent-based tutors easier/more efficient.

References

- Allen, J., Byron, D., Dzikovska, M., Ferguson, G., Galescu, L. & Stent, A. (2000). An architecture for a generic dialogue shell. *Natural Language Engineering*, 6(3&4), 213-228.
- Baker, R. S., DeFalco, J. A., Ocumpaugh, J. & Paquette, L. Towards Detection of Engagement and Affect in a Simulation-based Combat Medic Training Environment.
- Nye, B., Hu, X., Graesser, A. & Cai, Z. (2014). Autotutor In The Cloud: A Service-Oriented Paradigm For An Interoperable Natural-Language Its. *Journal of Advanced Distributed Learning Technology*, 2(6), pp49-63.
- Brawner, K. & Sinatra, A. (2014). *Intelligent Tutoring System Authoring Tools: Harvesting the Current Crop and Planting the Seeds for the Future*. Paper presented at the Intelligent Tutoring Systems.
- Cohn, J., Olde, B., Bolton, A., Schmorrow, D. & Freeman, H. (2015). Adaptive and Generative Agents for Training Content Development. In K. W. Brawner (Ed.), *Design Recommendations for Intelligent Tutoring Systems: Authoring Tools (Volume 3)*. Army Research Laboratory.
- Goldberg, B., Brawner, K., Sotilare, R., Tarr, R., Billings, D. R. & Malone, N. (2012). *Use of Evidence-based Strategies to Enhance the Extensibility of Adaptive Tutoring Technologies*. Paper presented at the The Interservice/Industry Training, Simulation & Education Conference (I/ITSEC).
- 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.
- Lester, J., Mott, B., Rowe, J. & Taylor, R. (2015). Design Principles for Pedagogical Agent Authoring Tools In K. W. Brawner (Ed.), *Design Recommendations for Intelligent Tutoring Systems: Authoring Tools (Volume 3)*. Army Research Laboratory.
- MacLellan, C. J., Koedinger, K. R. & Matsuda, N. (2014). *Authoring Tutors with SimStudent: An Evaluation of Efficiency and Model Quality*. Paper presented at the Intelligent Tutoring Systems.

- Nye, B. D., Yang, M., Hays, P., Silva-Lugo, R., Cai, Z., Rahman, M. F., . . . Graesser, A. C. (2015). *Rapid, Form-Based Authoring of Natural Language Tutoring Trialogs*. Paper presented at the Generalized Intelligent Framework for Tutoring (GIFT) Users Symposium (GIFTSym2).
- Rus, V., Stefanescu, D., Niraula, N. & Graesser, A. C. (2014). *DeepTutor: towards macro-and micro-adaptive conversational intelligent tutoring at scale*. Paper presented at the Proceedings of the first ACM conference on Learning@ scale conference.
- Shaffer, D. W., Ruis, A. R. & Graesser, A. C. (2015). Authoring Networked Learner Models in Complex Domains. In K. W. Brawner (Ed.), *Design Recommendations for Intelligent Tutoring Systems: Authoring Tools (Volume 3)*. Army Research Laboratory.
- Sottilare, R., Graesser, A., Hu, X. & Goldberg, B. (2014). *Design recommendations for intelligent tutoring systems: Instructional Strategies (Volume 2)*. www.gifttutoring.org; U.S. Army Research Laboratory.
- Sottilare, R., Graesser, A., Hu, X. & Holden, H. (2013). *Design Recommendations for Intelligent Tutoring Systems: Learner Modeling (Volume 1)*. www.gifttutoring.org; U.S. Army Research Laboratory.
- Zapata-Rivera, D., Jackson, T. & Katz, I. R. (2015). Authoring Conversation-based Assessment Scenarios. In K. W. Brawner (Ed.), *Design Recommendations for Intelligent Tutoring Systems: Authoring Tools (Volume 3)*. Army Research Laboratory.

CHAPTER 12 Design Principles for Pedagogical Agent Authoring Tools

James Lester, Bradford Mott, Jonathan Rowe and Robert Taylor

Center for Educational Informatics, North Carolina State University

Introduction

Pedagogical agents hold great promise for enhancing the learning experience of students within intelligent tutoring systems (ITSs). There is mounting evidence that ITSs lead to improved student learning (Beal, Walles, Arroyo & Woolf, 2007; Schroeder, Adesope & Gilbert, 2013) and in some cases, have been found to be nearly as effective as one-on-one human tutoring (VanLehn, 2011). The timely and customized advice of ITSs may be further enhanced by the addition of pedagogical agents embodied as virtual characters that have the ability to motivate students while simultaneously providing complementary feedback through deictic gestures, motions, and utterances (Lester, Voerman, Towns & Callaway, 1999; Rus, D'Mello, Hu & Graesser, 2013). Advancing the case for employing pedagogical agents in tutoring systems is the increase in availability of game engines and graphics hardware capable of rendering lifelike virtual characters with significantly reduced development effort (Petridis et al., 2012).

Despite the potential for increased student engagement and the reduced cost of creating lifelike virtual characters, pedagogical agents have not yet achieved widespread adoption in computer-based learning environments. A formidable and well-known barrier to building and widely deploying a pedagogical agent is the complexity and expense associated with instilling the pedagogical agent with domain-specific knowledge and tutoring strategies (Murray, 2003; Woolf, 2009). Furthermore, an additional complication in creating an effective pedagogical agent is that the agent must present believable, lifelike behaviors such that students feel they are observing and interacting with “a sentient being with its own beliefs, desires, and personality” (Lester & Stone, 1997). Thus, a limiting factor in the widespread deployment of pedagogical agents is the significant effort and pedagogical agent expertise required to codify knowledge and behaviors from subject matter experts into the ITS.

An approach to solving this problem is improving the efficiency of codifying expert knowledge by creating pedagogical agent authoring tools that are tailored for subject matter experts rather than researchers. However, creating an effective authoring tool for subject matter experts poses two principal challenges. First, it must facilitate the creation of curricular content for the learning environment by subject matter experts who are not pedagogical agent experts and are often not software engineers. Second, it must support the creation or modification of pedagogical agent behaviors without exposing the complexity of the pedagogical agent itself to the subject matter expert. In practice, a majority of the design and programming effort expended on pedagogical agents is developing the agent and the learning environment itself. This often results in the authoring tool being treated as an afterthought, leaving little time or resources to design and develop authoring tools that are suitable for subject matter experts. Based on our experience developing a pedagogical agent authoring tool for educators, this chapter identifies promising authoring tool principles and features that could improve the authoring efficiency of subject matter experts. To conclude, we reason that the Generalized Intelligent Framework for Tutoring (GIFT) (Sottilare, Brawner, Goldberg & Holden, 2012) could be used to provide a high-quality implementation of these authoring tool design principles and, therefore, act as a force multiplier for creating new pedagogical agent-based tutoring systems that use GIFT.

Related Research

Creating authoring tools for building ITSs is receiving ever-increasing attention from the research community. With a goal of making ITS creation and authoring accessible to subject matter experts who are not computer scientists, progress is being made in researching approaches to create authoring tools (Susarla, Adcock, Van Eck, Moreno & Graesser, 2003; Jordan, Hall, Ringenberg, Cue & Rose, 2007) and automate aspects of pedagogical agents such as dialogue (André et al., 2000; Si, Marsella & Pynadath, 2005; Piwek, Hernault, Prendinger & Ishizuka, 2007) or nonverbal behaviors (Lhommet & Marsella, 2013).

Authoring tools for conversation-based learning environments have focused on assisting non-technical users in the creation of pedagogical agent dialogues. AutoTutor provides multi-agent conversational interactions to tutor students using the discourse patterns of a human tutor. AutoTutor has been used across multiple domains including computer literacy and physics (Graesser, Chipman, Haynes & Olney, 2005). To facilitate the application of AutoTutor to other domains, authoring tools have been developed to aid subject matter experts in creating dialogue-based tutors, such as the AutoTutor Script Authoring Tool (Susarla, Adcock, Van Eck, Moreno & Graesser, 2003) and AutoLearn (Preuss, Garc & Boullosa, 2010). Another example of an authoring tool for agent dialogue is TuTalk, which was created to support the rapid development of agent-based dialogue systems by non-programmers (Jordan, Hall, Ringenberg, Cue & Rose, 2007). This tool facilitates the authoring of domain knowledge and resources required by the dialogue agent in the form of artificial intelligence (AI) planning techniques that address high-level goals of the dialogue system. Similarly, an authoring tool has been created for the Tactical Language and Culture Training System (TLCTS) that allows subject matter experts to create pedagogical dialogue for a foreign language learning training system at reduced cost and time (Meron, Valente & Johnson, 2007).

Another approach to improving pedagogical agent authoring is to remove the need for authoring altogether through the use of automation. In particular, automating the creation of pedagogical agents' lifelike nonverbal behaviors eliminates a potentially significant amount of authoring effort. Cerebella is a system that monitors an agent's utterances (in both text and audio formats) and automatically generates lifelike nonverbal behaviors such as averting gaze, raising an eye brow, or slumping shoulders (Lhommet & Marsella, 2013). The automatically generated nonverbal behaviors inferred from the communicative intent and underlying mental state of the agent can be used as an additional channel of communication between the pedagogical agent and the student, increasing the agent's believability as well as students' engagement. The THESPIAN system reduces the effort to author pedagogical agents by facilitating the creation of interactive pedagogical dramas (Si, Marsella & Pynadath, 2005). In THESPIAN, the learner and the pedagogical agents interact with each other as characters within a story. THESPIAN accepts as input a set of scripts that it uses to automatically generate and adjust agents' goals to guide their behavior. Another example of automating pedagogical agent authoring tasks is to convey domain knowledge to the student through observations of simulated conversations and interactions between agents. The agent dialogue, character selection, and content rendering tasks would be automatically performed by the presentation system as described by André et al.(2000). In this approach, information is communicated by decomposing knowledge into atomic information units that are then conveyed to the student through verbal and nonverbal interactions between two or more agents.

Authoring of pedagogical agents can be accelerated by leveraging knowledge that has already been recorded in other forms such as Wikipedia pages, PowerPoint presentations, dialogue scripts, or PDFs. The Tools for Rapid Automated Development of Expert Models (TRADEM) project parses existing domain content and automatically generates dialogue, questions, and a script that represents the order of instruction based on the ordering of the original content (Robson, Ray & Cai, 2013). This system can be used to create a minimal dialogue-based tutoring system where a pedagogical agent can ask questions and

evaluate student answers related to the original content without requiring a subject matter expert to explicitly author the knowledge or assessments in the ITS (Brawner & Graesser, 2014). Text2Dialogue is another system that can use existing knowledge represented as text files to produce dialogue that is acted out by 3D virtual characters (Piwek, Hernault, Prendinger & Ishizuka, 2007). A significant difference between this approach and the previously described presentation system developed by André et al.(2000) is that Text2Dialogue can accept textual information as input without presentation goals being defined by a subject matter expert, which means that dialogue may be generated from existing text files without requiring annotation by a subject matter expert.

Even though the aforementioned research into implementing, augmenting, and eliminating the need for pedagogical agent authoring tools holds great promise, there is still an immediate need for effective and efficient tools that enable subject matter experts to codify knowledge and tutoring strategies as pedagogical agents without requiring the subject matter experts to possess or acquire programming or intelligent tutoring expertise.

Discussion

To address the immediate need for effective and efficient authoring tools, we present seven design principles that are grounded in software engineering practice and have the potential to significantly improve pedagogical agent authoring tools intended for subject matter experts. We illustrate our discussion with the COMPOSER authoring tool, which was developed for non-technical subject matter experts to author pedagogical agents. We describe our lessons-learned using COMPOSER to create a pedagogical agent for a widely deployed ITS for upper elementary science education.

Design Principles for Pedagogical Agent Authoring Tools

To make pedagogical agent-based learning environments more widely available, authoring tools must be designed and implemented that empower subject matter experts to quickly and efficiently populate the domain knowledge and tutoring strategies used by the pedagogical agent. To this end, creating usable and efficient authoring tools can be framed as a software engineering problem that may be addressed by general software design principles. The principles we advocate are well established in software engineering. Our contribution is discussing how to operationalize these principles in the context of authoring pedagogical agents. Since the design and implementation of authoring tools directly impacts the design and implementation of intelligent pedagogical agents (and vice versa), we recommend that the following pedagogical agent authoring tool design principles be considered at the beginning of a project, and leveraged in concert with the development of the learning environment, rather than leaving the tool development for the end of the project, where the tool will be constrained by an existing pedagogical agent implementation. In this section, we enumerate software design principles and features that should be considered for inclusion in a subject matter expert-centered pedagogical agent authoring tool.

Adopt a Familiar User Interface Paradigm

From a usability standpoint, the most important feature of an authoring tool is its user interface (UI). Ideally, a pedagogical agent authoring tool should present a UI that is familiar and intuitive for the type of subject matter expert who is intended to use it. Instead of requiring the subject matter expert to conform to unfamiliar ITS naming conventions and authoring workflow, the authoring tool should be modeled after software that the subject matter expert is already comfortable using. For example, if the intended user of the tool is a K-12 teacher, this type of user is likely very comfortable using Microsoft PowerPoint to create presentations to be shown in the classroom. Likewise, if the type of subject matter expert is a

computer scientist, this user will be comfortable writing code and using an integrated development environment (IDE), such as Eclipse. Of course, existing UIs and usage paradigms can (and should) be improved upon; however, instead of starting from scratch when designing an authoring tool, modeling upon an existing tool leverages decades of real-world usability and efficiency improvements.

Modeling a pedagogical agent authoring tool's UI after an existing authoring tool, such as Microsoft PowerPoint, does not imply that the tutoring knowledge constructs must be as simple as the content in a typical PowerPoint presentation. This would indeed be challenging since tutoring systems are likely to require authoring pedagogical strategies or annotating answer correctness, which are features that are not afforded by the PowerPoint UI. Instead, this principle implies that the authoring tool should model the existing tool by using similar naming conventions, presenting similar software features, and mimicking its workflow. For example, a pedagogy-oriented authoring tool might represent blocks of curriculum knowledge as "slides" in a PowerPoint-like authoring tool (Figure 1). Likewise, a slide might provide

The figure shows the COMPOSER authoring tool's user interface. On the left, a slide titled 'Look at the diagram below that shows the parts needed to make a flashlight work.' contains a circuit diagram with two batteries and a light bulb. A callout points to the batteries with the text 'Tags that represent skills associated with this knowledge (e.g., Next Generation Science Standards codes)'. Below the slide is a text input field asking 'What do you think the word energy means?' and a 'Finished' button. A callout points to this area with the text 'Curriculum knowledge represented as a slide'. At the bottom, a 'Pedagogical agent dialogue editor' window shows a 'Members:' pane with text inputs for scientific explanations and a 'Properties:' pane for a 'PadMate Action' dialog. A callout points to the 'Properties:' pane with the text 'Launches agent-specific authoring window for advanced users'. The right side of the interface is a vertical sidebar with sections for 'Common', 'Essay Entry', 'Input Entry', 'Layout', 'Tutor', and 'User Data', each containing various configuration options.

Figure 1. Slide-based authoring paradigm illustrated by the COMPOSER authoring tool user interface

static text or multimedia that is used to convey information to the student, as well as embedded assessments that are used to gauge student competencies. Slides could be associated with production rules for teaching the specific concepts and skills represented by the slide, while tags enable the pedagogical agent to associate students' performance with an overarching set of knowledge components. Without the subject matter expert explicitly authoring it, the pedagogical agent could use this metadata and the student model to determine the next slide to display to the student.

Include Standard Editing Features

Modeling a pedagogical agent authoring tool after a mature software package, such as Microsoft PowerPoint, suggests the implementation of several software features that are expected and relied upon by typical software users; however, these features are often nontrivial to implement and have profound effects on how data are represented, stored, and manipulated within the authoring tool, which is likely to affect how the data are represented in the ITS itself. For example, copy, cut, and paste features are expected by users to be available on any data type that can be authored in a tool. This feature may require deep or shallow copies of data models used to represent curriculum and pedagogical data while maintaining relationships between the data. Similarly, the undo and redo features enable users to experiment and quickly repair authoring mistakes. Undo and redo can drastically impact the design and implementation of the authoring tool itself and, therefore, should not be left as a feature to be added at the end of project when there is limited time to refactor data models or add revision tracking to the content being authored.

Support Author Collaboration

A pedagogical agent authoring tool should implement features that allow multiple subject matter experts to collaborate while authoring domain knowledge and pedagogical strategies. Collaboration has the potential to increase both the quality and quantity of content available to the ITS. Users have come to expect and rely upon collaboration features in other contexts. For example, at one extreme, multiple authors can use web browsers to simultaneously edit a single Google document, presentation, or spreadsheet. The authors can view each other's modifications and chat with one another while editing. Likewise, many content authoring tools enable change tracking to record which author made a change and when, or allow an author to comment on a piece of content without changing it in the form of a note. Implementing collaboration in an authoring tool will have significant impacts on the design of data models, the architecture of the application, and user authorization in regards to who is allowed to access which data. For example, storing domain knowledge and pedagogical strategies in a cloud-based server and implementing a web browser-based authoring tool would simplify implementation of collaboration features. Of course, this decision would need to be considered early in the design of the pedagogical agent and the authoring tool since it would impact the architecture and implementation of the entire system.

Facilitate Rapid Iteration and Testing

To facilitate refining the domain knowledge and pedagogical agent behaviors, the authoring tool should support a “rapid iteration” mode where small changes made in the authoring tool can be quickly seen and interacted with in the context of the ITS. In this mode, the subject matter expert can ideally interact with the pedagogical agent while editing content in real time or with only a minor delay. This feature allows the subject matter expert to quickly confirm that content is presented in a visually appealing manner in the learning environment and that the pedagogical agent behaves in a believable manner while the subject matter expert is modifying properties or settings that influence the pedagogical agent’s behavior. This feature could be implemented as a real-time connection to the ITS running as a separate application or the ITS could be embedded in the authoring tool to provide a what you see if what you get (WYSIWYG)

experience. In either situation, the data models would be required to support incremental dynamic updates and the ITS itself would have to respond to commands from the authoring tool such as navigating to specific domain content or modify the current state of the pedagogical agent depending on the types of edits the subject matter expert is making.

Accommodate Novice and Expert Authors

The pedagogical agent authoring tool should support editing methods that are specifically tailored to novice and expert users rather than presenting a one-size-fits-all UI. For example, a novice user is likely to be overwhelmed and discouraged by an authoring tool that exposes too many ITS-specific properties or settings. Conversely, an expert will be less efficient and will be frustrated by a UI that repeatedly walks through a series of basic steps. Therefore, for less frequently used authoring activities, or when authoring complex knowledge representations or pedagogical agent-specific behavior, the authoring tool should present a step-by-step wizard interface for novice users and a more direct authoring UI for expert users. For example, when authoring rules to evaluate the answer to an essay question, a wizard UI might ask the subject matter expert a series of questions that are used to generate a set of rules for evaluating the answer. On the other hand, an expert user would have the option of bypassing the wizard and authoring the rules directly. Interestingly, this design principle could be realized by embedding a pedagogical agent in the authoring tool itself to assist the subject matter expert in authoring content.

Automate Complex and Tedium Tasks

Some aspects of authoring domain knowledge or pedagogical agent behavior may be too complicated, labor intensive, or tedious for a subject matter expert to accomplish manually using a pedagogical agent authoring tool. In these situations, the authoring tool should provide automated mechanisms for generating curriculum content, pedagogical strategies, and pedagogical agent behaviors. This is where the automatic agent behavior generation techniques, as illustrated by Cerebella (Lhommet & Marsella, 2013), and automatic dialogue generation methods, as leveraged by THESPIAN (Si et al., 2005), can be used within the pedagogical agent authoring tool to reduce the authoring load for a subject matter expert.

Another approach to simplify the authoring of knowledge and pedagogical strategies is to assist the subject matter expert though the use of data mining techniques. Instead of authoring a pedagogical agent with strategies for every conceivable situation, authoring effort could be placed on the most common misconceptions or areas where students are showing weakness. In an educational data-mining study by Merceron and Yacef, student data from a web-based learning environment was mined to inform teachers about students who were at risk (Merceron & Yacef, 2005). Students were grouped into learner cohorts using clustering techniques to identify students who were having difficulties. In a similar way, an ITS could initially be deployed with curriculum content but a relatively primitive pedagogical agent. After collecting student answers, the data could be mined to identify common misconceptions or domain knowledge that may require additional scaffolding by the pedagogical agent. The authoring tool would flag sections of the domain knowledge or identify broader concepts that the subject matter expert should focus on improving. This would naturally lead to an iterative authoring process where the pedagogical agent continues to evolve by focusing effort on the issues most relevant to students who are using the ITS. Using this type of authoring assistance feature has the potential to dramatically reduce the amount of authoring effort, because the subject matter expert is not required to exhaustively predict and annotate all possible correct and incorrect answers. On the other hand, the initial iterations of the pedagogical agent are unlikely to be particularly effective since they will have limited ability to provide remediation to students who are having difficulty.

Avoid the Blank Page

Pedagogical agent authoring tools should assist the subject matter expert in getting started. Starting from a blank page using an unfamiliar tool can be a daunting task for any author of any skill level. This is particularly the case for someone who is authoring content for software as complex as a pedagogical agent-based ITSs. Therefore, authoring tools should provide templates and sample systems that can be used as starting points for authoring domain knowledge and pedagogical agent behaviors and dialogue. In addition, allowing subject matter experts to easily share their work with others has the potential to create a community that can evolve pedagogical agents by starting with another author's agent and building upon it rather than starting from scratch.

Importing existing knowledge that is already authored in the form of Microsoft Word documents, web pages, PowerPoint presentations, databases, or text files is a powerful feature for authoring tools to assist subject matter experts in quickly moving past the blank page. Taking it several steps further, automated systems such as TRADEM (Robson et al., 2013) and Text2Dialogue (Piwek et al., 2007) import existing knowledge and then automatically author agent dialogue, further reducing (or possibly eliminating) the pedagogical agent authoring load on the subject matter expert.

Lessons Learned from the LEONARDO Digital Science Notebook

For the past four years, our laboratory has been developing a digital science notebook for upper elementary science education, the LEONARDO CyberPad, which runs on the Apple iPad and within web browsers on Windows and Mac OS X computing platforms. LEONARDO integrates a pedagogical agent into a digital science notebook that enables students to graphically model science phenomena. With a focus on the physical and earth sciences, the LEONARDO PadMate, a 3D embodied pedagogical agent, supports students' learning with real-time, problem-solving advice. LEONARDO's curriculum is based on the Full Option Science System (Mangrubang, 2004). Throughout the inquiry process, students using the LEONARDO CyberPad are invited to answer multiple-choice questions, write answers to constructed response questions, and create symbolic sketches of different types, including electrical circuits. To date, LEONARDO has been implemented in over 70 elementary school classrooms across the United States.

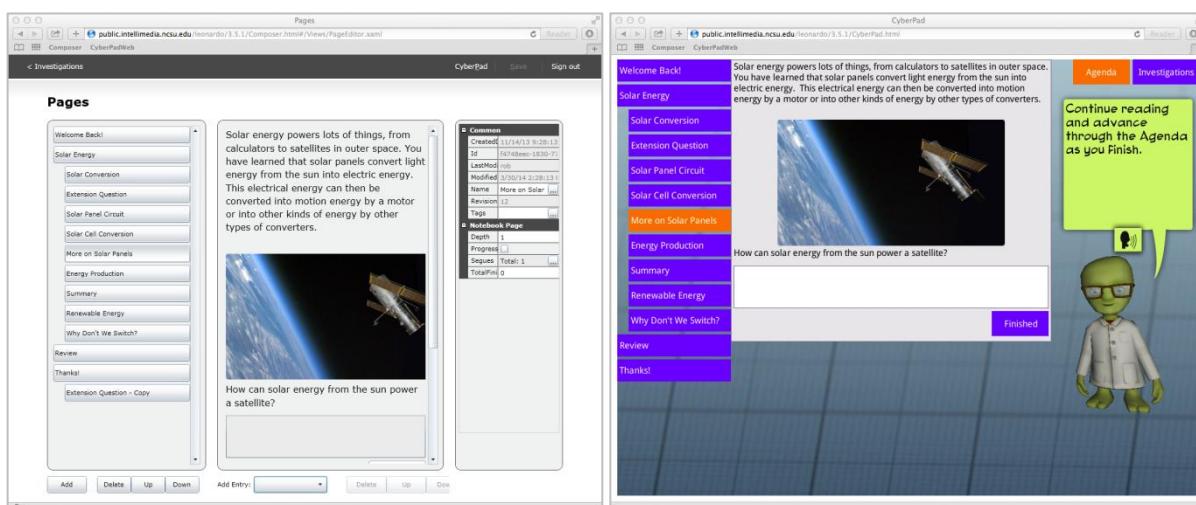


Figure 2. The COMPOSER tool (left) and CyberPad (right) in rapid iteration editing mode

LEONARDO consists of three major components: the CyberPad digital science notebook, the COMPOSER authoring tool, and a cloud-based server. Fourth and fifth grade elementary students learn about magnetism, circuits, and electricity using the CyberPad software (Figure 2, right). Subject matter experts use the COMPOSER (Figure 2, left) authoring tool to create curriculum content displayed in the digital science notebook, as well as rules, dialogue, and gestures that drive the pedagogical agent, which is embodied as a green alien within the CyberPad UI. The cloud-based server is used to store all curriculum knowledge, tutoring rules, and student data. During the design and development of the COMPOSER authoring tool, many of the principles for creating subject matter expert-centered authoring tools were identified as necessary features or enhancements that would improve the productivity of subject matter experts.

The LEONARDO project did not originally include the COMPOSER authoring tool in its work plan. The first year of the project was spent designing and implementing a prototype of the CyberPad application to field test with fourth and fifth grade students to assess the practicality and ergonomics of using iPads in elementary school classrooms. During the first year, subject matter experts, who were science education faculty and graduate students, used Microsoft Word to author all of the curriculum content and pedagogical agent dialogue. The development team, who were computer science research staff and graduate students, manually copied the text from the Microsoft Word document into multiple extensible markup language (XML) documents. The XML documents were then embedded in the CyberPad iPad application as fixed resources that were then installed on iPads. The agent dialogue and rules were coded directly into the CyberPad's source code. Needless to say, this approach to content authoring was highly inefficient. It was labor intensive and error prone due to the need to repeatedly copy data by hand. In addition, pedagogical agent rules, dialogue, and gestures were tightly coupled with the contents of the XML documents making the entire system highly susceptible to syntax and typographical errors.

This initial approach to pedagogical agent authoring for the LEONARDO project had several significant drawbacks: First, the subject matter experts did not have a means to visualize what the curriculum content and pedagogical agent dialogue would look like when it was displayed in the CyberPad UI as they were authoring content in Microsoft Word. Second, it was extremely slow to make small changes to the content since it required a development team member to be available to (a) make the change in XML, (b) rebuild the application, and (c) redeploy the CyberPad application to the iPads. Third, this dependency resulted in frustration for the subject matter experts and development team members. As a result, the curriculum content lacked polish, which is typically achieved by making many small changes after the original content is created. Since making small changes was highly inefficient, these changes were not made due to lack of resources and time. Using this approach to authoring content, 1 hour of instruction required more than the estimated 300 hours of development time often cited for ITS authoring (Tom Murray, 2003).

Based on this initial authoring experience and future plans to more than triple the amount of curricular content and pedagogical agent dialogue, it became imperative to design and implement the COMPOSER authoring tool in the second year of the project. We started requirements gathering by identifying the types of subject matter experts who would use the tool in the future: elementary school teachers, education graduate students, and education faculty. We then proceeded to design COMPOSER's UI by reviewing authoring tools from other areas that our subject matter experts were comfortable using. This included applications such as Microsoft PowerPoint, Google documents, and Edmodo. In the new system, curriculum content, agent dialogue, and rules would be stored in a cloud-based server where it could be directly accessed by both the COMPOSER tool and the CyberPad application. This approach formed the basis for the authoring tool principles and features proposed in this chapter.

The COMPOSER authoring tool improved the authoring workflow for the LEONARDO project in years two and three by decoupling content authors from the development team. Subject matter experts were

empowered to refine curriculum content and pedagogical agent behavior independently of the development team. In addition, a familiar workflow and editing feature set further improved the efficiency of subject matter experts. However, since authoring wasn't considered in the initial design, these improvements did come at a development cost of refactoring data models, logic, and storage to make it possible to edit and track small discrete parts of the curriculum.

Recommendations and Future Research

Widespread development and deployment of pedagogical agents in ITSs depends on efficient transfer of domain knowledge and pedagogical agent dialogue, strategies, and behaviors from subject matter experts to the tutoring system. Authoring tools hold great promise to facilitate knowledge engineering. However, it should be emphasized that authoring tools should be tailored to the subject matter expert using features and workflows that have been proven effective by authoring software from non-ITS domains.

In future work, it will be important to investigate the addition of automation features to assist in the authoring of pedagogical behaviors and tutoring strategies. In the near term, leveraging educational data mining techniques to discover prevalent student behaviors, as well as misconceptions, from ITS datasets could further enhance ITS authoring tools and identify parts of curricula that require additional scaffolding. Future work should strive to immediately incorporate decades of software engineering knowledge in the design and implementation of novice and expert UIs to simplify authoring complex knowledge and underlying ITS mechanisms.

The design principles for pedagogical agent authoring tools presented in this chapter are not specific to any given tutoring system. Since these authoring tool principles are broadly applicable across ITSs, GIFT affords a unique opportunity to act as a authoring tool platform where many of these authoring tool design principles could be implemented once and used by many tutoring systems. This would allow a single high-quality authoring tool implementation to be established that could then be shared across multiple tutoring systems, thereby reducing redundant authoring tool design and development effort across multiple projects while simultaneously raising the quality of the authoring tools based on GIFT. It follows that this approach has the potential to produce higher-quality pedagogical agent-based learning environments more quickly and at reduced cost.

References

- Beal, C. R., Walles, R., Arroyo, I. & Woolf, B. P. (2007). On-line tutoring for math achievement testing: A controlled evaluation. *Journal of Interactive Online Learning*, 6(1), 43–55.
- Brawner, K. & Graesser, A. (2014). Natural Language, Discourse, and Conversational Dialogues within Intelligent Tutoring Systems: A Review. In R. Sottilare, A. Graesser, X. Hu & B. Goldberg (Eds.), *Design Recommendations for Intelligent Tutoring Systems* (pp. 189–204).
- 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.
- Jordan, P. W., Hall, B., Ringenberg, M., Cue, Y. & Rose, C. (2007). Tools for Authoring a Dialogue Agent that Participates in Learning Studies. In R. Luckin, K. R. Koedinger & J. Greer (Eds.), *Artificial Intelligence in Education: Building Technology Rich Learning Contexts That Work* (pp. 43–50). IOS Press.
- Lester, J. C. & Stone, B. A. (1997). Increasing believability in animated pedagogical agents. In *AGENTS '97 Proceedings of the First International Conference on Autonomous Agents* (pp. 16–21).
- Lester, J. C., Voerman, J. L., Towns, S. G. & Callaway, C. B. (1999). Deictic Believability: Coordinated Gesture, Locomotion, and Speech in Lifelike Pedagogical Agents. *Applied Artificial Intelligence*, 13(4-5), 383–414.
- Lhommet, M. & Marsella, S. C. (2013). Gesture with meaning. In *Intelligent Virtual Agents* (pp. 303–312). Springer Berlin Heidelberg.

- Mangrubang, F. R. (2004). Preparing elementary education majors to teach science using an inquiry-based approach: The Full Option Science System. *American Annals of the Deaf*, 149(3), 290–303.
- Merceron, A. & Yacef, K. (2005). Educational Data Mining: a Case Study. In *AIED* (pp. 467–474).
- Meron, J., Valente, A. & Johnson, W. L. (2007). Improving the authoring of foreign language interactive lessons in the tactical language training system. In *Speech and Language Technology in Education (SLaTE2007)* (pp. 33–36).
- Murray, T. (2003). An Overview of Intelligent Tutoring System Authoring Tools: Updated analysis of the state of the art. In T. Murray, S. Blessing & S. Ainsworth (Eds.), *Authoring Tools for Advanced Technology Learning Environments* (pp. 493–546). Springer Netherlands.
- Petridis, P., Dunwell, I., Panzoli, D., Arnab, S., Protopsaltis, A., Hendrix, M. & de Freitas, S. (2012). Game Engines Selection Framework for High-Fidelity Serious Applications. *International Journal of Interactive Worlds*, 2012, 1–19.
- Piwek, P., Hernault, H., Prendinger, H. & Ishizuka, M. (2007). T2D: Generating Dialogues Between Virtual Agents Automatically from Text. In *Intelligent Virtual Agents* (pp. 161–174). Springer Berlin Heidelberg.
- Preuss, S., Garc, D. & Boullosa, J. (2010). AutoLearn's Authoring Tool: A Piece of Cake for Teachers. In *Proceedings of the NAACL HLT 2010 Fifth Workshop on Innovative Use of NLP for Building Educational Applications* (pp. 19–27). Association for Computational Linguistics.
- Robson, R., Ray, F. & Cai, Z. (2013). *Transforming Content into Dialogue-based Intelligent Tutors*. Paper presented at The Interservice/Industry Training, Simulation & Education Conference (I/ITSEC), Orlando, FL.
- Rus, V., D'Mello, S. K., Hu, X. & Graesser, A. C. (2013). Recent advances in intelligent tutoring systems with conversational dialogue. *AI Magazine*, 34(3), 42–54.
- Schroeder, N. L., Adesope, O. O. & Gilbert, R. B. (2013). How Effective are Pedagogical Agents for Learning? A Meta-Analytic Review. *Journal of Educational Computing Research*, 49(1), 1–39.
- Si, M., Marsella, S. C. & Pynadath, D. V. (2005). THESPIAN: An Architecture for Interactive Pedagogical Drama. In *AIED* (pp. 595–602).
- Sottilare, R. A., Brawner, K. W., Goldberg, B. S. & Holden, H. K. (2012). A modular framework to support the authoring and assessment of adaptive computer-based tutoring systems (CBTS). In *Proceedings of the Interservice/Industry Training, Simulation, and Education Conference*.
- Susarla, S., Adcock, A., Van Eck, R., Moreno, K. & Graesser, A. C. (2003). Development and evaluation of a lesson authoring tool for AutoTutor. In *AIED2003 supplemental proceedings* (pp. 378–387).
- VanLehn, K. (2011). The Relative Effectiveness of Human Tutoring, Intelligent Tutoring Systems, and Other Tutoring Systems. *Educational Psychologist*, 46(4), 197–221.
- Woolf, B. P. (2009). *Building Intelligent Interactive Tutors: Student-centered strategies for revolutionizing e-learning*. San Francisco, CA: Morgan Kaufmann.

Chapter 13 Adaptive and Generative Agents for Training Content Development

Joseph Cohn¹, Brent Olde², Ami Bolton², Dylan Schmorrow³ and Hannah Freeman⁴

¹Office of the Secretary of Defense; ²Office of Naval Research; ³SOARTech; ⁴Strategic Analysis, Inc.

Introduction

In this chapter, we provide a vision, based on our combined 40+ years of developing training and education technologies, for the next stage in training system content development. While this vision is informed by specific Defense needs gaps and requirements, it is developed in a manner that makes it broadly applicable to a much wider range of training needs. Our vision treats training content as the foundation upon which effective training systems are developed and addresses a deep limitation in how content is developed today. We believe that the limits on content development are a key factor in preventing the development of large-scale adaptive training systems. Simply put, regardless of how quickly and accurately a training system can diagnose a student's performance deficits, the effectiveness with which the training system can develop and enact remedial strategies relies wholly on the depth and scope of the content from which specific instances of these strategies can be applied.

We envision an automated capability to generate new, context-appropriate training content with limited human supervision, based on integrating training system authoring tools with expert system technologies and using unbounded data sets. A critical enabler of this approach is the ability to deliver training through student interactions with one or more *agents* within a simulated training environment. These agents will accomplish three important training goals. First, by interacting with students in real time, these agents' behaviors will provide an experiential type of learning, arguably one of the strongest types of learning strategy. Second, by virtue of interacting with students, these agents will have the ability to assess student performance against a set of training goals and objectives, and identify specific training deficits. Lastly, using knowledge about the student's current state and their desired end-state, these agents will have the basic information necessary to *generate* new and appropriate behaviors—an entirely new form of adaptive content that is not solely dependent on instructor forethought or scripting.

This new capability will replace hand-coded rule sets, automatically generating new and appropriate agent behaviors from one or more data sources including data captured during live exercises; data captured from experts operating their systems within a simulated environment; or data provided in a script-like format. On the basis of one or more of these initial data sets, it should then be possible to reproduce the behaviors, model them for more general uses, and extend those models to provide new behaviors in a training environment. This approach will require integrating cognitive modeling approaches with machine learning techniques to generate tactically authentic behaviors. Cognitive models provide a means of formally representing the underlying behaviors of interest. Machine learning techniques provide a wide range of inductive approaches to generalize modeled behaviors to new missions and contexts. Training objectives, doctrine and tactics, techniques and procedures (TTPs) bound the initial cognitive models and subsequent machine learning generalization to ensure that new behaviors are tactically authentic and also responsive to training needs. The resultant behaviors can then be validated as part of a new training scenario. The need for new approaches for delivering effective training is clear. Using live assets for training exercises is becoming prohibitively costly, both due to reduced access to live training ranges (Mehta, 2014) and range space for conducting live training exercises continues to be reduced (or eliminated; Oslon, 2014). Increased operational tempos, and reduced manpower across the Services, further limits access to training. Intelligent tutoring systems (ITSs) are meant to address these challenges

by providing tailored, adaptive training “on demand” but these approaches often struggle to show a significant return on investment (O’Connor & Cohn 2010). On the one hand, the very best ITSs, which mimic the very best student-instructor interactions, are still too costly to develop for large-scale use (Cohn & Fletcher, 2010). On the other hand, more affordable ITSs offer only pre-scripted training or minimally adaptive training, which while significantly lower in cost, is also less effective (Woolf, 2009).

Against this backdrop, new combat platforms, like unmanned systems and cyber combat systems, are being procured to address a new set of threats and challenges to our Nation’s security. These platforms will require training that is more focused on cognitive skills sets like problem solving, decision making, multi-tasking, task switching, and mission management, rather than on physical skill sets. This training is best delivered through interactive and adaptive approaches rather than less personal “one size fits all” classroom approaches (Vogel-Wolcott, 2013). O’Connor & Cohn (2010) and Cohn & Fletcher (2010) suggest that if ITSs are the indicated solution, then key drivers in making this type of training affordable lie in reducing the amount of effort needed to develop training content, while advancing the level of training system adaptability.

State of the Art

Adaptive, generative, and modular agents provide a key tool for enabling ITSs, by providing a new approach for developing and delivering content. In our view, content is the hub through which the spokes of any training system must be connected. To that end, we explore not only current research in content development, but, also, current research in other elements of adaptive training systems. Figure 1 provides one representation of the various elements necessary for developing adaptive training tools, based in part on Conati (1997), Woolf (2009) and Pardos et al.(2013), with details discussed below.

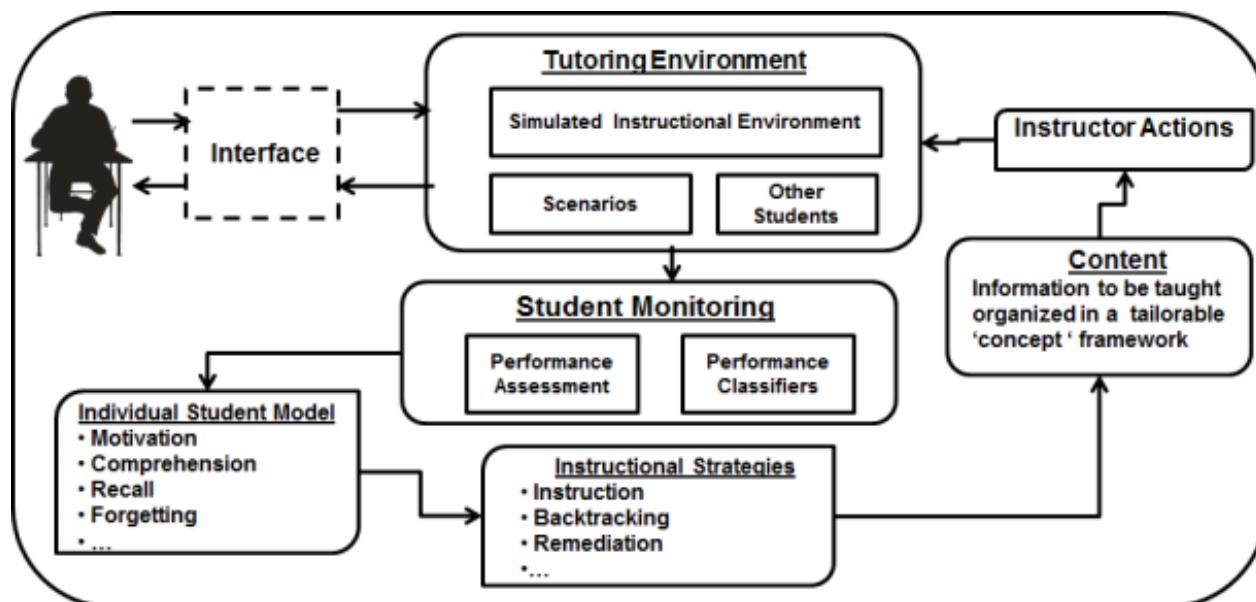


Figure 1: Elements that are critical to delivering effective instruction, mapped onto science and technology efforts, indicated in “()”: understanding each student’s overall learning needs (Individual Student Model, Student Monitoring), identifying specific approaches for addressing any learning gaps and building instructional modules (Instructional Strategies and Instructor Actions), that deliver content using these approaches (Content) through some hardware or software connection (Tutoring Environment, Interface).

Individual Student Models and Student Monitoring

Adaptive tutoring systems are meant to modify their instructional content and delivery to each individual student's learning needs (Jeremic et al.2012). This is best accomplished through a student model (Sison & Shimura, 1998), which integrates into an executable representation as much information about the individual student as can be reasonably and meaningfully captured, to provide to the tutoring system a picture of the student's current "state." Yet, despite the strong development of numerous, different, computational approaches to modeling student state, like Bayesian knowledge tracing (Corbett & Anderson, 1995) and performance factors analysis (Pavlik & Koedinger, 2009), it is becoming increasingly clear that current approaches may be reaching their upper limits accurately representing individual student state (Pardos, et al.2013). A critical reason for this may be that information about the student is often "latent" (Pardo et al.2013) or intangible, like meta-cognition, motivation, and affect (Desmarais & Baker, 2012).

Instructional Strategies and Instructor Actions

While representing a student's current learning needs and predicting their future ones is necessary for effective adaptive training (Clemente, Ramírez & De Antonio, 2011), it is not sufficient. An adaptive training system, like an expert instructor, must be able to tailor and deliver instruction in a way best suited to match these learning needs. This includes identifying the different types of strategies that enable effective instruction (see, for example, Lunenburg & Irby, 2011); establishing a framework for selectively applying these strategies to different student styles, as well as different student levels of expertise (e.g., Koedinger, Corbett & Perfetti (2012)'s "Knowledge–Learning–Instruction Framework,"); and developing a capability to computationally represent this framework in a way that can be integrated into an adaptive training system.

Tightly linked to instructional strategies is the method by which these strategies are delivered—the instructor actions that will impart information using one or more strategies. Simply requiring a training system to deliver reinforcement says nothing about *how* that reinforcement should be delivered or the form that such reinforcement should take. At their most fundamental, these actions should lead to learning, "...the process by which long-lasting changes occur in behavioral potential as a result of experience..." (Anderson, 2000). Learning, in turn, is enabled by activating the short- and long-term memory systems (Anderson, 2000). Consequently, the delivery of these strategies should be done in a way that durably establishes these memories in a way that also eases their retrieval. Recent work by Rohrer & Pashler (2014), Pashler et al. (2007) and Karpicke & Roediger (2008) indicates that "enforced retrieval" of information through a blend of studying, rehearsal, and testing can increase the ease with which information is stored, maintained, and retrieved.

Content

The cost of developing content is a major challenge to building effective ITSs. One reason for the high cost of content development is that it is expensive to create the corpus of knowledge that will inform content. While there are some advances being made on this front, such as Robinson et al.'s (2012) simulation based knowledge elicitation approach to elicit and model expert behaviors, this approach may only shift the cost away from content development to environment development. A second reason for the high cost of content development is that as more content is required for more complex adaptive systems, the framework (ontology) into which this content is embedded may need to expand pseudo-exponentially, with corresponding cost (Simperl & Mochol, 2006). Some interesting and new approaches for developing content include crowdsourcing (Koedinger, McLaughlin & Stamper, 2012; Weld et al.2012) and "big data" collection (Arroyo & Woolf, 2005) approaches, and the development of new types of knowledge

structures (Boyce & Pahl, 2007; Koenig, Lee, Iseli & Wainess, 2009) and associated techniques to represent these data sets (Pardos & Heffernan, 2010).

Tutoring Environment and Interface

There are many examples of “training systems,” which, lacking the elements indicated in Figure 1, are little more than practice platforms (Vogel-Walcutt, 2013). As Woolf (2009) suggests, training must be delivered in an authentic and relevant fashion to be effective. This means that not only must the interface to the system be “realistic” and transparent and the environment must be engaging, but the content must also be delivered in a motivating and stimulating fashion. As a result, the veridicality of the training may be significantly enhanced, leading to better, positive, learning transfer rates (Grossman & Salas, 2011).

Discussion

Our vision for adaptive training systems hinges on developing training content from as wide a range of sources as possible, making this content adaptive to student needs in real time, and embedding this content in agents that can deliver this training.

General Approach

The steps necessary to achieving this vision include the following (Figure 2):

- Develop the knowledge structures (ontologies) that will be used to capture source data.
- Define boundaries of behavior patterns that are of interest. This includes identifying what kind of activities to look for in real entity behaviors.
- Find the behavior patterns of interest using the boundary definitions.
- Develop representative cognitive models from the behavior data.
- Apply doctrine training goals and objectives to define and constrain agent behaviors.
- Use machine learning techniques to generate novel, doctrinally accurate, agents.

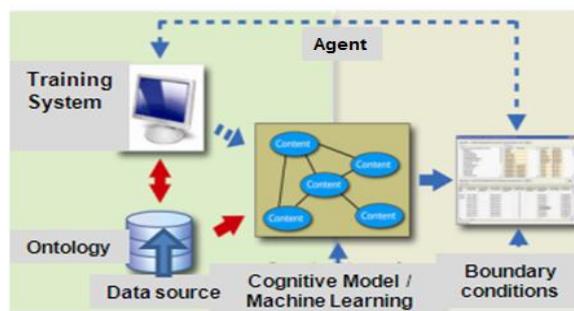


Figure 2: General approach for developing adaptive and generative agents. Data are placed into an ontology (left side). Boundary conditions are identified, based on doctrine or other sources (right side). The ontology and the boundary conditions are merged using cognitive models, and machine learning techniques evolve and adapt the represented behaviors to guide the agent (center) which is then integrated into the training system.

Source Office of Naval Research Fact Sheet Unmanned Aerial Systems Interface, Selection & Training Technologies Dynamic Adaptive & Modular entities for UAS (DyAdeM)

There are still challenges associated with applying these steps to specific training needs. As instructional system developers look to use more and varied data sources, fundamentally new types of ontologies will need to be developed to accommodate these data, which will certainly have a wide range of spatial and temporal fidelity. Early efforts to build these blended types of ontologies have been used with some success in the development of cognitive based control systems for autonomous systems (Stacy et al., 2010) and are now being expanded to include much larger and more varied types of data, including blending neural, behavioral, and machine-based sources (Cohn et al., 2015). Identifying boundaries for behavior patterns and discovering behavior patterns of interest is in many ways a big data analytics challenge, focusing on identifying an often minor signal against a backdrop of seemingly random “noise.” Modeling and pattern recognition approaches developed in other contexts, such as those used in the Office of the Secretary of Defenses’ Human Social Cultural Behavior Modeling Program (Boiney & Foster, 2013), could provide a foundation from which to build these techniques. An equally challenging problem is developing affordable methods for building executable representations (cognitive models) from these data to generate in real time novel, contextually appropriate, and doctrinally accurate agent behaviors to drive instruction. Today, building these behaviors requires significant time investment by scenario authors (Koedinger, et al. 2004). In the future, it will be critical to generate these models autonomously from the source data.

Example Application

Unmanned aerial system (UAS) training represents a new and complex domain that will strongly leverage modeling and simulation (M&S) solutions to develop embedded and emulated training environments, and in which agent-delivered content will play a key role for delivering training. Because the kinds of tasks in operating a UAS involve observing, tracking, and identifying many different types of entities (e.g., blue, red, and white forces), ITS training for UAS operators requires the integration of hundreds, if not thousands, of simulated entities into the overall training scenario. Currently, developing these entities requires significant time and effort, and results in entities whose behaviors are strictly guided, scripted, and limited based on pre-determined rules that define the entities’ behaviors over the course of the training scenario. The net result is entities whose behaviors are not realistic, leading to reduced training effectiveness, yet at the same time require significant effort to create, leading to prohibitively high authoring costs.

Applying the process described in Figure 2 to this challenge allows us to automate the development of new behaviors to drive a range of different types of simulated entities, providing an alternative, and potentially more effective and less costly, solution. The process begins with automating the recognition of live entity behaviors, captured from various UAS sensor data streams, and transforming those data into digital representations (Figure 3a). This requires ontologies that can capture both discrete and continuous data, across representations that can accommodate data with both high and low spatial and temporal resolution. This provides the foundation from which to model and generate behaviors to drive simulated entities. Next, the transformed data are bounded by user-specified parameters to create behavior envelopes, which represent goals and associated constraints. This sets the conditions for developing rules for generating new behaviors that are related to those captured from the live entity. During a training exercise, student performance is monitored to detect when goals are either archived or potentially not achieved, and when constraints are close to being violated. When these conditions are met, machine learning algorithms are applied to generate new behaviors (Figure 3b).

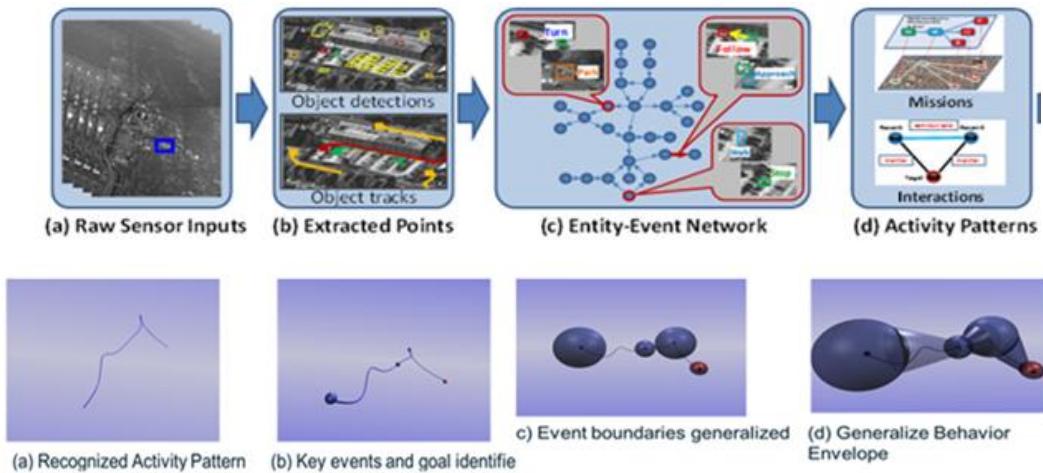


Figure 3: (a) Automated activity recognition identifies behaviors of live entities (e.g., aircraft, vehicles, people) to model using pattern recognition techniques applied to real sensor data received from UASs. (b) Generalized behavior envelopes are then developed from these patterns to provide rules for generating related behaviors. These rules are applied in response to student performance to deliver adaptive and generative agent behaviors. Courtesy of Aptima Inc. and SOARTech

Recommendations and Future Research

Extending the Approach

The approach for developing adaptive and generative agents overlays nicely on the different elements that comprise ITSs (Figure 1). The student models provide one of the functions that drive the agents to seek new behaviors. Student performance is monitored and assessed, and the outcome of this assessment provides the basis for either capturing new data or evolving current data sets into new behaviors that can, in turn, help remediate the student. At the same time, these agents would be able to build, through continuous interaction with each trainee, dynamic and highly individualized models that could capture “missing” information, such as latent (Pardos 2013) or affective (Desmarais & Baker, 2012) behaviors. How this information could be elicited remains to be determined, but one possible solution may lie in recent advances in the development of classifiers for inferring cognition from brain activity. In these efforts, individuals are shown a wide array of objects, of different categories, while simultaneously having their brain activity captured through non-invasive techniques. Using machine learning routines, a classifier can be built that can then scan brain activity when subjects view a new object and, with some degree of accuracy, predict what sort of object an individual is looking at (Mitchell, et al., 2004). Importantly, these classifiers appear to be transferrable to new categories of objects as well as to new groups of individuals, while maintaining reasonable levels of predictive accuracy (Shinkareva et al., 2008). In a similar manner, it might be possible to develop generalizable approaches to train classifiers to detect certain kinds of latent and affective variables. Alternatively, it may be possible to leverage and adapt machine learning approaches pioneered by the affective computing community (Picard, 1997), which allow computer systems to adapt their actions to the affective state of the user, inferred through facial feature recognition technologies. In both instances, a major leap that must be made is to move away from using physiological or physical based data (brain data or facial expression data) and focus on behaviors detected only through the student’s interface with the training system.

The resultant, data could support the development of models that would, in turn, guide the development of content specific instructional strategies to address learning deficits. At the other end of the spectrum,

the behaviors that could be driven through this approach provide new opportunities to realize a range of actions that the ITS can take to deliver instruction. Lunenburg & Irby (2011) identify a set of effective strategies, like Set Induction, Stimulus Variation, Reinforcement, and Questioning. Precisely how these strategies could be delivered using this approach also remains to be determined. Lastly, the current approach is being developed for a specific application, UAS instruction, in which data naturally are provided in digital format. Extending this approach to other domains in which the data are not inherently digital, like math or science instruction, will require new approaches for capturing and eliciting data from expert instructors.

Impact to the Generalized Intelligent Framework for Tutoring (GIFT)

GIFT already provides a strong foundation into which this approach may be integrated, with modifications potentially required for only a few modules. The GIFT sensor module offers a way for new data to be captured, although it would need to be extended to include large-scale ontologies and data from non-traditional sensor sources. The GIFT learner module is analogous to the student modeling and student monitoring elements (Figure 1), and may require only minor modifications to support the approach proposed here, allowing it to boot-strap from the output of the agents. The GIFT pedagogical module would similarly need to be modified to allow for instruction to be delivered via agents, as discussed above.

References

- Anderson, J. (2000) Learning and Memory: An integrated approach. Wiley
- Baker, R. S. J. (2007). Modeling and understanding students' off-task behavior in intelligent tutoring systems. In: M. B. Rosson and D. J. Gilmore (eds.): Proceedings of the 2007 Conference on Human Factors in Computing Systems, CHI 2007, San Jose, California, USA, April 28 - May 3, 2007. pp. 1059{1068.
- Boiney, J. & Foster, D. (2013). Progress and Promise: Research and engineering for Human Social cultural Behavior Capability in the U.S. Department of Defense. Accessed on 09 March 2015 from <http://www.mitre.org/publications/technical-papers/progress-and-promise-research-and-engineering-for-human-sociocultural-behavior-capability-in-the-us-department-of-defense>
- Boyce, S. & Pahl, C. (2007). Developing domain ontologies for course content. Educational Technology & Society, 10 (3), 275- 288.
- Clemente, J., Ramírez, J., & De Antonio, A. (2011). A proposal for student modeling based on ontologies and diagnosis rules. Expert Systems with Applications, 38(7):8066-8078.
- Corbett, A.T., Anderson, J.R., (1995). Knowledge Tracing: Modeling the Acquisition of Procedural Knowledge. User Modeling and User-Adapted Interaction, 4, 253-278 to Performance Factors Analysis
- Cohn, J.V. & Fletcher, D.F. (2010). What is a pound of training worth? Proceedings of the 31st Interservice/Industry Training, Simulation and Education Conference, Orlando, FL.
- Cohn, J.V., Stacy, W., Geyer, A., Squire, P. & O'Neill, E. (2015) Improving Human System Interactions Through a Neural Cognitive Architecture: A shared context approach (In preparation for submission to Theoretical Issues in Ergonomic Sciences)
- Desmarais, M.C & Baker, R.S.J.D. (2012). A Review of Recent Advances in Learner and Skill Modeling in Intelligent Learning Environments. User Model User-Adapt International, 22:9-38.
- Grossman, R. & Salas, E. (2011) The transfer of training: what really matters. International Journal of Training and Development 15:2 1468-2419.
- Karpicke, J.D.& Roediger, H.L.,III (2008): The critical importance of retrieval for learning. Science 15, 966–968.
- Koedinger, K. R., Aleven, V., Heffernan, N., McLaren, B. & Hockenberry, M. (2004). Opening the door to non-programmers: Authoring intelligent tutor behavior by demonstration. In J.C. Lester, R.M. Vicari & F. Parguacu (Eds.) Proceedings of the 7th International Conference on Intelligent Tutoring Systems, 162-174. Berlin: Springer-Verlag.
- Koedinger, K.R.,Corbett, A.T.,& Perfetti,C.(2012).The knowledge-learning-instruction framework: Bridging the science-practice chasm to enhance robust student learning. Cognitive Science, 36, 757–798.

- Koedinger, K. R.; McLaughlin, E. A. & Stamper, J. C. (2012). Automated Student Model Improvement. International Educational Data Mining Society, Paper presented at the International Conference on Educational Data Mining (EDM) (5th, Chania, Greece, Jun 19-21, 2012).
- Koenig, A. D., Lee, J. J., Iseli, M. R. & Wainess, R. A. (2009). A conceptual framework for assessing performance in games and simulations. Proceedings of the Interservice/Industry Training, Simulation and Education Conference, Orlando, FL.
- Lunenburg, F. C. & Irby, B. J. (2011). Instructional Strategies to Facilitate Learning. *International Journal of Educational Leadership Preparation*, 6(4), n4.
- Mehta A. (2014) Under Budget Pressure US Air Force Looks to Live Virtual Constructive Training. Retrieved from <http://www.defensenews.com/article/20140520/TRAINING/305200048/Under-Budget-Pressure-US-Air-Force-Looks-LVC-Training> 18 Nov 2014
- Mitchell, T. M., Hutchinson, R., Niculescu, R.S. Pereira, F. Wang, X., Just, M. & Newman, S. (2004). Learning to Decode Cognitive States from Brain Images. *Machine Learning*, 57, 145–175.
- O'Connor, P.E. and Cohn, J.V. (Eds.) (2010). Human Performance Enhancement in High-Risk Environments. Santa Barbara, CA: Praeger Security International.
- Olson, W. (2014). With deadline looming, agreement uncertain on Hawaii live-fire range Retrieved from <http://www.stripes.com/news/with-deadline-looming-agreement-uncertain-on-hawaii-live-fire-range-1.284813> 18 Nov 2014
- Pardos, Z. A., Heffernan, N. T. (2010) Modeling Individualization in a Bayesian Networks Implementation of Knowledge Tracing. In Proceedings of the 18th International Conference on User Modeling, Adaptation and Personalization. pp. 255-266. Big Island, Hawaii.
- Pashler,H. et al. (2007) Enhancing learning and retarding forgetting: choices and consequences. *Psychonom.Bull.Rev.* 14, 187–193
- Pavlik, P.I., Cen, H., Koedinger, K.R., 2009a. Learning Factors Transfer Analysis: Using Learning Curve Analysis to Automatically Generate Domain Models. In: Proceedings of the 2nd International Conference on Educational Data Mining, 121-130
- Robinson, S., Lee, E.P.K. & Edwards, J.E. (2012). Simulation based knowledge elicitation: Effect of visual representation and model parameters. *Expert Systems with Applications*, 39(9): 8479-8489
- Rohrer, D. & Pashler, H. (2010). Recent research on human learning challenges conventional instructional strategies *Educational Researcher*, 39(5) pp. 406-412.
- Picard, R. W. (1997) Affective Computing, MIT Press, 0-262-16170-2, Cambridge, MA, USA.
- Simperl, E.P.B & Mochol, M. (2006). Cost Estimation for Ontology Development In: Witold Abramowicz (ed.), *Business Information Systems*, Proceedings of BIS 2006, Poznań, Poland Retrieved from <http://page.mi.fu-berlin.de/mochol/papers/BIS06.pdf> 10 Nov 2014.
- Shinkareva, S. V., Mason, R. A., Malave, V. L., Wang, W., Mitchell, T. M. & Just, M. A. (2008). Using fMRI brain activation to identify cognitive states associated with perception of tools and dwellings. *PLoS ONE*, 3, e1394
- Stacy E.W., Cohn J.V., Geyer A., Wheeler T.A. (2010) Cognition-based control system for autonomous robots. Poster: Human Factors and Ergonomics Society 54th Annual Meeting, San Francisco.

CHAPTER 14 Authoring Conversation-based Assessment Scenarios

Diego Zapata-Rivera, Tanner Jackson, and Irvin R. Katz
Educational Testing Service

Introduction

At Educational Testing Service (Princeton, NJ), current research seeks to adapt technologies and techniques originally developed for intelligent tutoring systems (ITSs) to create innovative forms of assessment. This chapter focuses on one such project, working from the dialogue-based instruction of Graesser and colleagues (Graesser, Person & Harter, 2001) to develop a series of “conversation-based assessments” (CBAs). CBAs use dialogues between automated computer agents and test-takers to help measure the level of a construct — knowledge and skill in a particular domain — that a test taker possesses. To date, we have developed prototype CBAs each designed to measure a distinct skill such as science inquiry (Zapata-Rivera, Jackson, Liu, Bertling, Vezzu & Katz, 2014), formulating and justifying arguments (Song, Sparks, Brantley, Jackson, Zapata-Rivera & Oliveri, 2014), and reading, listening, and speaking skills for English language learners (Evanini, So, Tao, Zapata, Luce, Battistini & Wang, 2014).

The assessment, rather than instructional, context for dialogues lead to unique challenges when designing CBAs. To meet these challenges, we have built authoring tools to support the processes of designing and developing automated conversations for assessment purposes. These tools include conversation-space diagrams (Zapata-Rivera et al., 2014), an automated testing tool, and a version of the AutoTutor Script Authoring Tool (Susarla, Adcock, Van Eck, Moreno & Graesser, 2003), which we call the AutoTutor Script Authoring Tool for Assessment (ASATA).

Each task within a CBA is defined to measure a particular set of constructs; a conversation-space diagram shows how evidence (test taker performance) of each construct is collected through various discourse paths of the conversation. The diagram helps the designers to place recognizable discourse patterns in the conversations to create authentic-seeming situations. These conversation-space diagrams lead directly to dialogue scripts in ASATA and test-taker response “scripts” for the automated testing system. These authoring tools have helped speed up the design and testing of CBAs.

Related Research

As the need for assessing more complex skills increases, more researchers are exploring the use of new technologies to implement technology-enhanced assessments (TEAs) that can make use of multiple sources of evidence to support claims about students’ skills, knowledge and other attributes (Invitational Research Symposium on Technology Enhanced Assessments, 2012; Perrotta & Wright, 2010). Some of these TEAs include the use of computer simulations (Bennett, Persky, Weiss & Jenkins, 2007; Clarke-Midura, Code, Dede, Mayrath & Zap, 2011; Quellmalz et al., 2011) and games (Shute, et al., 2009; Mislevy, et al., 2014). TEAs frequently involve the use of authoring tools to facilitate the design and implementation process of these systems.

A variety of authoring tools have been implemented and evaluated for ITSs. These tools include authoring tools for dialogue systems (Susarla, et al., 2003; Butler, et al., 2011), constraint-based tutors (Mitrovic, Martin, Suraweera, et al., 2009), model-tracing cognitive tutors (Aleven, McLaren, Sewall & Koedinger,

2006, Blessing, Gilbert, Ourada & Ritter, 2009) and other problem-specific tutors (Blessing et al., 2009). An overview of authoring tools in ITSs can be found in Murray (2003). Relevant research also includes prior work on authoring tools for creating data collection instruments (Katz, Stinson & Conrad, 1997).

Although this prior work provides a guide, the intent of a conversation differs between an ITS and an assessment, changing also the elements of a dialogue. When a conversation is part of an ITS, the primary goal is instruction. Graesser's dialogic framework consists of a computer agent asking a main question of the human student, then following up with additional questions or prompts if the initial response is incomplete. Different follow-up questions, prompts, or hints would be offered depending on the specific way that the initial response is incorrect or incomplete. In the case of assessment, follow-up questions, prompts, and hints take on a new meaning. Rather than guiding the student to a good answer to the main question, the goal of the assessment is to make sure that any incompleteness in the initial answer reflects that the student does not know the answer rather than the student simply did not express what he or she knows. Of course, in an assessment, even if the student answered a question incorrectly or provided an incomplete answer, the system would not attempt to teach, but rather create situations that help students elaborate on their initial incomplete or incorrect responses. These sequences of interactions are recorded (and later scored). Thus, compared with the original AutoTutor framework, the assessment focuses on incompleteness and in drawing out additional information about student understanding.

Discussion

Traditional authoring tools for dialogue systems assume that authors are familiar with computer natural language processing techniques (e.g., regular expressions and latent semantic analysis) and have some computer programming skills (e.g., rule-based and constraint programming). Most of these tools have been designed to be used by dialogue engineers who have years of experience designing, implementing, and testing these types of systems. Even though assessment developers are highly skilled at developing valid assessments using traditional task types, they are not familiar with the use of conversations as assessment tasks and do not usually have programming experience. In addition, other team members such as psychometrists, game programmers, research assistants, and research scientists do not necessarily understand how these CBAs are created and scored.

In order to support the work of assessment developers, a different layer of authoring needed to be explored. This layer includes support for assessment design concepts and processes (e.g., target constructs, evidence identification, and scoring). Although we have tried several tools in the creation of CBAs, such as text documents and chat-like tools to document how dialogue interactions are used to gather evidence of target constructs, these tools quickly became cumbersome to use and did not include all the elements required to develop assessment tasks. Conversation-space diagrams were created to facilitate the authoring of conversational tasks. The next sections describe the process of authoring CBAs.

Authoring CBAs

Building on the principles of evidence-centered design (ECD; Mislevy, Steinberg & Almond, 2003), the development process of these conversation-based tasks involves an iterative process that starts from a clear definition of the construct, followed by the identification of the evidence (e.g., types of responses to particular questions) required to support particular claims about what students know or can do in regards to each target construct (Figure 1).

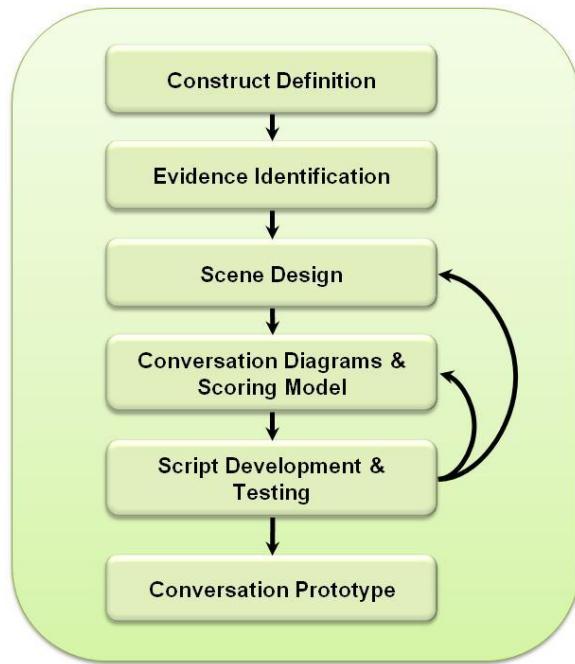


Figure 1. CBA development process

Scenes are designed in order to create the context where intended conversations can take place and the evidence needed can be gathered. Scene design elements include the situation or context of the conversation, main question, conversation moves/patterns, who asks each question, the type of responses that are intended to be elicited, and how characters respond to each type of response. This information is represented in a conversation space diagram (see the next section). A scoring model is also developed for each particular conversation. The scoring model has two components: (1) path-based scoring (partial credit scores per each relevant construct based on expert judgment) and (2) revised scores based on additional evidence from human raters or other automated scoring engines. Conversation scripts are implemented in ASATA based on these conversation space diagrams. These conversation scripts can be tested within ASATA (text-based interface). Finally, a conversation prototype that includes all the graphical components, interactive tasks (e.g., simulations) and conversations is produced and used to collect assessment data. These data are used to refine the various elements of the system in an iterative cycle.

Conversation Space Diagrams

Conversation diagrams have been designed to facilitate authoring of CBAs (see Figure 2). These diagrams serve as communication tools to facilitate communication about task design among an interdisciplinary group of experts that may not share the same location or have the same level of expertise in particular area. Conversation space diagrams provide a common language for these experts to collaborate in the CBA design and testing process.

Conversation space diagrams include the definition of the construct that is being assessed along with a column for each virtual character/real student. Utterances and potential conversational branches are displayed in the body of the diagram to form conversation paths (including sample user responses). These paths may involve several turns (i.e., columns of the diagram) depending on the conversation. Paths within a diagram can be used to represent several types of conversation moves/patterns (e.g., Comparison

-> Selection -> Agree/Disagree -> Why?; Define -> Explanation -> Scaffolding -> Rephrase; Irrelevant -> Rephrase & Ask Again).

Interactions with characters are designed to provide opportunities for assessing the construct(s) of interest. Each task includes an opening that sets the stage for the interactions with virtual characters and a closing that concludes the current scene and connects it to the next one. Each scene includes a main question that is directed to the student. Depending on how the student responds to this question, virtual characters react.

There is usually a predefined set of possible responses: (a) a correct response is usually connected to a closing statement, (b) partially correct responses are handled in various ways depending on the nature of the response (e.g., characters may ask for additional information, provide a hint, or restate the question), (c) irrelevant responses are usually handled by a character showing lack of understanding and restating the question, (d) no response usually involves a character asking “Are you still thinking?” and giving the student additional time, if appropriate, and (e) meta-communicative responses (e.g., “What did you say?,” “Please repeat”) and meta-cognitive responses (e.g., “I have no idea,” “I am not sure,” “I forgot”) are handled by repeating the question or rephrasing it.

Each conversational script typically has up to three cycles or opportunities for students to answer different types of questions related to the main question. If the student does not answer the question after the initial attempt and follow-up prompts, then a character may provide a closing statement and move the conversation along to the next scene.

Path information is based on expert judgment and is implemented using regular expressions and rules as part of the script. Figure 2 shows sample closing statements including path-based scores for the target constructs. Figure 3 (top) shows a sample rule telling the system that ClosingPath1 should be executed if the student response is classified as Good. A fragment of a regular expressions for a good response is displayed at the bottom of Figure 3. Path-based scores for target constructs are assigned at the closing statement.

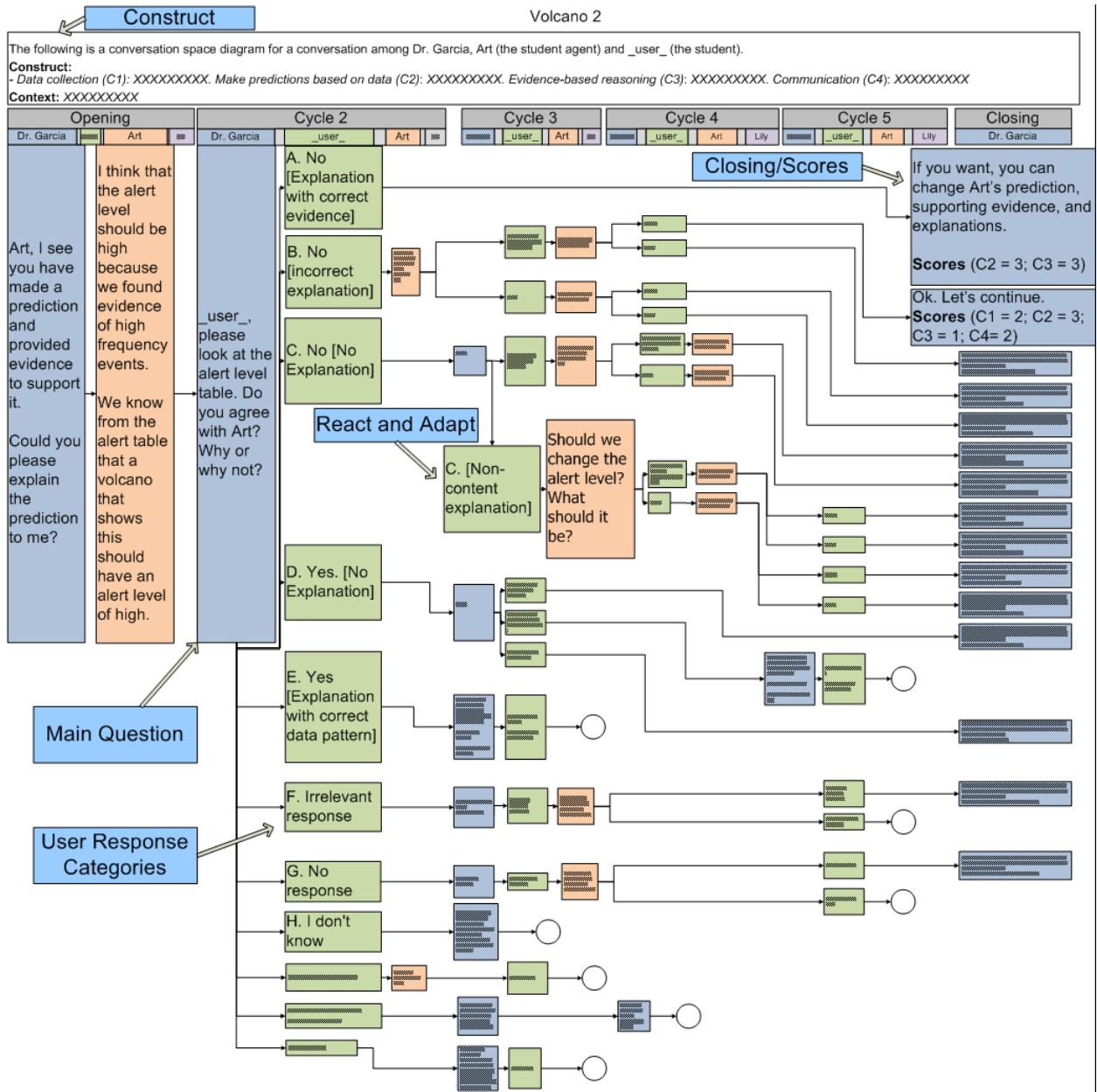


Figure 2. Fragment of a conversation space diagram for the Volcano scenario (Zapata-Rivera, et al., 2014).
Note: To allow exemplar text to be legible, the text in other boxes was purposefully obscured in this figure.

The screenshot displays the ASATA (AutoTutor Script Authoring Tool for Assessment) interface. It includes four main sections:

- Rules**: A table showing rule definitions. Row 4: AskMainQuestion, status MainQuestion, response Main-Good, event true, priority 2, frequency 2. Row 5: QuestionGood, status AskMainQuestion, response Main-Good, event true, priority 2, frequency 2. Row 6: QuestionGoodCritique, status AskMainQuestion, response Main-GoodCritique, event true, priority 2, frequency 2.
- QuestionGood**: A table showing steps for the QuestionGood rule. Step 1: System, act GetRigidPack, data ClosingPath1. Step 2: System, act Set Status, data ClosingPath1. Step 3: System, act GetRule. Step 4: *4, act GetRule.
- Question**: A form for creating a question. Fields include Question ID (1), Agent (Professor), Text (_user_, please look at the alert level table. Do you agree with Art? Why or why not?), Speech (_user_, please look at the alert level table. Do you agree with Art? Why or why not?), and Media (dropdown).
- Answers**: A table showing answers. Row 1: Good, agent Professor, type Good, keys containing a large regular expression fragment.

Figure 3. Sample rule and regular expression (fragment) in ASATA

AutoTutor Script Authoring Tool for Assessment (ASATA)

ASATA offers many features for creating a variety of conversation-based tasks. Dialogue engineers can create, test, and revise conversations for tutoring and assessment purposes using the modules available in ASATA.

ASATA provides conversation authors with a graphical interface that includes modules such as the following:

- Agents this module is used to define agent characteristics such as name, title, gender, and canned expressions for predefined categories of responses (e.g., meta-communicative responses);
- Speech Acts used to define regular expressions for general categories of responses;
- Rigid Packs used to represent non-interactive conversations among the agents (e.g., opening and closing statements);

- Tutoring Packs determine how agents react to particular student responses, establish thresholds for classification purposes, and contain linguistic information like regular expressions, text for latent semantic analysis, expected answers, misconceptions, hints, and prompts;
- Rules Implement conversation sequences (paths); and
- a Testing module that uses a chat-like environment to display the internal state of the system (e.g., rules fired and matching values) as the user interacts with each conversation.

ASATA shares many of the same features as ASAT. Many of the improvements made to ASAT have been transferred to ASATA and vice versa. Figure 3 shows some of the components of ASATA.

Automated Testing

Testing CBA scripts can be a time-consuming process of manually entering possible student responses and observing whether the conversation flows as expected. This process usually requires several iterations of testing/refining, which becomes a bottleneck for the use of these systems in operational contexts. We have developed an approach for automated testing of CBAs. This process makes use of sample responses gathered from an interdisciplinary group of experts and allows for the creation of predefined response categories that are represented in the form of a conversation diagram. This information is used to create extensible markup language (XML)-based testing scripts that can evaluate individual responses and complete sequences (and alternative sequences) of responses (paths). This process has already shown value by reducing the number of iterations and testing time required in implementing CBAs. The next section describes some of the results that have been achieved so far using various authoring and testing tools.

Initial Results

We have implemented CBAs in various areas including English language learning, mathematics, science, and argumentation during the last two years. Conversation space diagrams for these domains may share similar components including path structure, conversation patterns, and graphical components (e.g., virtual characters and delivery environment). This helps in terms of reducing the cost of CBA development and improving scalability. For example, it is possible to design a parallel version of a CBA by reusing and adapting the elements from an existing one. We are currently testing a newly developed isomorphic environment to an existing CBA and comparing them in terms of their psychometric and other properties.

By using conversation space diagrams, we have been able to assign work that was done by scientists or assessment developers to research assistants (e.g., modification of conversation based diagrams, generation of additional materials, and testing), making better use of resources while still producing high quality work.

We have collected data on the time required by our team to design and test CBAs across different target domains. The development can be divided into three different stages based on various authoring tools available for designing and testing CBAs. Initially, we used text documents and chat tools to create the scripts before using ASATA to implement them and testing was done manually. Later, we started using conversation-based diagrams design and ASATA and manual testing. Currently, we are using conversation-based diagrams, ASATA, and automated testing. The introduction of automated testing of scripts has made the process of detecting and fixing errors more efficient. Table 1 shows some indicators of the development and testing process using various types of authoring tools. These data have been

collected at different stages of this work. The introduction of conversation based diagrams and automated testing have increased the efficiency of the process. We continue to make improvements to these tools by enhancing their usability and integration with other development tools.

Table 1. Development indicators for conversation-based assessments using various authoring tools.

Development Indicator	Authoring Tool		
	Text Documents/Chat tools + ASATA	Conversation Space Diagrams + ASATA	Conversational Space Diagrams + ASATA + Automated Testing
Number of scripts developed	2	8	~20
Time designing and testing a new script	4–8 weeks	1–4 weeks	1–2 weeks
Percentage of errors identified before data collection	20%–30%	40%–60%	60%–80%
Time correcting errors	1 week	2–4 days	1–2 days

Next Steps

The conversation space diagram and testing system are distinct components, separate from each other and ASATA, the latter of which is the way that conversations are implemented. In our current work, we are combining these three systems into an authoring tool that we call ASAT-V (“V” for visualization). ASAT-V is a visual programming environment in which an author draws the conversation space diagram and, for each node, specifies metadata (what is to be said by which agent, for example). Another user of the system, the dialogue engineer, would add metadata associated with the technical aspects of the conversation, such as regular expressions and other parameters to ensure that the conversation would work correctly. The psychometrician might add in scoring metadata associated with scoring. Once the diagram is created in ASAT-V, it produces files that can be read by the dialogue engine to execute a dialogue, with no intermediary steps. Additionally, the ASAT-V will produce testing scripts and script models (tailorable by the author) and execute those scripts to ensure that the conversation flows as expected.

Recommendations and Future Research

Authoring tools such as conversation space diagrams and automated testing modules facilitate the development and testing process of CBAs. Through our authoring tools, we have been able to reuse domain-independent structures (e.g., conversation patterns), accelerate the development of CBAs, and improve communication among the members of our development teams. Conversation-based diagrams have also helped new members get familiar with these innovative assessment tasks and improved the acceptance of a new assessment design paradigm. In addition, it has allowed for an effective allocation of resources so people can do what they know best.

Some recommendations for the Generalized Intelligent Framework for Tutoring (GIFT; Sottilare, Brawner, Goldberg, and Holden, 2012), and future ITS include the following:

- Develop integrated authoring tools that take into account the needs, knowledge, and attitudes of particular team members.
- Keep important design information readily available throughout the development process (e.g., construct information for assessment tasks)
- Develop technical development infrastructure and representations to help integrate/reuse components across different CBAs.
- Make use of automated testing tools to help speed-up the development and testing process of conversation-based systems.

References

- Aleven, V., McLaren, B.M., Sewall, J. & Koedinger, K.R. (2009). A New Paradigm for Intelligent Tutoring Systems: Example-Tracing Tutors. *International Journal of Artificial Intelligent in Education. Special Issue on Authoring Systems*, 19(2), 105-154.
- Bennett, R. E., Persky, H., Weiss, A. & Jenkins, F. (2007). Problem-Solving in technology rich environments: A report from the NAEP technology-based assessment project. NCES 2007-466, U.S. Department of Education, National Center for Educational Statistics, U.S. Government Printing Office, Washington, DC.
- Blessing, S. B., Gilbert, S. B., Blankenship, L. A. & Sanghvi, B. (2009). From sdk to xpst: A new way to overlay a tutor on existing software. In Proceedings of the Twenty-second International FLAIRS Conference (pp. 466-467), Sanibel Island, FL. AAAI Press.
- Blessing, S. B., Gilbert, S.B., Ourada, S. & Ritter, S. (2009). Authoring model-tracing cognitive tutors. *International Journal of Artificial Intelligent in Education. Special Issue on Authoring Systems*, 19(2), 189-210.
- Butler, H., Forsyth, C., Halpern, D., Graesser, A.C. & Millis, K (2012). Secret agents, alien spies, and a quest to save the world: Operation ARIES! Engages students in scientific reasoning and critical thinking. In R. L. Miller, R. F. Rycek, E. Amsel, B. Kowalski, B. Beins, K. Keith & B.Peden (Eds.), Volume 1: Programs, Techniques and Opportunities. Syracuse, NY: Society for the Teaching of Psychology.
- Clarke-Midura, J., Code, J., Dede, C., Mayrath, M. & Zap, N. (2011). Thinking outside the bubble: Virtual performance assessments for measuring complex learning. In M.C. Mayrath, J. Clarke-Midura & D. Robinson (Eds.), Technology-based assessments for 21st century skills: Theoretical and practical implications from modern research. Charlotte, NC: Information Age. 125-147
- Evaniini, K., So, Y., Tao, J., Zapata, D., Luce, C., Battistini, L. & Wang, X. (2014). Performance of a triologue-based prototype system for English language assessment for young learners. Proceedings of the Interspeech Workshop on Child Computer Interaction (WOCCI 2014), Singapore, September 19, 2014.

- Graesser, A. C., Person, N. K. & Harter, D. (2001) The Tutoring Research Group: Teaching tactics and dialogue in AutoTutor International Journal of Artificial Intelligent in Education. 12, 257-279.
- Katz, I., Stinson, L.L, and Conrad, F.G. (1997). Questionnaire designers versus instrument authors: Bottlenecks in the development of computer administered questionnaires. Fifty-Second Annual Conference of the American Association for Public Opinion Research, Norfolk, VA. 1029-1034.
- Mislevy, R., Oranje, A., Bauer, M., von Davier, A., Hao, J., Corrigan, S., Hoffman, E., DiCerbo, K. & Michael, J. (2014) Psychometric Considerations In Game-Based Assessment. Retrieved October 5, 2014, from http://www.instituteofplay.org/wp-content/uploads/2014/02/GlassLab_GBA1_WhitePaperFull.pdf
- Invitational Research Symposium on Technology Enhanced Assessments. (2012). Center for K–12 Assessment & Performance Management at ETS. Retrieved October 5, 2014, from http://www.k12center.org/events/research_meetings/tea.html
- Mitrovic, A., Martin, B., Suraweera, P., Zakharov, K., Milik, N., Holland, J. & McGuigan, N., (2009) ASPIRE: An Authoring System and Deployment Environment for Constraint-Based Tutors, International Journal of Artificial Intelligent in Education. Special Issue on Authoring Systems, 19(2), 155-183.
- Murray, T. (2003) An Overview of Intelligent Tutoring System Authoring Tools: Updated Analysis of the State of the Art. *Authoring tools for advanced technology learning environments*. 491-545.
- Perrotta, C. & Wright, M. (2010) New Assessment Scenarios. Retrieved October 5, 2014, from <http://www.futurelab.org.uk/resources/new-assessment-scenarios>
- Quellmalz, E. S., Timms, M. J., Buckley, B. C., Davenport, J., Loveland, M. & Silbergliitt, M. D. (2011). 21st Century Dynamic Assessment. In M.C. Mayrath, J. Clarke-Midura & D. Robinson (Eds.), *Technology-based assessments for 21st century skills: Theoretical and practical implications from modern research*. Charlotte, NC: Information Age. 55-90.
- Shute, V. J., Ventura, M., Bauer, M. I. & Zapata-Rivera, D. (2009). Melding the power of serious games and embedded assessment to monitor and foster learning: Flow and grow. In U. Ritterfeld, M. J. Cody & P. Vorderer (Eds.), *Serious Games: Mechanisms and Effects*. Philadelphia, PA: Routledge/LEA. 295-321.
- Sottilare, R., Graesser, A., Hu, X., and Goldberg, B. (Eds.). (2014). *Design Recommendations for Intelligent Tutoring Systems: Volume 2 - Instructional Management*. Orlando, FL: U.S. Army Research Laboratory. ISBN 978-0-9893923-3-4. Available at: <https://gftutoring.org/documents/>
- Song, Y., Sparks, J., R., Brantley, J. W., Jackson, T., Zapata-Rivera, D. & Oliveri, M. E. (2014) Developing Argumentation Skills through Game-Based Assessment. In Proceedings of the 10th Annual Game Learning Society Conference, Madison, WI.
- Susarla, S., Adcock, A., Van Eck, R., Moreno, K. & Graesser, A.C. (2003) Development and evaluation of a lesson authoring tool for AutoTutor. In: Aleven, V., et al. (eds.) AIED 2003 Supplemental Proceedings, pp. 378–387
- Zapata-Rivera, D., Jackson, T., Liu, L., Bertling, M., Vezzu, M. & Katz, I. R., (2014) Science Inquiry Skills using Trialogues. 12th International conference on Intelligence Tutoring Systems. 625-626.

Chapter 15 Authoring Networked Learner Models in Complex Domains

David Williamson Shaffer¹, A. R. Ruis¹, and Arthur C. Graesser²

¹University of Wisconsin–Madison, ²University of Memphis

Introduction

Education leaders have called for a significant expansion in the use of computer games and simulations, intelligent tutoring systems (ITSs), and other virtual learning environments in both formal and informal learning contexts (Graesser, 2013; Honey & Hilton, 2011; Sotilare, Graesser, Hu & Holden, 2013). To accomplish this will require that curriculum developers be able to author and customize such technologies for integration into specific curricula, adaptation to local needs, and alignment with changing standards (Clark, Nelson, Sengupta & D'Angelo, 2009; Honey & Hilton, 2011; Mitrovic et al., 2009). Research on the development of ITSs has shown that anywhere from 100 to 1000 hours of authoring time are needed to produce just 1 hour of instruction (Koedinger & Mitrovic, 2009). The substantial time commitment and expertise required place significant limitations on the creation of sophisticated virtual learning environments. “Our holy grail,” Vincent Aleven and colleagues have suggested, is “to create cost-effective tools that non-programmers can use to create and deliver sophisticated tutors for real-world use” (Vincent Aleven et al., 2009).

Prior studies on authorware development suggest that building such tools is both ambitious and potentially transformative (Koedinger, Aleven, Heffernan, McLaren & Hockenberry, 2004; Murray, Blessing & Ainsworth, 2003; Murray, 1999). Recent efforts to design authorware for sophisticated systems has revealed the many difficulties involved in creating a platform that is rich in features but easy to use. It is challenging for curriculum developers and instructors to use authoring tools effectively, and adding additional intelligent features could make it even more challenging (Ainsworth & Grimshaw, 2004; Major, Ainsworth & Wood, 1997). Authoring tools must be able to account for essential components, such as conversation management, semantic representations, production rules, pedagogical strategies, and other technical modules (Vincent Aleven, Sewall, McLaren & Koedinger, 2006; Murray et al., 2003; Woolf, 2010). The curriculum or learning modules created also need to fit theory-driven constraints, discourse processes, cognitive science, and computer science, as well as the practical constraints created by state standards, assessments, and education practices. Pedagogical authoring thus requires deep and broad knowledge to manage these constraints, accommodate tradeoffs, and negotiate incompatibilities.

The complexity inherent in the pedagogical authoring of virtual learning environments raises a key question: *Can authorware systems be designed that facilitate this process without requiring the curriculum developer to have expertise in computer programming or educational software development?*

While progress has been made toward this goal, most sophisticated authoring systems (there are many for ITSs alone) are used primarily in research contexts. Those that have received broader usage, such as Cognitive Tutor Authoring Tools (CTAT) and Authoring Software Platform for Intelligent Resources in Education (ASPIRE), primarily support the development of modules that help students learn to solve well-formed problems, such as those common in basic mathematics, computer science, or language acquisition. One notable exception is the AutoTutor Script Authoring Tools (ASAT and ASAT-Lite), which support intelligent conversational agents in any subject matter (Hu et al., 2009; Nye, Graesser & Hu, 2015; Cai, Graesser & Hu, this volume). ASAT handles a single human learner who interacts with

one or more conversational agents. In this chapter, we discuss the potential to develop authorware for virtual learning environments in which students work in small teams to solve complex, ill-formed problems. In particular, we explore the parameters, affordances, and challenges of designing authoring tools for *Syntern*, a platform for the development and deployment of *virtual internships* (Arastoopour, Chesler & Shaffer, 2014; Bagley & Shaffer, 2009; Chesler et al., 2015; Chesler, Arastoopour, D'Angelo, Bagley & Shaffer, 2013; Shaffer, 2007). Virtual internships are online learning environments that simulate professional practica in complex science, technology, engineering, and mathematics (STEM) domains.

Virtual internships are based on the theory of *situated learning* (Anderson, Reder & Simon, 1996; Lave & Wenger, 1991; Sadler, 2009), which suggests that students learn complex thinking best when they have an opportunity to take consequential action in a realistic setting. In STEM fields, this typically occurs in the context of an internship or other professional practicum through a process of *legitimate peripheral participation*, where novices learn to think like experts by working on problems similar in form to those of the practice but with reduced intensity and risk (Lave & Wenger, 1991). What distinguishes an internship from other learning environments is the combination of *action*, the ability to do authentic work, and *reflection-on-action* (Schön, 1983, 1987; Shaffer, 2003), the opportunity novices have to think about what went well, what did not, and why, and then discuss this with peers and mentors. Virtual internships simulate the key features of a professional practicum, especially the close mentorship that is critical to learning in professional contexts (Bagley & Shaffer, 2010; Nash & Shaffer, 2011, 2013; Nulty & Shaffer, 2008).

In a STEM virtual internship, students are presented with a complex, real-world problem for which there is no optimal solution. Student project teams read and analyze research reports, perform experiments using virtual tools and analyze the results, respond to the requirements of stakeholders and clients, write reports and proposals, and present and justify their proposed solutions. During the virtual internship, students communicate with one another using built-in email and instant message systems. They also receive directions, feedback, and guidance from non-player characters (NPCs), such as their boss or company stakeholders, whose actions are controlled by a combination of *artificial intelligence* (AI) and human *domain managers* using scripted material in the simulation. Through flexible scripts and automated processes, NPCs answer students' questions, offer suggestions, guide reflective conversations, facilitate student collaboration, and provide support. The goal of a virtual internship is to provide an authentic simulation of the internships, practica, and cooperative research experiences with which STEM professionals are trained in the real world.

In the virtual internship *Nephrotex*, for example, students work at a fictitious biomedical engineering company, which has tasked them with designing a new ultrafiltration membrane for use in hemodialysis equipment. To accomplish this task, students review technical documents, conduct background research, and examine research reports based on actual experimental data. After these tasks are complete, they develop hypotheses based on their research, test those hypotheses in the provided design space, and then analyze the results, first individually and then in teams. Students also become knowledgeable about consultants within the company who have a stake in the outcome of their designed prototypes. These consultants value different performance metrics. For example, the clinical engineer is most interested in biocompatibility and flux, and the manufacturing engineer values reliability and cost. During the last days of the internship, interns present and justify their final design selections.

Our goal is to develop authorware that allows curriculum developers to design or modify STEM virtual internships to address different audiences, topics, or purposes without requiring significant expertise in computer programming or educational software development. We believe this is possible because (a) the *pedagogical foundation is well developed and the design space is constrained*, reducing the specialized knowledge required for pedagogical authoring; (b) the *computational module for natural language*

processing (NLP) is STEM domain general, so it does not require rewriting for new STEM virtual internships; updates to the semantic coding system automatically propagate to the AI modules; and (c) the Syntern platform has a *modular design* consisting of a core application programming interface (API) and plug-ins, so each component may be added, removed, or modified without affecting other components. Although we focus on the design of one particular system, the principles of authorware design are applicable to learning environments in ill-formed domains more generally. Given the relatively small body of research on the processes with which curriculum developers design content, however, we argue that a key element of developing such authorware systems is to develop a science of the pedagogical authoring process.

Related Research

In the past two decades, there has been a proliferation of sophisticated virtual learning environments in STEM. There are now ITSs that can outperform human teachers on certain tasks, such as determining student knowledge and identifying student misconceptions (Graesser, Conley & Olney, 2012; Woolf, 2010). STEM educational games, such as *Quest Atlantis* (Barab et al., 2009; Hickey, Ingram-Goble & Jameson, 2009), *River City* (Dieterle, 2009; Ketelhut, Dede, Clarke-Midura & Nelson, 2006), *SAVE Science* (Nelson, Ketelhut & Schifter, 2010), *Operation ARA* (Halpern et al., 2012), and *Mission Biotech* (Sadler, Romine, Stuart & Merle-Johnson, 2013), have been shown to help students learn important STEM concepts and engage more fully with material. And our own work on STEM virtual internships has shown that computer simulations based on authentic STEM practices help students learn how to solve problems in the ways that innovative STEM professionals do (Arastoopour et al., 2014; Bagley & Shaffer, 2009; Chesler et al., 2015, 2013; Shaffer, 2007).

Despite these successes, use of such technologies in education is still quite modest. If ITSs, educational games and simulations, and virtual internships are so effective, why have they not been more widely incorporated into learning? There are numerous issues that contribute to this problem, but a key element is that it is *too difficult, too expensive, and too slow* to create or modify sophisticated learning technologies to fit the wide range of learners and learning contexts. Creating authorware that enables curriculum developers to easily, cheaply, and quickly produce or modify learning technologies, while also ensuring that the products are pedagogically sound, is thus a crucial requirement for scaling up the use of such technologies in education. While this research and development effort is still in its early stages, significant steps have been taken toward this goal.

A number of authorware systems have been developed that allow curriculum developers to construct virtual learning environments in which students learn to solve problems in a variety of domains. Initial research on CTAT, for example, found that an example-tracing approach to pedagogical authoring, which requires no programming ability, cut authoring time by as much as 50% (Vincent Aleven, McLaren, Sewall & Koedinger, 2006). CTAT is now perhaps the most widely used authoring system for ITSs, and the gains in efficiency have improved as well. Large-scale CTAT-created tutors used in educational settings have been built with fewer than 100 hours invested per hour of instruction produced. By eliminating the need for programming assistance, CTAT can reduce overall development costs by a factor of 4–8 (Vincent Aleven et al., 2009).

Similarly ASPIRE, an authorware platform for the creation of constraint-based ITSs, has been used to create a wide range of learning technologies (Mitrovic, 2012; Mitrovic et al., 2009). ASPIRE is domain agnostic, allowing curriculum designers in any field to author ITSs. This generality is a tremendous advantage, but it also means that the learning curve is steep for new users and that best results have been achieved by authors with more advanced technological abilities (Mitrovic, 2012).

Of course there are many other authoring tools, as well as other approaches to authorware design. But there remains a fundamental challenge: in making the authoring process easier and faster, the more advanced features of cognitive tutors are often lost. Example-tracing systems, for instance, significantly reduce authoring time and require no programming skill, but the pseudo-tutors produced are not as dynamic as those that expert programmers can build (Vincent Aleven, Sewall, et al., 2006; Koedinger et al., 2004). As a result, most of the learning modules that have been produced with accessible authoring tools help students learn to solve *well-formed* problems. But many problems with which innovative professionals engage are *ill formed*, requiring the kinds of complex thinking that is beyond the capacities of most systems to teach, unless the system has the capacity for natural language interaction and a statistical representation of world knowledge (Graesser, D'Mello, et al., 2012; Halpern et al., 2012; Hilton, 2008; McNamara, Levinstein & Boonthum, 2004; Rotherham & Willingham, 2009; VanLehn et al., 2007). Given this issue, a key next step in the development of authorware is to enable curriculum developers, even those with limited technological skill, and design virtual learning environments that simulate realistic practices or allow students to solve ill-formed or non-routine problems.

Discussion

Virtual internships, and the Syntern platform with which they are developed and deployed, have three key features that make it possible to develop authorware for curriculum developers who have limited technological skill: (1) the design space is constrained, (2) the NLP components are STEM domain general, and (3) the system is modular. Although virtual internships simulate ill-formed domains and problems, these three elements reduce the scope and complexity of the pedagogical authoring environment, allowing us to design authoring tools that can scaffold the curriculum development process. Of course, this limits the range of virtual learning environments that a curriculum developer will be able to design, but it also ensures that the final product will be functional, pedagogically and structurally sound, and able to accurately simulate non-routine problem solving in a practice-based context. In what follows, we first describe the existing Syntern platform. Then, we outline the design of an authorware system for Syntern virtual internships, and in doing so, we discuss our approach to studying the pedagogical authoring process itself. Although we focus on one specific system, the principles of authorware design that we discuss are generalizable to learning environments in ill-formed domains as a whole.

Syntern, a Modular Development and Deployment Platform for Virtual Internships

The Syntern virtual internship platform (Figure 1) is comprised of six distinct structural elements, which when combined produce (a) an online *user experience* that authentically simulates real-world STEM practices, and (b) a *log file* that records all the actions and interactions of students and domain managers in the system for subsequent analysis:

- (1) *Frameboard*. The Syntern frameboard contains the content for each STEM virtual internship and determines the sequence and structure of activities in the virtual environment. For example, virtual internships consist of a progression of *rooms*. Each room consists of three related and sequential *activities*: (a) an *introduction*, in which interns receive a specific task from their supervisor via email; (b) a *sandbox*, which contains the tools and resources interns need to complete the task; and (c) one or more *deliverables*, or the work output that the supervisor has asked interns to submit, including a notebook entry documenting their work. The frameboard is structured as a series of possible actions that the computer-generated NPCs use to interact with students in the internship. The Syntern system tracks students' progress through the internship and presents the human domain manager with context-appropriate choices for NPC action,

including grading rubrics for deliverables that students complete, response options for student questions, and guide questions for reflective discussions.

- (2) *Workbench*. The workbench provides actual or simulated tools from the STEM domain that help students solve problems in the field. In the urban planning virtual internship *Land Science*, for example, students use a geographic information system to model the effects of land-use changes on various social, economic, and environmental indicators.
- (3) *Templates for Automated Mentoring*. Syntern uses NLP algorithms to automatically deliver some content from the frameboard (such as task assignments from the supervisor). During team meetings, for example, in which student project teams discuss their recent activities with the NPC mentor and plan their next steps, the system can use the AutoReflect template to determine when student responses achieve a pre-defined learning objective. This helps the domain manager decide whether to revoice the response(s) and move on to a new topic or send a follow-up question to provide further scaffolding. In an engineering design simulation, for example, students may graph data to get a better sense for how various design choices affect certain performance metrics. After this activity, one question the NPC mentor may ask during the team meeting is: *Based on your surfactant graph, how did the surfactants perform relative to one another?* Because no one surfactant performs best on all performance metrics in this particular case, the target student response would be something like: *No surfactant performed best on all the design attributes*. The AutoReflect template uses an automated coder (see component five, below) to code student discourse in real time, alerting the domain manager when a student response achieves this goal. Of course, questions, targets, and coding criteria must be defined in advance, which requires experience in both curriculum design and educational technology development.
- (4) *Assessment Rubrics*. The frameboard contains an assessment rubric for every deliverable in the virtual internship. Assessment criteria are linked to pre-composed responses from the NPC supervisor. A custom NLP module uses a range of syntactic and semantic criteria, including word count, sentence complexity, and a domain-specific coding scheme, to determine whether a deliverable is *above threshold*, meaning it clearly meets evaluation criteria established by the rubric. Deliverables that are above threshold are automatically approved by the Syntern system, and the appropriate response from the NPC supervisor is sent. Deliverables that are not clearly above threshold are tagged by the system for manual evaluation by the domain manager using the assessment rubric.
- (5) *Domain Coder*. The automation of functions in a virtual internship is made possible by a domain coder that uses a combination of keywords and regular expressions to code chat messages, emails, notebook entries, and students' actions in the system for specific attributes of the domain.
- (6) *Application Programming Interface*. The API ensures that all Syntern elements integrate seamlessly and allows for easy addition, modification, or removal of modules. The API is comprised of six core components (Figure 1). The *Java 7 Hub* governs basic operations and links content, assessment, and the user experience. The *R Project for Statistical Computing* (R) supports NLP and learning analytics tools. The *MySQL database* holds content from the frameboard and records the actions of students and domain managers during the virtual internship. The *NLP module* uses R and the domain coder to analyze student and mentor interactions in the system. The *learning analytics module* evaluates coded discourse, deliverables, and other activity in the system to determine whether pre-defined learning objectives have been met. The *WorkPro graphical user interface* (GUI) simulates an online productivity suite through which students access resources and tools and interact with NPCs and their project team. The API thus ensures integration of the frameboard (curriculum content), workbench, automated

mentoring templates, assessment rubrics, and domain coder and provides the core architecture needed to produce a coherent user experience from them.

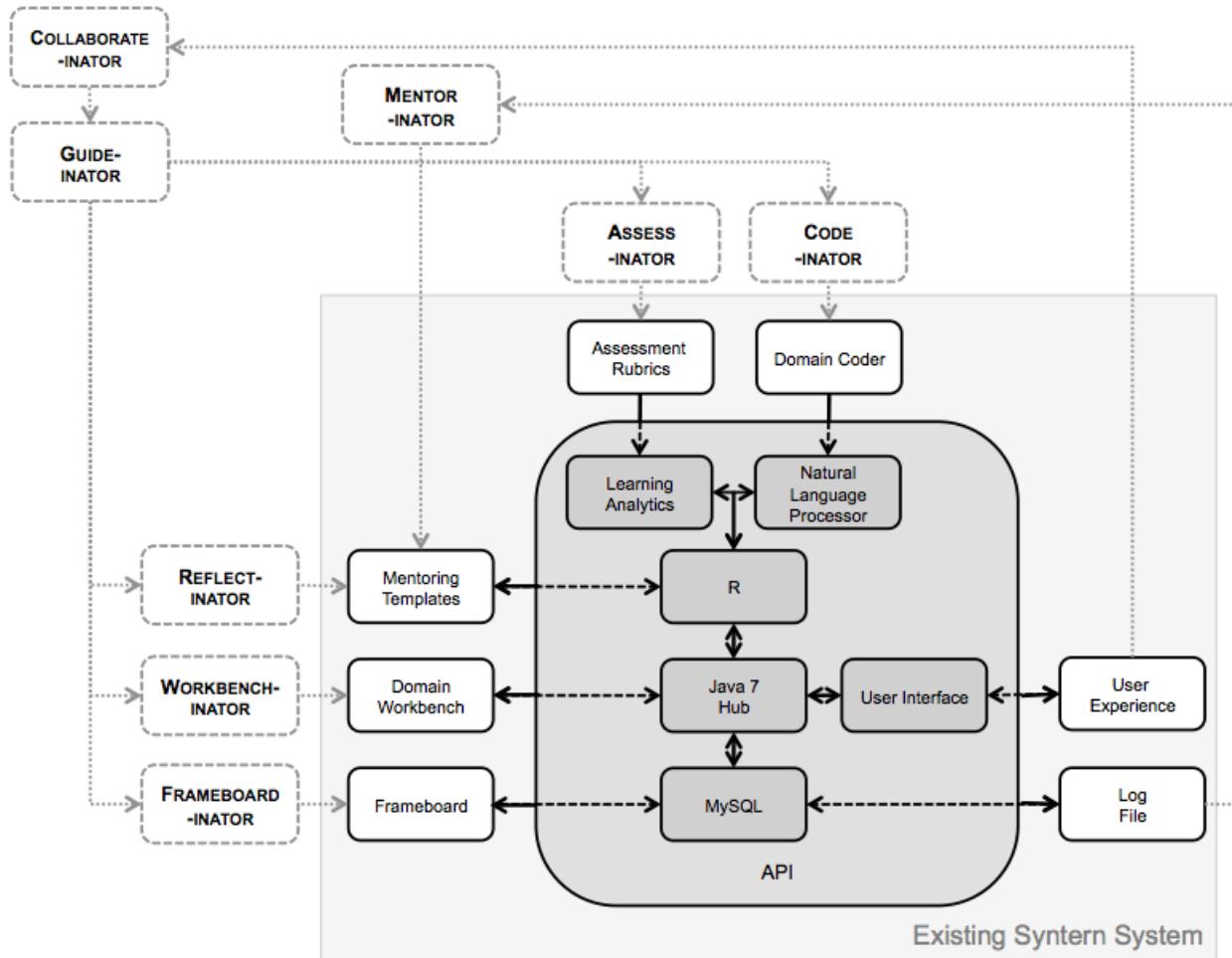


Figure 3. The existing Syntern virtual internship system and the eight components of the Internship-inator authorware platform.

With these components, Syntern recreates the key elements of an internship experience in an online environment. The frameboard and the STEM workbench provide students with the ability to take consequential *action*; the frameboard, mentoring templates, and learning analytics (using the assessment rubrics, domain coder, and NLP) support *reflection-on-action*; and the learning analytics, user experience, and log files enable the *iterative development* of virtual internships.

The Internship-inator, an Authoring System for Syntern Virtual Internships

A key challenge for the development of authorware that would enable curriculum developers to design STEM virtual internships for the Syntern platform is that we lack a science of the pedagogical authoring process. In what follows, we describe plans for the Internship-inator, an authorware system for the Syntern platform. Our goal is not only to design a functional authorware system but to use both the design process and the resulting tools to study the pedagogical authoring process. Of course, studying one particular authoring context with a relatively small number of curriculum developers will not support

generalization to all pedagogical authoring contexts, but this research will suggest useful directions for future work.

Developing authorware for the Syntern platform thus merits a systematic investigation of the curriculum design process and requires iterative prototyping and refinement of the authoring tools. We conceive of this project as a form of design research (Brown, 1992; Cobb, Confrey, Lehrer & Schauble, 2003; Confrey, 2006; Kelly, Lesh & Baek, 2008), where initial hypotheses about authorware design, Syntern modularity, and pedagogical authoring are revised by subsequent research in each area. To do this, we will work with a core network of early-adopters: STEM curriculum developers who will help us create initial prototypes of the Internship-inator, use the system to modify and design virtual internships, and create support materials. To minimize development time and make evidence-based design decisions, we will develop different components of the authorware system as standalone modules (described in detail below) and employ a Wizard of Oz (Dow et al., 2005) approach. Rather than building a complete version of each component initially, we will build a minimum viable version of each tool. Specifically, we will only automate those processes that need to be run in real time during the content-development process. Wherever possible, we will use members of the development team to perform functions of the tool in its early stages, and later build automated systems to replicate and replace the work of these human experts.

Throughout these iterative design cycles, we will collect three kinds of data in order to study the pedagogical authoring process: (1) *focus groups and interviews* conducted with the early adopters before and during the design process and after they have used authorware prototypes will help us understand their approach to curriculum development, the supports they need to use the authorware effectively, and their preferences for features, user experience, and so forth; (2) the Internship-inator will document in log files the *actions and interactions of early-adopters while using the authoring tools*, giving us a rich record of authoring behavior for further analysis; and (3) *pre/post tests and log files from implementations of virtual internships* modified or created by curriculum developers will provide rich information about the quality of the learning simulations produced with the Internship-inator. Evaluation of the pedagogical authoring process will thus encompass investigation of both technology use (e.g., the human-computer interaction process) and the quality of the content produced.

Collecting these data will allow us to address fundamental research questions about pedagogical authoring: Are some components of the authorware system more useful for editorial versus creative use? Are some components used more (or easier to use) in conjunction with others? Which aspects of the system influence whether, how, and to what extent curriculum developers use different authoring components? And so forth. For example, we can look at the pattern of use of different authoring components, including the order in which components are accessed, the frequency and duration of use, and other log file data, combined with focus groups, to better understand how to sequence and scaffold the authoring tools within the system to align with pedagogical authoring practices.

We conceive of the Internship-inator as a suite of eight online authoring tools (see Figure 1). For analytical purposes, we divide the system into *content* components (the Frameboard-inator and Workbench-inator), *automation* components (the Reflect-inator, Assess-inator, Code-inator, and Mentor-inator), and *support* components (the Guide-inator and Collaborate-inator).

Content Development Components

- (1) *Frameboard-inator*. The Frameboard-inator will enable STEM content developers to create or modify content for a virtual internship, including the structure and sequence of activities, assignments (such as readings or videos), assessments and rubrics, and other content that students or domain managers will need during the virtual internship. A key challenge is ensuring that STEM content developers include all of the information that the Syntern system needs to make

the content function. Thus, the Frameboard-inator will require a GUI that indicates (a) what kinds of content are required to make each element of the simulation function, and (b) what kinds of content are acceptable for different portions of the simulation. For example, every room in a virtual internship begins with an email from the supervisor NPC describing the activities to follow. Emails have specific properties in the system, so the Frameboard-inator GUI will need to indicate to the curriculum developer that (a) an email is required to begin a room and (b) what the constituent components of an email are.

- (2) *Workbench-inator*. The Workbench-inator will provide mechanisms through which curriculum developers can include problem-solving tools, such as AutoCAD or Matlab, in a virtual internship. Open-source or editable tools, such as Geogebra or Google Maps, can be connected to the Syntern API if the curriculum developer has programming expertise. For tools that are not open-source but that store their output in one of Syntern's supported file types (including XML, JSON, YAML, HTML, CSV, TXT, and Properties), the Workbench-inator will provide an interface that lets the developer tag elements of the file as Syntern readable. (This will also work in a more limited way for graphics files in JPG, GIF, or PNG format for content such as location, date, and time.) Finally, the Syntern system will allow students in a simulation to upload any file as a deliverable. As long as the domain managers have the appropriate program to read the file, they will be able to assess it using the system's rubrics. In the first two cases (open source program or output), the Syntern system would be able to apply automated assessment rules to the deliverables created. In the third case (proprietary tool or output), the system will store and track the file, but a human would have to assess its content.

Automation Components

- (1) *Reflect-inator*. The AutoReflect template makes it easier for domain managers to control reflective conversations between students and NPC mentors by identifying a set of reflection topics and, for each topic, specifying (a) a set of prompting questions for the NPC to use, (b) a set of NLP rules and other components to identify possible appropriate responses to the topic, and (c) a pre-scripted revoicing of the key ideas about the topic that students should be able to articulate. While prompting questions (a) and a revoicing (c) are relatively easy for curriculum developers to construct, NLP rules and other components for identifying candidate answers (b) will be more difficult. The Reflect-inator will scaffold the construction of these NLP components. For example, curriculum developers could enter hypothetical answers from students (as well as incorrect answers, if desired), and from the set of answers and non-answers, the Reflect-inator would abstract a matching NLP rule. Because of the limited context of students responding to a specific question in a reflective meeting taking place at a specific point in a specific STEM simulation, we have found empirically that relatively simple rules can distinguish appropriate responses from inappropriate responses. We therefore hypothesize that a limited set of model responses will be sufficient for the system to extract functional rules.
- (2) *Assess-inator*. Assessment rubrics in the Syntern system can automatically determine whether certain student deliverables are acceptable. The system uses a custom NLP algorithm that involves three computations: (a) a word type count (the number of unique words used), (b) a domain code count, and (c) a measure based on four measures from the text analysis program Linguistic Inquiry and Word Count (Pennebaker, Booth & Francis, 2007). We are also currently exploring including latent semantic analysis of deliverables to further refine the accuracy of the automated scoring algorithm (Graesser, Penumatsa, Ventura, Cai & Hu, 2007). The current algorithm uses six thresholds, which determine whether the deliverable is accepted automatically or sent to the domain manager for further evaluation. We hypothesize that, as a result, a relatively small number of sample answers will be required for the Assess-inator to automatically compute

appropriate values for these thresholds. The Assess-inator will initially set all thresholds to zero—which means all deliverables will be reviewed by hand. Log files created when the simulation is run will include real examples of deliverables and the domain manager’s determination of whether or not they are acceptable. The Assess-inator will then use these data to adjust the thresholds automatically and with each subsequent iteration, the Assess-inator can automatically refine the adjustments over time.

- (3) *Code-inator.* The domain coder uses a combination of keywords and regular expressions to interpret student-generated chats, emails, notebook entries, and actions in the Syntern system. The resulting codes are then used by components of the system to automate responses to student verbal contributions and actions. The domain coder can achieve the level of semantic accuracy needed to create believable responses because the *domain of possible speech acts and actions in the virtual internship is limited* (Graesser & McNamara, 2012; Grishman & Kittredge, 1986; Richard & Lehrberger, 1982; Rupp, Gustha, Mislevy & Shaffer, 2010). Curriculum developers, however, will not be able to easily create appropriate sets of keywords and expressions.
- (4) We have already developed a tool, the HandCoder/AutoCoder, to create codesets for virtual internships. This tool takes either manufactured or real examples from the target context (that is, the STEM virtual internship) and uses a coding loop to create a set of keywords and expressions. In the coding loop, a user codes a subset of examples from the target context for a given domain code. These are compared to the existing codeset, and the user is able to adjust the codeset based on the discrepancies. Further excerpts are hand-coded, and the process is repeated until the desired level of agreement is reached—typically Cohen’s kappa > 0.69 , which is excellent for automated coding. Two key features support the rapid identification of an appropriate codeset: (a) the system computes changes in the level of agreement on the fly as keywords are added or removed from the codeset, and (b) a custom-written algorithm computes the confidence interval for the level of agreement, thus reducing the number of coded excerpts needed to establish an acceptable level. This system has been used in several different domains to establish coding schemes, and we hypothesize that it can be easily adapted for use by curriculum developers.
- (5) *Mentor-inator.* The frameboard for a Syntern virtual internship contains scripted responses from NPCs that the domain manager can send to students. The Mentor-inator will automatically extend the range of scripted material by (a) providing an interface through which curriculum developers can easily add custom responses from previous runs to the frameboard, and (b) updating the interface through which domain managers access scripted content so that they can manage the larger number of scripted responses. To do this, the Mentor-inator will automatically extract composed responses from the log file. Responses that were used multiple times will be automatically added to the frameboard. Responses that appear only once will be presented with their surrounding context to the curriculum developer, who can decide whether to include them as scripts in future implementations.

Support Components

- (1) *Guide-inator.* The Guide-inator will provide templates for curriculum developers to use in creating support materials for virtual internships, along with an Internship-inator user guide. The Guide-inator will be designed as a comprehensive interface to the Internship-inator and Syntern systems, integrating design, support, curricular, and implementation materials for curriculum developers.
- (2) *Collaborate-inator.* The Collaborate-inator will provide a social networking component to the Internship-inator system. Curriculum developers and educators will be able to create individual

accounts on the system, which will (at their discretion) be linked to their email or other social media tools. The Collaborate-inator will facilitate content-focused exchanges about the Internship-inator, Syntern system, and virtual internships. Users will be able to comment on and link to content directly from the system. The result, we hypothesize, will be a self-sustaining community of STEM content developers who can share virtual internship designs, curricula, and experiences. By providing critical feedback and input on one another's designs, support materials, and implementation practices, the Collaborate-inator will provide the "real world tips" that our focus groups suggest education professionals want to supplement formal information about such systems.

- (3) We hypothesize that this suite of authoring tools will enable curriculum developers to design and modify virtual internships without needing programming experience or extensive training. The pedagogical framework, the mode of communication (email and chat), and the structure of the intervention are all relatively fixed, which makes it easier to scaffold the design process and ensure that the product is pedagogically and structurally sound. The NLP computational module is STEM domain general, so the semantic coding system can be automatically updated; this reduces the need for curriculum developers to have expertise in instructional technology design. The modular design of the Internship-inator has several advantages. First, it will allow curriculum developers to develop simulations quickly for testing. For example, a functional virtual internship could be developed from scratch using only the Frameboard-inator; the resulting simulation would not have automated features, but those could be incorporated more gradually. The initial time commitment can thus be relatively low if a curriculum developer wants to experiment with virtual internship design or content. Second, the Internship-inator will allow curriculum developers to make precise modifications to virtual internships, such as adding resources or workbench tools, altering scripted content, or expanding the codeset, without altering the rest of the simulation. Lastly, the system will accommodate different design processes: there isn't a single, linear progression that all curriculum developers must follow. Of course, that can be a disadvantage as well, as it may make the learning curve steeper, but we believe this is a useful trade-off because it will allow curriculum developers to use the tools in the ways that best fit their specific needs and design approach.

Recommendations and Future Research

Designing authorware that makes the creation of virtual learning environments easier, cheaper, and faster is critical for expanding the use of ITSs, educational games, and virtual internships. Most authorware design research, however, has focused on the *technological* challenges. We suggest that developing a *science of pedagogical authoring* is an equally important but largely neglected aspect of this problem. Just as there are established sciences that systematically investigate the processes that underlie learning, writing, design, problem solving, and other human achievements, there needs to be a comparable science of creating advanced learning environments with authoring tools. This science would need to (a) track the behavior of authors; (b) identify technological features that promote or impede authors' progress and the quality of the final products; (c) collect verbal protocols on the design processes of the authors while authoring material; (d) modify the features of authoring tools as data are collected; (e) formulate a testable theory of the authoring process; and (f) identify characteristics of authors that predict authoring quality. The current lack of a science of the authoring process explains in part why most authoring is accomplished by experts.

Designers of authoring tools generally agree that it is important to document the many versions of authored content over time (i.e., the authoring process) and analyze the trajectory of changes: To what extent are the authors using particular components of the authorware? To what extent are particular

learning principles instantiated in the materials that end up being designed? What components are frequently deleted or modified? But such questions have yet to guide the development of a science of pedagogical authoring. A key goal of the Internship-inator project is to contribute to the foundation of such a science. By tracking the actions of curriculum developers who use the authoring tools (log files) and developing a community of users (Collaborate-inator) from whom we can obtain feedback, we will be able to study systematically the processes at work in pedagogical authoring.

Our vision is compatible with the Generalized Intelligent Framework for Tutoring (GIFT) architecture both pragmatically (scaling up) and technically. Scholars are aware of the challenges involved in scaling up and having an architecture that can handle different media. This level presents no significant problems. We do see two efforts needed to expand GIFT. First, there needs to be a mechanism to handle groups, teams, and other collaborations that go beyond the individual learner. The main technical challenge is organizing the database, grouping individuals, and making systematic claims about individuals, groups, and organizations. The data stream needs to be time stamped and populated with adequate metadata to handle multiparty and sometimes multiteam interactions. Second, there needs to be a systematic facility for handling the authoring analytics. We need to store data on multiple versions of software content and track the authoring process. This is required to build a science of the pedagogical authoring process.

We have made considerable progress in the design of authorware for sophisticated virtual learning environments, and there are many projects currently underway that are likely to continue and even accelerate this progress. To improve uptake of such environments, however, we must develop authorware that can be used successfully beyond the research context. Aleven and colleagues suggest that our Holy Grail is to create cost-effective, user-friendly authorware; we suggest that our El Dorado is to develop a science of the pedagogical authoring process.

Acknowledgements

This work was funded in part by the MacArthur Foundation, the National Science Foundation (DRL-0918409, DRL-0946372, DRL-1247262, DRL-1418288, DUE-0919347, DUE-1225885, EEC-1232656, EEC-1340402, and REC-0347000), the Institute of Education Sciences (R305H050169, R305C120001), the Office of Naval Research, and the Army Research Laboratory. The opinions, findings, and conclusions do not reflect the views of the funding agencies, cooperating institutions, or other individuals.

References

- Ainsworth, S. E. & Grimshaw, S. K. (2004). Evaluating the REDEEM authoring tool: Can teachers create effective learning environments? *International Journal of Artificial Intelligence in Education*, 14, 279–312.
- Aleven, V., McLaren, B. M., Sewall, J. & Koedinger, K. (2009). A new paradigm for intelligent tutoring systems: Example-tracing tutors. *International Journal of Artificial Intelligence in Education*, 19(2), 105–154.
- Aleven, V., McLaren, B. M., Sewall, J. & Koedinger, K. R. (2006). The cognitive tutor authoring tools (CTAT): Preliminary evaluation of efficiency gains. In M. Ikeda, K. D. Ashley & T.-W. Chan (Eds.), *Intelligent Tutoring Systems* (pp. 61–70). Berlin, Germany: Springer.
- Aleven, V., Sewall, J., McLaren, B. M. & Koedinger, K. R. (2006). Rapid authoring of intelligent tutors for real-world and experimental use. In R. K. Kinshuk, P. Kommers, P. A. Kirschner, D. Sampson & W. Didderen (Eds.), *Proceedings of the 6th IEEE International Conference on Advanced Learning Technologies (ICALT 2006)* (pp. 847–51). Los Alamitos, CA: IEEE Computer Society.
- Anderson, J. R., Reder, L. M. & Simon, H. A. (1996). Situated learning and education. *Educational Researcher*, 25(4), 5–11.
- Arastoopour, G., Chesler, N. C. & Shaffer, D. W. (2014). Epistemic persistence: A simulation-based approach to increasing participation of women in engineering. *Journal of Women and Minorities in Science and Engineering*, 20(3), 211–234.

- Bagley, E. A. & Shaffer, D. W. (2009). When people get in the way: Promoting civic thinking through epistemic game play. *International Journal of Gaming and Computer-Mediated Simulations*, 1(1), 36–52.
- Bagley, E. A. & Shaffer, D. W. (2010). Stop talking and type: Mentoring in a virtual and face-to-face environmental education environment. *International Journal of Computer-Supported Collaborative Learning*.
- Barab, S. A., Scott, B., Siyahhan, S., Goldstone, R., Ingram-Goble, A., Zuiker, S. & Warrant, S. (2009). Transformational play as a curricular scaffold: Using videogames to support science education. *Journal of Science Education and Technology*, 18(3), 305–320.
- Brown, A. L. (1992). Design experiments: Theoretical and methodological challenges in creating complex interventions in classroom settings. *Journal of the Learning Sciences*, 2(2), 141–178.
- Chesler, N. C., Arastoopour, G., D'Angelo, C. M., Bagley, E. A. & Shaffer, D. W. (2013). Design of professional practice simulator for educating and motivating first-year engineering students. *Advances in Engineering Education*, 3(3), 1–29.
- Chesler, N. C., Ruis, A. R., Collier, W., Swiecki, Z., Arastoopour, G. & Shaffer, D. W. (2015). A novel paradigm for engineering education: Virtual internships with individualized mentoring and assessment of engineering thinking. *Journal of Biomechanical Engineering*, 137(2).
- Clark, D. B., Nelson, B., Sengupta, P. & D'Angelo, C. M. (2009). *Rethinking science learning through digital games and simulations: Genres, examples, and evidence*. Proceedings of the National Academies Board on Science Education Workshop on Learning Science: Computer Games, Simulations, and Education. Washington, D.C.: National Academies Press.
- Cobb, P., Confrey, J., Lehrer, R. & Schauble, L. (2003). Design experiments in educational research. *Educational Researcher*, 32(1), 9–13.
- Confrey, J. (2006). The evolution of design studies as methodology. In R. K. Sawyer (Ed.), *The Cambridge handbook of the learning sciences* (pp. 135–152). New York, NY: Cambridge University Press.
- Dieterle, E. (2009). Neomillennial learning styles and River City. *Children, Youth and Environments*, 19(1), 245–278.
- Dow, S., MacIntyre, B., Lee, J., Oezbek, C., Bolter, J. D. & Gandy, M. (2005). Wizard of Oz support throughout an iterative design process. *Pervasive Computing*, 4(4), 18–26.
- Graesser, A. C. (2013). Evolution of advanced learning technologies in the 21st century. *Theory into Practice*, 52(S1), 93–101.
- Graesser, A. C., Conley, M. W. & Olney, A. (2012). Intelligent tutoring systems. In K. R. Harris, S. Graham, T. Urdan, A. G. Bus, S. Major & H. L. Swanson (Eds.), *APA educational psychology handbook, Vol. 3: Application to learning and teaching* (pp. 451–473). Washington, D.C.: American Psychological Association.
- Graesser, A. C., D'Mello, S. K., Hu, X., Cai, Z., Olney, A. & Morgan, B. (2012). AutoTutor. In P. McCarthy & C. Boonthum-Denecke (Eds.), *Applied natural language processing: Identification, investigation, and resolution* (pp. 169–87). Hershey, PA: IGI Global.
- Graesser, A. C. & McNamara, D. S. (2012). Automated analysis of essays and open-ended verbal responses. In H. Cooper, P. M. Camic, D. L. Long, A. T. Panter, D. Rindskopf & K. J. Sher (Eds.), *APA handbook of research methods in psychology, Vol. 1: Foundations, planning, measures, and psychometrics* (pp. 307–325). Washington, D.C.: American Psychological Association.
- Graesser, A. C., Penumatsa, P., Ventura, M., Cai, Z. & Hu, X. (2007). Using LSA in AutoTutor: Learning through mixed initiative dialogue in natural language. In T. K. Landauer, D. S. McNamara, S. Dennis & W. Kintsch (Eds.), *Handbook of latent semantic analysis* (pp. 243–262). Mahwah, NJ: Erlbaum.
- Grishman, R. & Kittredge, R. (1986). *Analyzing language in restricted domains: Sublanguage description and processing*. Hillsdale, NJ: Erlbaum.
- Halpern, D. F., Millis, K., Graesser, A. C., Butler, H., Forsyth, C. & Cai, Z. (2012). Operation ARA: A computerized learning game that teaches critical thinking and scientific reasoning. *Thinking Skills and Creativity*, 7, 93–100.
- Hickey, D., Ingram-Goble, A. & Jameson, E. (2009). Designing assessment and assessing design in virtual educational environments. *Journal of Science Education and Technology*, 18(2), 187–209.
- Hilton, M. (2008). *Research on future skills demands: A workshop summary*. Washington, D.C.: National Academies Press.
- Honey, M. A. & Hilton, M. H. (2011). *Learning science: Computer games, simulations, and education*. Washington, D.C.: The National Academies Press.

- Hu, X., Cai, Z., Han, L., Craig, S. D., Wang, T. & Graesser, A. C. (2009). AutoTutor Lite. In *Proceedings of the 2009 conference on Artificial Intelligence in Education: Building learning systems that care: From knowledge representation to affective modelling* (p. 802). Amsterdam, Netherlands: IOS Press.
- Kelly, A., Lesh, R. A. & Baek, J. Y. (2008). Handbook of design research methods in education. New York, NY: Routledge.
- Ketelhut, D. J., Dede, C., Clarke-Midura, J. & Nelson, B. (2006). A multi-user virtual environment for building higher inquiry skills in science. *American Educational Research Association Annual Conference*. San Francisco, CA.
- Koedinger, K., Aleven, V., Heffernan, N., McLaren, B. & Hockenberry, M. (2004). Opening the door to non-programmers: Authoring intelligent tutor behavior by demonstration. In V. Aleven, J. Kay & J. Mostow (Eds.), *Intelligent tutoring systems* (pp. 162–174). Berlin, Germany: Springer.
- Koedinger, K. & Mitrovic, A. (2009). Preface: Authoring intelligent tutoring systems. *International Journal of Artificial Intelligence in Education*, 19(2), 103–104.
- Lave, J. & Wenger, E. (1991). *Situated learning: Legitimate peripheral participation*. Cambridge, MA: Cambridge University Press.
- Major, N., Ainsworth, S. E. & Wood, D. . (1997). REDEEM: Exploiting symbiosis between psychology and authoring environments. *International Journal of Artificial Intelligence in Education*, 8, 317–40.
- McNamara, D. S., Levinstein, I. B. & Boonthum, C. (2004). iSTART: Interactive strategy training for active reading and thinking. *Behavioral Research Methods, Instruments, and Computers*, 36(222-33).
- Mitrovic, A. (2012). Fifteen years of constraint-based tutors: What we have achieved and where we are going. *User Modeling and User-Adapted Interaction*, 22(1-2), 39–72.
- Mitrovic, A., Martin, B., Suraweera, P., Zakharov, K., Milik, N., Holland, J. & McGuigan, N. (2009). ASPIRE: an authoring system and deployment environment for constraint-based tutors. *International Journal of Artificial Intelligence in Education*, 19(2), 155–188.
- Murray, T. (1999). Authoring intelligent tutoring systems: An analysis of the state of the art. *International Journal of Artificial Intelligence in Education*, 10, 98–129.
- Murray, T., Blessing, S. & Ainsworth, S. (2003). *Authoring tools for advanced technology learning environments: Toward cost-effective adaptive, interactive and intelligent educational software*. Berlin, Germany: Springer.
- Nash, P. & Shaffer, D. W. (2011). Mentor modeling: The internalization of modeled professional thinking in an epistemic game. *Journal of Computer Assisted Learning*, 27(2), 173–189.
- Nash, P. & Shaffer, D. W. (2013). Epistemic trajectories: Mentoring in a game design practicum. *Instructional Science*, 41(4), 745–771.
- Nelson, B. C., Ketelhut, D. J. & Schifter, C. (2010). Exploring cognitive load in immersive educational games: The SAVE Science project. *International Journal of Gaming and Computer-Mediated Simulations*, 2(1), 31–39.
- Nulty, A. & Shaffer, D. W. (2008). Digital zoo: The effects on mentoring on young engineers. In *International Conference of Learning Sciences*. Utrecht, Netherlands.
- Nye, B. D., Graesser, A. C. & Hu, X. (2015). AutoTutor and family: A review of 17 years of natural language tutoring. *International Journal of Artificial Intelligence in Education*, in press.
- Pennebaker, J. W., Booth, R. J. & Francis, M. E. (2007). LIWC2007: Linguistic inquiry and word count. Austin, TX: LIWC.net.
- Richard, K. & Lehrberger, J. (1982). *Sublanguage: Studies on language in restricted semantic domains*. Berlin: Walter de Gruyter.
- Rotherham, A. J. & Willingham, D. (2009). 21st century skills: The challenges ahead. *Educational Leadership*, 9, 16–21.
- Rupp, A. A., Gustha, M., Mislevy, R. & Shaffer, D. W. (2010). Evidence-centered design of epistemic games: Measurement principles for complex learning environments. *Journal of Technology, Learning and Assessment*, 8(4), 4–47.
- Sadler, T. D. (2009). Situated learning in science education: Socio-scientific issues as contexts for practice. *Studies in Science Education*, 45(1), 1–42.
- Sadler, T. D., Romine, W. L., Stuart, P. E. & Merle-Johnson, D. (2013). Game-based curricula in biology classes: Differential effects among varying academic levels. *Journal of Research in Science Teaching*, 50(4), 479–499.
- Schön, D. A. (1983). *The reflective practitioner: How professionals think in action*. New York, NY: Basic Books.
- Schön, D. A. (1987). *Educating the reflective practitioner*. San Francisco, CA: Jossey-Bass.

- Shaffer, D. W. (2003). When Dewey met Schön: Computer-supported learning through professional practices. *World Conference on Educational Media, Hypermedia, and Telecommunications*. Honolulu, HI.
- Shaffer, D. W. (2007). *How computer games help children learn*. New York, NY: Palgrave Macmillan.
- Sottilare, R., Graesser, A. C., Hu, X. & Holden, H. (2013). *Design recommendations for intelligent tutoring systems: Learner modeling*. Orlando, FL: Army Research Laboratory.
- VanLehn, K., Graesser, A. C., Jackson, G. T., Jordan, P., Olney, A. & Rosé, C. P. (2007). When are tutorial dialogues more effective than reading? *Cognitive Science*, 31(1), 3–62.
- Woolf, B. P. (2010). *Building intelligent interactive tutors: Student-centered strategies for revolutionizing e-learning*. Burlington, MA: Morgan Kaufmann.

SECTION IV

AUTHORING DIALOGUE-BASED TUTORS

Art Graesser, Ed.

CHAPTER 16 Authoring Conversation-based Tutors

Arthur Graesser
University of Memphis

Introduction

Conversation-based intelligent tutoring systems (ITSs) attempt to help students learn by holding a conversation in natural language. Most of the systems consist of dialogues between the human learner and the computer tutor, who take turns in the conversation. Two or more agents can also hold conversations with a learner. For example, in trialogues the learner interacts with two agents, such as a tutor and a peer, or two peers. Trialogues allow the ITS to exhibit conversation skills, for the learner to view, in addition to advancing the learning of the subject matter. Most of these dialogue-based ITSs also have external media, such as a picture, table, diagram, or interactive simulation. The designers of conversation-based ITSs need to worry about the coordination and timing of the conversational turns among the learners, agents, and dynamic external media. Designers of these ITSs also need to worry about what each agent looks like. An agent can vary from a minimalist depiction of the human persona (such as a chat message) to a very realistic depiction, such a fully embodied avatar in a virtual world.

The core of the conversation-based ITSs resides in natural language, discourse, and communication. There are three basic tasks in these conversation systems: (1) interpret the meaning of the learner's language and discourse, (2) assess how these verbal contributions might update the student model on knowledge, skills, and strategies, and (3) generate tutor dialogue moves that advance the pedagogical agenda. The authoring tools need incorporate components that accommodate all of these tasks in addition to creating the agent persona and external media. This is particularly difficult because most designers of curricula with subject matter expertise have never been trained on the mechanisms that underlie language, discourse, and communication; instead most of their training is on the subject matter, pedagogy, and curriculum.

Conversation-based ITS are not able to interpret and intelligently respond to any verbal expression that a human expresses. One reason is because much of natural language is fragmentary, vague, imprecise, ungrammatical, and filled with spelling errors. A second reason is that the computer can effectively handle only input that matches content that it anticipates ahead of time, such as expected good answers, bad answers, and misconceptions that the author specifies in the curriculum. In essence, the ITS computes semantic matches between student verbal input and the expected content in the curriculum and student model. Advances in computational linguistics and statistical models of world knowledge have impressively increased the accuracy of the semantic matches. However, the author or automated components need to specify how the expected content is represented; this requires expertise in computational linguistics, cognitive science, corpus analysis, and perhaps other fields if these representations are anything other than natural language. In an ideal world, there would be a large suite of automated utilities in the authoring tool to minimize the burden on the author. But most systems in current practice require methodical annotation of the curriculum content for semantic match computations.

The authors need to create the tutorial dialogue moves that get launched under specific conditions in response to the learner's contributions. Most conversation-based ITSs have production rules that declare what an agent says under particular conditions. For example, the tutor agent gives positive feedback ("That's correct") after the learner's verbal contribution has a high semantic match to a good answer. Or the tutor agent generates a hint if the semantic match is close but not quite high enough. Unfortunately,

computer science expertise is normally needed to set up the production rules in the production systems that intelligently generate the agents' discourse moves. The rules get particularly tricky when there are many conditions to check, when there are links to dynamic external media, and when the timing of discourse move production is important. Again, one option is for the author to specify this content meticulously. Visualizations such as Excel tables and chat maps can sometimes help the author. Another approach is to copy previous production systems that are successful and modify them for specific content. More advanced methods include machine learning and crowd sourcing methodologies to minimize the burden on the author.

Chapters

The chapter by Cai, Graesser, and Hu describes its AutoTutor Script Authoring Tool (ASAT) that is used to develop content for AutoTutor. AutoTutor helps students learn subject matters (e.g., science, technology, engineering and mathematics (STEM) topics) and skills (e.g., comprehension, scientific reasoning) by holding a conversation in natural language with conversational agents. The conversations often refer to components in external media, such as pictures, diagrams, video, and virtual reality scenarios. Agents can converse with each other in addition to the human learner. The ASAT tool needs to specify the characteristics of the expected learner input (ranging from mouse clicks to natural language), the alternative messages produced by the agents under particular conditions, the external media, and the flow of the conversation. The AutoTutor Conversation Engine (ACE) is responsible for evaluating the student input, updating the learner's performance scores, selecting a new set of conversational messages, and sending all of this back to the learning system. The learner's verbal input is compared with expected input through semantic matching algorithms that can accommodate language that is often ungrammatical, vague, imprecise, and filled with spelling errors. This chapter describes some new visualization tools in ASAT-V that help the author create the content and production rules in such complex and multifaceted conversations.

The chapter by Ward and Cole describes the processes and tools for authoring content in an ITS called My Science Tutor (MyST). This virtual science tutor engages children in spoken dialogues to help them construct explanations of science phenomena that are presented in illustrations, animations, and interactive simulations in a curriculum that incorporates science standards (Full Option Science System). The chapter describes an iterative process of recording, annotating, and analyzing logs of natural language from sessions with students, which, in turn, update the automated tutor model. A major challenge in all conversational systems lies in representing and extracting the semantics of student language, which, in turn, guides selecting tutor actions. The chapter describes some computational linguistics tools, natural language corpora, and machine learning methods that help the author create content for new material.

In the chapter by Johnson, there is a focus on virtual role-play simulations in which learners perform roles similar to what they would perform in real life. Virtual role play is a category of training that is particularly well suited to interpersonal skills. It has been applied to training foreign language, cross-cultural skills, negotiation, motivational interviewing, and customer service. Processes and tools are described for creating such simulations. The development process has distinct phases, including background sociocultural research, instructional design, scenario authoring, media production, and quality assurance. The authoring tools need to handle the creation of agents, social scenarios, conversational discourse, and other dimensions of a rich social environment. Johnsons authoring tools are selected or developed to handle all phases and attributes of these simulations. Multiple types of expertise are needed in those who author these learning environments so it is unlikely that a single author could handle all dimensions of these simulations. Expertise in the natural language component is distinctively different than the other levels, but one important message is that cultural sensitivity must be integrated with the language and dialogue.

The chapter by Olney, Brawner, Pavlik, and Koedinger describes some new trends in the authoring process that can potentially improve the quality, speed, and cost of ITS authoring. These new alternatives have additional layers of automation that attempt to reduce some of the authoring tasks, and in some cases, make the authoring tasks invisible. For example, instead of an author hand-authoring a production system (i.e., what should the computer do when there are different student inputs), in systems like SimStudent the author tutors a machine learning system that learns the production system from scratch. In the BrainTrust system for conversational tutoring, novices do some authoring, the computer generates additional expressions automatically, and other novices check the work to ensure quality. In advanced component-based authoring, previous components from a learning registry are reused and new combinations of components are assembled; these candidate learning objects can be modified to fit constraints of a new application. In these examples, content can be more quickly authored by interacting with a simulation, generating content automatically, reusing content from previous applications, and crowd sourcing. These new approaches are promising because expertise in authoring and subject matter knowledge is typically limited and also requires exceptional analytical skills in more complex learning environments.

Implications for the Generalized Intelligent Framework for Tutoring (GIFT)

The four chapters provide both general and specific recommendations for GIFT's suite of authoring tools for conversation-based tutors. GIFT has already developed one conversation-based tutor when it used the AutoTutor-Lite authoring tool to integrate AutoTutor with Physics Playground, a learning environment with multimedia, animation, and game features. This is an important beginning, but GIFT will need to be expanded to build the more complex conversation-based ITS that have been covered in this section.

One issue periodically raised addresses the degree or depth of integration between the language/discourse components and the subject matter knowledge/skills to be mastered. Should there be independent components, loose coupling, or tight integration? The different approaches have implications for the authoring tools in addition to the information that ends up being stored in the learner model (as in TinCan, the Learner Record Store, or other GIFT solutions). A tight integration will result in a more complex authoring tool and student model that incorporates language-discourse-knowledge-skill configurations. A tight integration allows new discoveries in data-mining explorations to improve the conversation-based ITS. However, it will also end up being more complex to author, more difficult to specify production rules, and a more detailed inventory of learning objects, all of which aggravates the analytical challenges for the author.

A second major issue addresses how GIFT can incorporate a suite of visualization tools, lexicons, corpora, and other facilities that are routinely used in computational linguistics. For example, the projects of AutoTutor (Cai et al.) and Virtual Role-Play Simulation (Johnson) both desired a chat map visualization facility in the authoring tools. The authoring tools in all of the projects in this section would benefit from standard computational linguistics resources, such as the WordNet lexicon, corpora in the Linguistic Data Consortium, frequently used syntactic parsers, regular expression generators, and machine learning tools for natural language. These would need to be integrated in the authoring tool so the author can quickly test the fidelity of a candidate linguistic or symbolic expression being annotated. Agent tool kits would be needed to quickly test out how an agent's spoken message, facial expression, or message is rendered. World knowledge representations, such as latent semantic analysis and semantic networks, are also periodically needed. GIFT needs to expand its library of facilities in computational linguistics, discourse, agent technologies, and world knowledge representations.

A third issue is to find ways for GIFT to automate aspects of the authoring process. The reuse of existing successful components, modules, and lessons is encouraged by everyone and fits perfectly with the GIFT

philosophy. So relevant existing components need to be discovered for a particular lesson and then reused and repurposed on the spot. A good authoring tool would essentially be good at modding a similar lesson. Some deep thought is needed on how to expand GIFT to include the SimStudent, BrainTrust, and crowd-sourcing approaches to iteratively improve the quality of the authored content, as was discussed in the Olney et al. chapter, and to some extent, in the chapter by Ward and Cole.

CHAPTER 17 ASAT: AutoTutor Script Authoring Tool

Zhiqiang Cai, Arthur Graesser, and Xiangen Hu
University of Memphis

Introduction

AutoTutor is a class of intelligent tutoring systems (ITSs) that helps students learn by holding a conversation in natural language (Graesser et al., 2004, 2012; Nye, Graesser & Hu, in press). AutoTutor's intelligent conversation framework has been integrated into many learning systems that range from tutorial dialogues on science, technology, engineering, and mathematics (STEM) topics (such as computer literacy, and physics) to trialogues (i.e., two agents and a human) on critical thinking and reading comprehension (Graesser, Li & Forsyth, 2014; Millis et al., 2011; Halpern et al., 2012; Forsyth et al., 2013). Examples of trialogues under construction (<https://www.youtube.com/channel/UCGoWLJj6BXZ6X2KIRLYrgZw>) can be viewed for a more concrete illustration of the nature of these conversations. AutoTutor is an advanced conversation framework that can be used to generate conversation scripts and be integrated into most learning systems. AutoTutor takes the learner's typed verbal contributions, speech, and actions as input and accommodates events in different media to trigger or change paths of conversations. It also sends commands to the learning system for execution, such as presenting pictures and launching scenarios (Cai, Feng, Baer & Graesser, 2014).

There are many steps in composing an AutoTutor conversation with one or more computer agents and a human learner. Authoring an AutoTutor conversation includes preparing spoken contributions for each computer agent, specifying conditions at which a speech is delivered, determining the points at which human learners' responses and/or environmental events are expected, formulating scores that can be used to track learners' performance, designing pedagogical strategies, creating commands that make changes to learning system parameters, and so on (Cai, Hu & Graesser, 2013). Because of this complexity, an AutoTutor script authoring process usually requires collaborative work by domain experts, language experts, learning experts and software developers. Domain experts use the tool to construct learning content in terms of agent questions and expected learner responses. Language experts revise the content of the dialogue moves to accommodate targeted learners and their possible responses. Learning experts design student models and pedagogical models. Software developers specify interaction constraints and develop interactive media units.

The AutoTutor Script Authoring Tool (ASAT) is a tool we developed to facilitate the process of authoring AutoTutor content. In this chapter, we present ASAT-V, the visualized version of ASAT. ASAT-V uses *graphical shapes* to represent agents' spoken contributions, questions, answers, world events, and system actions. Semantic cues and student performance scores are stored in shape data. Conversation rules are represented by directional connections from shape to shape. Pedagogical strategies are represented by partial flowcharts, which can be reused. The tool also integrates utility modules to help authors validate, test and refine scripts. In this chapter, we first give an overview of the AutoTutor framework and the major components that make AutoTutor work. We then describe ASAT-V and the AutoTutor shapes that are used to compose visual scripts. The chapter ends with suggestions for developing conversation modules and authoring tools for ITSs.

AutoTutor Framework

AutoTutor provides a framework to integrate intelligent conversations into learning systems. A learning system can start an AutoTutor conversation session by loading an AutoTutor script to the AutoTutor Conversation Engine (ACE). ACE sends messages to the learning system, including agent spoken utterances and system commands. An AutoTutor conversation usually starts with spoken turns by computer agents, together with background changes on the computer screen, such as page turning, video playing, image changing, and text highlighting. At particular points, the system stops and waits for the learner's input, in the form of speech, text, or action. The learner's input is then sent by the learning system to ACE. ACE is responsible for evaluating the input, setting learner's performance scores, selecting a new set of conversational messages, and sending all of them back to the learning system. The process repeats until the conversation session ends.

An example illustrates this process. Suppose a learning system is showing a video to a learner to review a lesson about the use of punctuations. While the video is playing, a tutor agent is talking about the video. The video pauses at a certain time and the tutor asks the learner questions about the learning material. The learning system starts this process by loading a script to ACE. After the script is successfully loaded, ACE sends to the learning system the following actions to execute:

- (1) System : LoadVideo : <https://www.youtube.com/watch?v=wTs6Q8Cs5AY>
- (2) System : SetPauseTime : 00:00:30
- (3) System : StartVidio
- (4) Tutor : Speak: Now, let us have a review of lesson 5. In this lesson, we learn about the use of punctuations. Please watch this video carefully and pay attention to how punctuations help reading.

When the learning system gets these four actions from ACE, the system first loads the video from the given URL. Then the system sets a timer for 30 seconds and starts to play the video. The tutor talks while the video is playing. Notice that ACE may send many different types of actions to the learning system. The learning system is responsible for interpreting the actions. AutoTutor authors have to collaborate with learning system developers on what actions are executable and how they should be executed.

When the video pauses at the specified time, the learning system sends to ACE a message that the video is paused. ACE then makes decisions on what to do next and sends a new set of actions to the learning system:

- (1) Tutor : Speak : OK. This video talked about punctuation definition signals. What are they?
- (2) System : WaitForInput : 20

The learning system then delivers the speech and waits 20 seconds for the learner to enter a response. Suppose the learner entered "They are dashes and commas." The response then is sent to ACE. ACE then analyzes the response and figures out that the response is a partial answer. ACE then sends out a new set of actions:

- (1) Tutor : Speak : Wonderful! You got some of them. Can you say more?

(2) System : WaitForInput : 20

This process continues until the conversation session ends.

The above example involves the conversation engine ACE, the conversation script, and semantic analysis. In order to understand the authoring process of AutoTutor conversation, it is important to know the following main features in AutoTutor framework.

Script

An AutoTutor script defines all elements in a conversation session, including agents, commonly used speech acts, spoken messages of agents, questions, answers, and so on. Conversation rules are also specified in the script, which is implemented in ASAT-V by connecting script shapes with single directional lines.

Natural Language Input Assessment

Evaluating natural language input in AutoTutor is accomplished in two steps. The first step is to determine the type of speech act of the learner input, such as definitional question (“What is X?”), yes/no question (“Is X?”), request (“Can you show me another page?”), meta-cognition (“I have no idea about that.”), meta-communication (“Can you repeat that?”), statement, and so on (Samei, Li, Keshtkar, Rus & Graesser, 2014). The second step is to identify semantic units in the input and match the input with prepared target units. For example, if the input is a definitional question, then AutoTutor will find for what concept the learner needs a definition. If the input is an answer to a question an agent asked, AutoTutor will match the input with prepared answers to the question. AutoTutor uses two ways to accomplish semantic matching. One is using regular expression matching. A regular expression is simply a string pattern that is used to check whether or not a target string has matches to the pattern. For example, if the expected answer is “they are dashes and commas,” the regular expressions could be {“\bdash”, “\bcomma”}, where “\b” indicates “word boundary”. For each target answer, a set of regular expressions is created to represent the key parts of the answer. The proportion of matched regular expressions is used as regular expression matching score as part of the semantic evaluation. Another one is latent semantic analysis (LSA) (Hu et al., 2007, Cai et al., 2011). LSA represents the meaning of text units by vectors of statistical semantic features. The cosine value between two vectors (the student input and an expected answer, both in natural language) is used as another part of semantic evaluation.

Student Models and Pedagogical Models

Student models keep track of students’ performance. The data from student model are used by pedagogical model for tutoring strategy selection. What should be used as variables for student modeling is still an unanswered question (Graesser, 2013). AutoTutor allows authors and learning system designers to create customized variables to track students’ learning process and performance. The customized student model is implemented as a set of name-value pairs, together with a few functions to do score operations, such as initializing scores, adding scores, etc. The tutoring strategies in AutoTutor are implemented as conversation patterns, such as vicarious learning, expectation-misconception tailored tutoring, teachable agent, etc. (Cai et al., 2014). In ASAT-V, conversation patterns are implemented as partial flowcharts, which can be reused in script authoring.

Communication between AutoTutor Conversation Engine and Learning Systems

When AutoTutor conversation is integrated into a learning system, the conversation engine needs to communicate with the learning system constantly. The conversation engine needs to know what is happening in the learning environment in order to choose the next step to move on. In the example above, when the video is paused, a “video paused” message is sent from the learning system to the conversation engine and the conversation engine decides that the next step is to ask the learner a question. AutoTutor allows learning system to send messages about what happens in the learning system as “world events.” World events are simply labels that are pre-negotiated between learning system developers and AutoTutor rule designers. In the example above, “VideoPaused” could be a label that is used to indicate the pause of any video in the learning environment. The learning system always sends such a world event to AutoTutor engine when a video is paused. It is up to the rule designer to decide what to do with this event. Therefore, a world event list needs to be shared by the learning system developers and AutoTutor rule designers, so that the system developers know what can be sent and the rule designers know what can be expected.

ACE: AutoTutor Conversation Engine

ACE is a web service that interprets AutoTutor scripts and communicates with learning systems. ACE is currently implemented as a RESTful web service, which can be easily integrated into any system.

With the above features, AutoTutor is capable of taking care of the conversation part of learning systems. However, authoring AutoTutor scripts is never an easy task. The AutoTutor research group at University of Memphis has worked for more than a decade to develop tools to help the script authoring process. ASAT-V, a visualized authoring tool, is the latest development.

ASAT-V

ASAT-V is a windows desktop application that requires .Net Framework 4.5 and Microsoft Visio 2013. This tool is used to define computer agents, view Visio flowcharts, and test scripts.

Figure 1 shows a screen shot of ASAT-V. On the menu strip, there are only two menu items. The “FILE” menu is used for creating a new project or open an existing project. A project is a set of Visio flowcharts. Developers can select “Sample Project” in the File menu to open the sample project. The sample project folder is in the installation directory of ASAT-V. Authors can make a new copy of the sample project folder to start a new project. The “HELP” menu is used to access an online help document, which is updated when new release of the tool comes out.

The left panel of the tool is a list box that contains the flowchart names of the opened project. Authors can click an item to select a flowchart to work on.

The right panel contains six tab pages, labeled as “Visio,” “Shape Data,” “Question,” “Test,” “Agent,” and “Speech Acts,” respectively. “Visio” page contains a standard Visio Viewer that displays a selected Visio flowchart. This page is connected to Visio 2013. When editing is needed, an author can press the Visio editing button to open the Visio script in Visio 2013. The flowchart shown in Figure 1 contains different Visio shapes (circles, rectangles, lines, etc.). In addition to the look of each shape, each shape type contains a set of attributes that are specifically defined in ASAT-V. We explain the data defined for every shape type in later sections. The tab page “Shape Data” lists all shapes in the selected script and displays the associated data of a selected shape. Authors can review the data shape by shape and see if there is any error. The tab page “Questions” is actually created for answer evaluation. The questions in a

selected flowchart are listed in this page. When a question is selected, all prepared answers associated with the selected question are then displayed. There is an input box on the page for an author to type in an answer to the question and see how much an answer can match each prepared answer. Authors can use this tab page to set the thresholds for semantic assessment. The “Test” page is for script testing. Authors can simulate a student’s interaction to a selected script by submitting expected textual responses or world events to find out if the system performs as desired. The “Agent” page defines computer agents. An author can find all defined agents in a dropdown menu. When an item in the menu is selected, the information about the selected agent will be displayed and can be edited. New agents can be added by typing in the text field of the dropdown menu. The tab page “Common Speech Acts” defines commonly used speech acts using regular expressions. The definition of agents and speech acts are for all flowcharts in a selected project. Therefore, the agents and speech acts are not defined in any of the flowcharts.

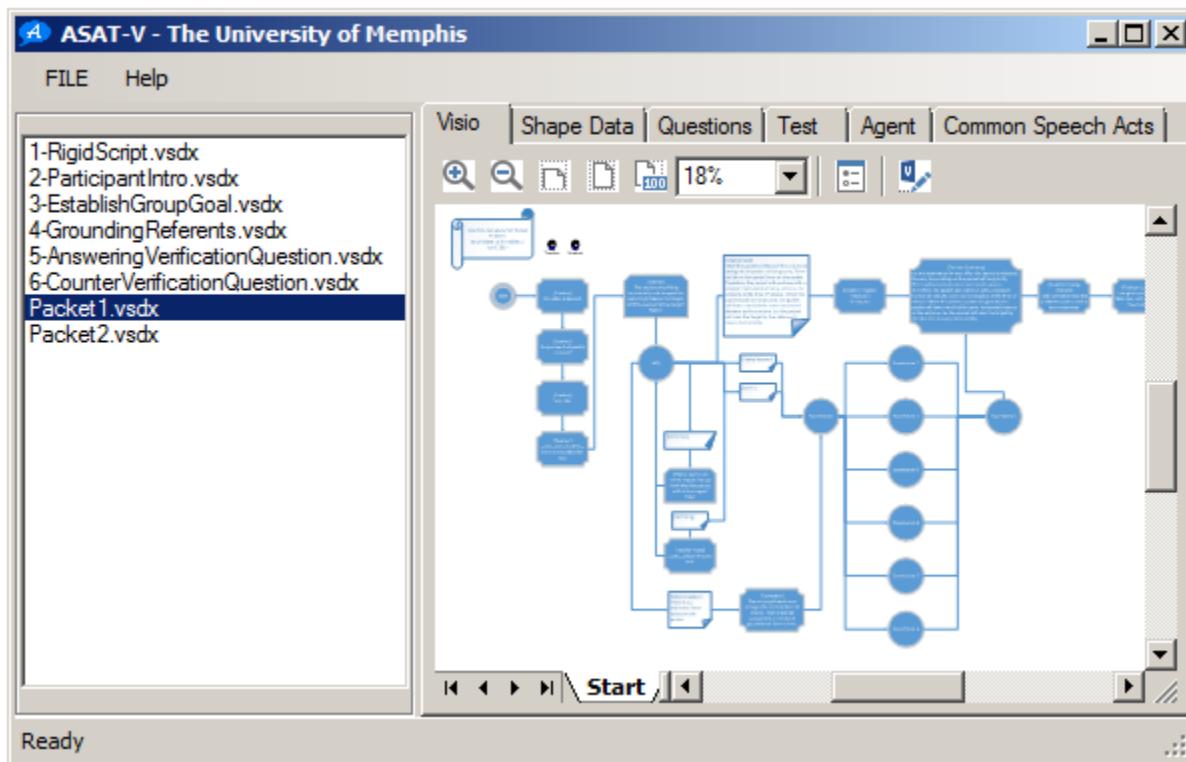


Figure 4. ASAT-V

In the next section, we describe all AutoTutor shapes, their text and data fields, and their use in constructing the scripts. Although these shapes are currently implemented in Visio 2013, it is possible to implement them in other drawing tools that store accessible shape data.

Autotutor shapes

Figure 2 shows an AutoTutor script flowchart drawn in Visio 2013. The flowchart is an AutoTutor conversation pattern called “Greeting.” The conversation begins with a greeting “Hello!” from a teacher. Then the system waits for user’s response. If the user says anything, the teacher says, “Terrific! We’ve connected.” The conversation then ends. If the user is silent, the teacher says, “Are you there, user?” Then the system waits for the user to respond. If the user is silent again, the teacher says, “Too bad.” Then the conversation ends.

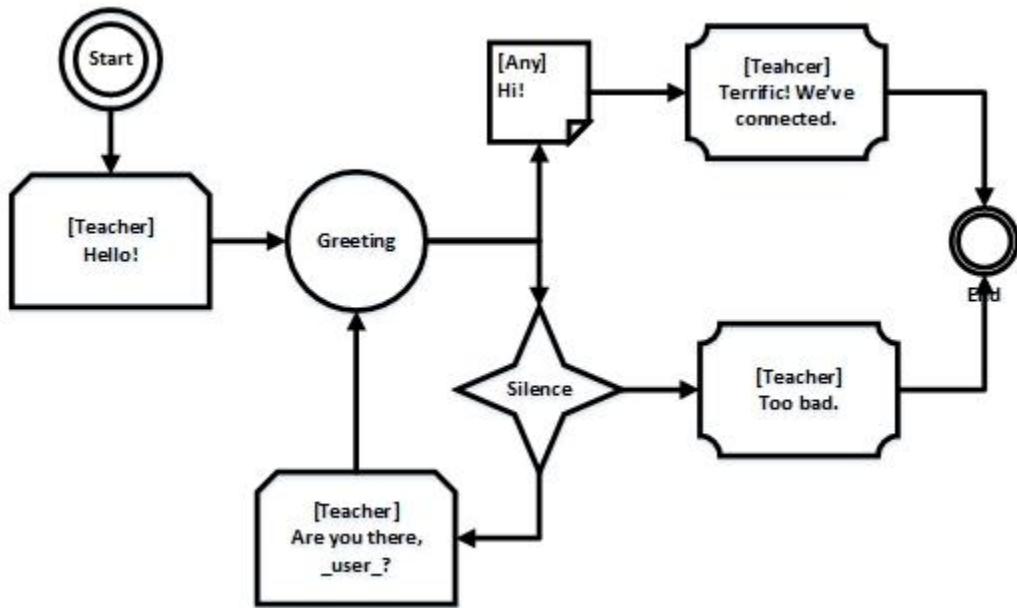


Figure 2. Script flowchart for Greeting

As one can see in Figure 2, the conversation script is represented by connected AutoTutor shapes. An AutoTutor shape refers to a visual shape and its associated data. Every shape has a type name and a text field. Any text inside a pair of brackets is considered commentary text and is ignored by ACE in the interpretation. Currently, ten shape types have been defined for AutoTutor script. Figure 3 shows these ten shapes as the AutoTutor stencil in Visio 2013. We explain these shapes below in more detail.

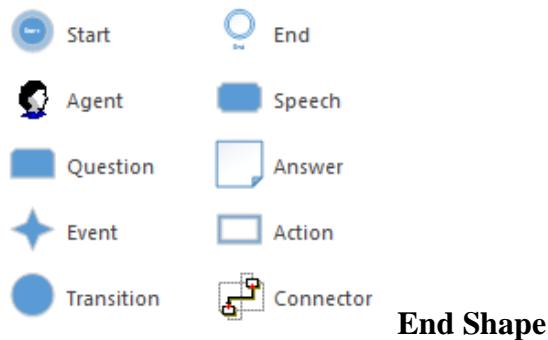


Figure 3. AutoTutor stencil in Visio 2013

Start Shape

A *Start* shape represents the beginning of a conversation. The text should be “Start.” Although, the text field of this shape is not really used in the script interpretation, using an explicit “Start” helps to make the

flowchart clear. No shape data are defined for *Start* shape. Each script should have one and only one *Start* shape, which should point to at least one other shape. Usually, *Start* is the first shape to put to a script flowchart.

An *End* shape represents an end of a conversation. A script must have at least one *End* shape. Multiple *End* shapes are allowed. The text field of the *End* shape helps to indicate the ending path of a conversation. Therefore, the text on different *End* shapes can be different, such as “End-1,” “Good-End,” “A-End,” etc. “Score” is the only data field in an *End* shape. Authors may specify this score for a shape in a flowchart to indicate the learner’s performance at the specific ending.

Agent Shape

AutoTutor agents are not defined in the flowchart, as we already explained earlier. However, we created *Agent* shape type for authors to put agent names together with a script flowchart to show what agents are used in the flowchart. *Agent* shape is not required in a script and will not be interpreted by ACE. The agents used in the flowchart are defined in the *Agent* tab page of ASAT-V.

Speech Shape

The *Speech* shape represents the conversational contribution of an agent. The text field is the text form of the speech content, together with optional commentary note in brackets. While the commentary note is arbitrary, it is recommended that, for a *Speech* shape, the note contains the agent information, such as “Teacher,” “Peer Student,” etc. The text form of the speech content can be displayed to the learner. There are two data fields in *Speech* shape. One is “*Agent*.” The value of “*Agent*” field is an ID created separately (see section ASAT-V). The other data field is “*Speech*.” The value of this field is optional. Authors may use this field for one of the two different purposes: (1) to create a tagged speech string for on-the-fly speech generation or (2) to store a label or URL of a stored speech. The stored speech could be recorded human speech or a pre-generated speech from a text-to-speech (TTS) engine. If the speech data are empty, the displayable text can be used to generate speech. When a conversation moves to this shape, the agent will deliver the speech. Once the speech is done, the flow moves to the next shape.

Question Shape

The *Question* shape has the same data fields as the *Speech* shape. However, this shape is always followed by answer shapes or transition shapes (see below). When the conversation moves to this shape, whether or not the question shape will be asked depends on if there is a good answer of the question that has already been answered by the learner. If the learner has already answered the question, this shape will not be selected and the conversation will move to other paths. One important issue for authors to keep in mind is that, alternative paths should be available when a question shape is not selected, so that the conversation always has a path to go.

Answer Shape

The *Answer* shape represents a possible answer that a learner may give to a preceding question. The text field of this shape is a sample answer of the type. There are several data fields in *Answer* shape, as described below:

- **AnswerType:** Answer type can be any arbitrary string. However, there are a few reserved types, including “Good,” “Bad,” “Irrelevant,” “Undetermined,” and “Blank.” These types have special interpretations in ACE and should be used correctly.
 - A “Good” answer (Figure 4) is a correct and complete answer to the question. If this answer is matched with the learner’s previous input, then the question associated with this answer will not normally be asked because the content is already covered. If, for some reason, one wants to ask the question anyway, that can be accomplished by not specifying any answer with the type as “Good.”

SHAPE DATA - ANSWER	
AnswerType	Good
RegEx	\bforce\b, \bgravity\b \bvertical\b \bdc
RegExThreshold	0.70
LSAThreshold	0.70
Score	+10
Sample1	
Sample2	
Sample3	
Sample4	
Sample5	
ResponseType	Global

Figure 4. Data for the Answer shape

- A “Bad” answer represents a typical bad answer that a learner may usually give. In addition to help determining the conversation path, “Bad” answer also helps to determine whether or not an answer is “Irrelevant” or “Undetermined.”
- An answer is “Irrelevant” if it does not match any “Good” answer or “Bad” answer.
- An answer is “Undetermined” if it matches at least one “Good” answer and one “Bad” answer.
- “Blank” answer is an answer without any word.
- **RegEx:** The value of this field is a set of regular expressions. Each regular expression represents the string pattern of a part of the answer. This field is used to assess a learner’s answer by regular expressions. The proportion of the matched regular expressions is the learner’s regular expression match score.
- **RegExThreshold:** This field is a value between 0 and 1, indicating the minimum regular expression score for an answer to be considered matched by regular expression.

- **LSAThreshold:** This field is a value between 0 and 1, indicating the minimum LSA match value for an answer to be considered matched by LSA. The LSA score is computed by comparing a learner's answer to the answer in the text field and the "Sample" fields (see below) of the answer shape. The largest cosine value of all comparisons is taken as the final LSA match score.
- **Score:** This field is a number to indicate a score that a learner should receive if this answer is matched by a regular expression or LSA.
- **SampleN:** The sample fields (Sample1, Sample2, ...) are possible answers of this type. These samples are used to compute LSA scores. The number of sample answers is not limited and an author can put as many samples as desired. The samples may come from real student responses after the script has been used. In this way, AutoTutor can learn from learners and improve its performance over time.
- **ResponseType:** The response type could be "Global" or "Local." This is used in nested questions. Usually, an answer to a main question or problem to solve is "Global" and an answer to a hint or prompt is "Local."

Event Shape

Event shape is used to integrate AutoTutor conversation with external environment. This shape is used when an external event is expected. An external event can be an action from the learner, such as a mouse click, a choice selection, etc. It can also be a system event, such as a scenario is loaded, a certain time has elapsed, and so on. *Event* shape data has an "Agent" field and a "Score" field. "Agent" indicates the source of the event, from learner or system. "Score" is a performance score assigned to the learner when this event is matched. The text field of this shape is the label of the event. When the label of any external event matches the text field of the shape, this event is considered matched.

Action Shape

The *Action* shape is used to send a sequence of actions to the system. There is only one "Name" field in the shape data. Authors can put a sequence of lines in the text field of the shape. Each line is of the form "Agent:Act:Data." An example line could be "System:Wait:30," meaning that the system should wait for 30 seconds. When this shape is encountered, ACE will send all actions to the external environment for execution. Authors have to negotiate with external environment developers to get a list of executable acts and associated data.

Transition Shape

The *Transition* shape does not have any data field. However, it plays a very important role in simplifying the structure of the flowchart. What an author should know is that all *Transition* shapes with the same text are considered "identical" in the flowchart. For example, in Figure 2, two shapes point to the "Greeting" shape and the "Greeting" shape points to two other shapes. If there is another *Transition* shape in the flowchart with the same text "Greeting," then that shape will also be considered as connected with those four shapes in the same way.

Connector Shape

The *Connector* shapes connect other shapes together to form a conversation flowchart. A connector shape is a single directional line that connects two shapes, indicating a move from one shape to another shape. A *Connector* shape has three data fields: “Priority,” “Frequency,” and “MaxVisit”. These three fields play important roles in controlling the conversation flow. We explain each of them in detail below:

- **Priority:** Priority is a positive integer (1,2,3,...) indicating the priority of a path. A value 1 indicates the highest priority. A shape may point to multiple shapes. ACE will consider the paths according to the priority. For example, in Figure 2, the *Transition* shape “Greeting” points to two shapes, an answer shape “Hi!” and an event shape “Silence.” The connector to “Hi!” has a priority 1 and “Silence” has a priority 2. When ACE selects a path from “Greeting,” it will first match the answer shape “Hi!” If the learner greets back, that path will be selected. Otherwise, it will consider the event “Silence.”
- **Frequency:** This field is a positive number. This number is used to set a selection probability for paths of same priority. The selection probability of a path is the frequency of that path divided by the sum of frequencies of all possible paths coming out from the same shape. Paths will be randomly selected with the given probability distribution.
- **MaxVisit:** This field is a positive number, indicating the number of times a path can be chosen. This value is used to terminate a loop. For example, in Figure 2, the connector from the “Silence” shape to the question shape “Are you there, _user_?” has MaxVisit = 1. Therefore, that path can be selected for only once. Otherwise, the system may keep asking “Are you there, _user_?” forever if the user keeps silent.

The above shapes are used to compose AutoTutor script flowcharts. With the help of the *Transition* shape, flowcharts can be drawn on multiple pages and connected by common *Transition* shapes. Step-by-step tutorials are available. Authors can click on the “Help” menu on ASAT-V to access online tutorials.

While currently these shapes are implemented in ASAT-V as Visio shapes, they can be implemented in any drawing tools that has the following features:

- Shapes can be customized;
- Each shape can be associated with a set of customized properties;
- Shapes can be connected by connector shapes to form flowcharts;
- One complete flowchart can be split into multiple pages; and
- The flowcharts can be exported as xml files.

Conclusion

As a generalized intelligent framework for tutoring, GIFT needs to include intelligent conversations. Unfortunately, creating intelligent conversations is a very complex process. AutoTutor conversation framework makes it possible to seamlessly integrate conversations into learning systems. When authoring conversations, the most challenging task is to set up conversation rules. The visualized authoring tool, ASAT-V, makes the rules visible and greatly reduces the complexity of the authoring process. Thus,

visualized conversation authoring tools like ASAT-V are important components of GIFT framework. To close this chapter, we give the following list of suggestions on general intelligent conversation modules and authoring tools for intelligent conversations:

- (1) Conversation modules should have good communication channels with learning systems.
- (2) Conversation modules should have flexible student model so that student's learning process and performance can be easily integrated into the conversation.
- (3) Conversation modules should have fast and high quality natural language processing (NLP) support. It is the best that the conversation module allows NLP plug-ins.
- (4) Conversation script authoring should have graphical rule editing tools.
- (5) Conversation authoring tools should have good validation and test utility.

References

- Cai, Z., Graesser, A. C., Forsyth, C., Burkett, C., Millis, K., Wallace, P., Halpern, D. & Butler, H. (2011, November). Trialog in ARIES: User Input Assessment in an Intelligent Tutoring System. In W. Chen & S. Li (Eds.), *Proceedings of the 3rd IEEE International Conference on Intelligent Computing and Intelligent Systems* (pp.429-433). Guangzhou: IEEE Press.
- Cai, Z., Forsyth, C. M., Germany, M. L., Graesser, A. C. & Millis, K. (2012). Accuracy of tracking student's natural language in OperationARIES!: A serious game for scientific methods. In S. A. Cerri & B. Clancey (Eds.), *Proceedings of the 11th International Conference on Intelligent Tutoring Systems (ITS 2012)* (pp. 629-630). Berlin: Springer-Verlag.
- Cai, Z., Hu, X. & Graesser, A. C., (2013, November). ASAT: *AutoTutor script authoring tool*. Paper presented at the meeting of the Society for Computers in Psychology, Toronto, CA.
- Cai, Z., Feng, S., Baer, W. & Graesser, A. C. (2014). Instructional strategies in trialog-based intelligent tutoring systems. In R. Sottilare, A. C. Graesser, X. Hu & B. Goldberg (Eds.), *Design Recommendations for Intelligent Tutoring Systems: Adaptive Instructional Strategies* (Vol.2)(pp. 225-235). Orlando, FL: Army Research Laboratory.
- Forsyth, C. M., Graesser, A. C., Pavlik, P., Cai, Z., Butler, H., Halpern, D. F. & Millis, K. (2013). OperationARIES! methods, mystery and mixed models: Discourse features predict affect in a serious game. *Journal of Educational Data Mining*, 5, 147-189.
- Gholson, B. & Craig, S. D. (2006). Promoting constructive activities that support vicarious learning during computer-based instruction. *Educational Psychology Review*, 18, 119-139.
- Graesser, A. C. (2013). A guide to understanding learner models. In R. Sottilare, A. C. Graesser, X. Hu & H. Holden (Eds.), *Design Recommendations for Intelligent Tutoring Systems: Learner Modeling* (Vol.1)(pp. 3-6). Orlando, FL: Army Research Laboratory.
- Graesser, A. C., D'Mello, S. K., Hu, X., Cai, Z., Olney, A. & Morgan, B. (2012). AutoTutor. In P. M. McCarthy & C. Boonthum (Eds.), *Applied natural language processing and content analysis: Identification, investigation and resolution* (pp. 169-187). Hershey, PA: IGI Global.
- Graesser, A. C., Li, H. & Forsyth, C. M. (2014). Learning by communicating in natural language with conversational agents. *Current Directions in Psychological Science*, 23, 374-380.
- Graesser, A. C., Lu, S., Jackson, G. T., Mitchell, H. H., Ventura, M., Olney, A. M. & Louwerse M. M. (2004). AutoTutor: A tutor with dialogue in natural language. *Behavior Research Methods, Instruments & Computers*, 36, 180-193.
- Graesser, A. C., Wiemer-Hastings, K., Wiemer-Hastings, P., Kreuz, R. & the Tutoring Research Group (1999). AutoTutor: A simulation of a human tutor. *Journal of Cognitive System Research*, 1, 35-51.
- Halpern, D. F., Millis, K., Graesser, A. C., Butler, H., Forsyth, C. M. & Cai, Z. (2012). Operation ARA: A computerized learning game that teaches critical thinking and scientific reasoning. *Thinking Skills and Creativity*, 7, 93-100.

- Hu, X., Cai, Z., Wiemer-Hastings, P., Graesser, A. C. & McNamara, D. S. (2007). Strengths, limitations, and extensions of LSA. In T. K. Landauer, D. S. McNamara, S. Dennis & W. Kintsch (Eds.), *Handbook of latent semantic analysis* (pp. 401-426). Mahwah, NJ: Lawrence Erlbaum.
- Hu, X., Morrison, D. M. & Cai, Z. (2013). On the use of learner micromodels as partial solutions to complex problems in a multiagent, conversation-based intelligent tutoring system. In R. Sotilare, A. C. Graesser, X. Hu & B. Goldberg (Eds.), *Design Recommendations for Intelligent Tutoring Systems: Adaptive Instructional Strategies* (Vol.2)(pp. 97-110) . Orlando, FL: Army Research Laboratory.
- Millis, K., Forsyth, C. M., Butler, H., Wallace, P., Graesser, A. C. & Halpern, D. F. (2011) Operation ARIES! A serious game for teaching scientific inquiry. In M. Ma, A. Oikonomou & J. Lakhmi (Eds.), *Serious games and edutainment applications* (pp.169-196). London, UK: Springer-Verlag.
- Nye, B. D., Graesser, A. C. & Hu, X. (in press). AutoTutor and family: A review of 17 years of natural language tutoring. *International Journal of Artificial Intelligence in Education*.
- Samei, B., Li, H., Keshtkar, F., Rus, V. & Graesser, A. (2014) Context-based Speech Act Classification in Intelligent Tutoring Systems. In S. Trausan-Matu, K. Boyer, M. Crosby & K. Panou (Eds.), *Proceedings of the 12th International Conference on Intelligent Tutoring Systems* (pp. 236 – 241). Berlin: Springer.

Chapter 18 Constructing Virtual Role-Play Simulations

W. Lewis Johnson, Ph.D.
Alelo Inc.

Introduction

Virtual role-play simulations are interactive simulations in which learners perform roles similar to what they would perform in real life. They are populated with virtual role players, i.e., non-player characters that fill out the roles in the simulation and interact with learners much as people typically do in real-life situations. Virtual role play is an important category of training that is particularly well suited to interpersonal skills. It has been applied to foreign language education (Johnson, 2010), cross-cultural skills training (Johnson et al., 2011), negotiation skills training (Kim et al., 2009), motivational interviewing (Radecki et al., 2013), and other clinical skills. Role-play scenarios are employed in sales and customer service training (Simmons, 2010). The impact of virtual role play is likely to grow as easy-to-use tools for creating such simulations become more widely available. It thus has a potentially important role to play as part of the Generalized Intelligent Frameworks For Tutoring (GIFT).

Virtual role play is inspired by training with live role players. In the military, it is common to employ people as role players in training exercises, acting as civilians and combatants, for example, see Wilcox (2012). Such training can be highly effective but unfortunately the costs involved in employing role players and the logistics involved in staging live exercises limit their use. Sometimes military members must play supporting roles in these training exercises, acting as foreign civilians or opposing forces, so they are supporting the exercise instead of receiving training themselves. Medical education also makes use of live role players in the form of standardized patients, actors trained to behave as if they have a particular medical condition (Barrows, 1993). Such training can be valuable but is limited by the availability of suitably trained actors. Role play is also very common in sales training (Robinson, 1987), but trainees often do not like it because it is not conducted in a way that is supportive and conducive to learning (Sandler Training, 2014). Best practices call for sales managers to role play the customer in such training episodes; this limits role play to times when busy sales managers are available to engage in training sessions.

Some researchers are seeking to make role-play training more convenient by moving interaction with live role players into virtual worlds. For example the Otago Virtual Hospital lets learners practice their clinical skills in a virtual world, interacting with simulated patients played by clinicians (Loke et al., 2012). Such training offers added convenience, but it still depends upon the availability of skilled role players to control the patient avatars in the virtual world. Virtual role play with virtual humans has no such constraint; trainees can practice as much as they want, whenever they want.

Alelo has been heavily involved in virtual role-play training since its inception. It draws on an extensive body of research in supporting technologies such as pedagogical agents (Johnson & Lester, in press). The development team at Alelo has broad experience in creating virtual role-play content for a variety of user groups. For example Alelo's Virtual Cultural Awareness Trainers (VCATs) have been developed to teach about culture in over 80 countries. Users of Alelo courses number in the hundreds of thousands. This gives us practical insights into the issues involved in creating, validating, and delivering virtual role-play training at scale.

This chapter provides an overview of the key capabilities of virtual role-play training systems, using deployed training systems as examples. This motivates the requirements for authoring tools. This is

followed by a discussion of authoring processes for creating and validating virtual role-play content. Authoring tools should be designed with these processes in mind. Next is an overview of available tools for authoring virtual role-play content. These include tools for creating simple role-play scenarios, tools for authoring complex role-play simulations, and emerging tools that empower trainers to construct and customize role-play training content themselves. Finally there is a discussion of future directions for this work and its implications for GIFT.

Examples of Virtual Role-Play Technologies

Figure 1 shows two example usage scenarios for virtual role play. These particular examples are intended to help learners develop their Chinese conversational skills. A common use case is shown on the left, where the learner has an on-screen avatar who interacts with on-screen virtual role players. If the user's computer or mobile device supports speech input, as in this example, the training system can employ speech understanding technology so that the virtual role players understand what the learner says and respond accordingly. This results in an engaging, immersive experience in which learners must apply their communication skills much as they would in real-life situations.



Figure 1. Learners can participate in a virtual role-play exercise by speaking and choosing actions for an on-screen avatar (left) or speaking directly with the virtual role player (right).

Advances in sensor technologies make it possible for learners to interact directly with virtual role players, instead of through an avatar. When integrated into lifelike robots, as in Figure 1 right, the virtual role player can interact with learners in the real world. This increases the realism of the role-play experience, particularly if the interface incorporates proximity sensors and gesture recognition to support mixed-initiative multimodal communication. In practice, similar software architectures can be used in both cases to control the virtual role players.

Mobile devices are also increasingly attractive as platforms for virtual role play (Johnson et al., 2012). Advances in the computing power of mobile devices make it possible to deliver interactive virtual role players on tablets and smart phones, for anywhere, anytime training. Mobile devices are increasingly equipped with cameras and other sensors that facilitate natural interaction between learners and virtual role players.

Techniques for Effective Use of Virtual Role Play

When used properly virtual role play offers a training experience that is realistic and similar to real-life interaction, but is in many ways actually superior to practice in real life. The example shown in Figure 2, taken from Alelo's VCAT Taiwan course, is a case in point. Here the learner is playing the role of an American officer on assignment in Taiwan. The learner has been invited to a formal banquet hosted by his Taiwanese counterpart. It is important that the learner make a good impression and avoid doing something embarrassing or culturally inappropriate. For example, many toasts tend to be exchanged at such dinners. How can one follow proper etiquette for exchanging toasts without getting drunk in the process? Virtual role play offers an alternative to learning the hard way by making mistakes in real-life high-stakes situations. In this example, the learner's avatar, on the left, has offered a toast saying, "Drink as you like." This gives the learner the option of offering the toast with his teacup instead of a shot glass, as his host on the right does. If the learner says or does something inappropriate, the virtual role players will react to it, so learner can see the consequences of mistakes. But since the training module is just a simulation the negative consequences of mistakes are minimal. The learner can practice multiple times until becoming comfortable saying and doing the right things at the right times. Alternative training media such as guidebooks may give learners a general understanding of the culture, but do not help learners acquire the skills they need for such situations.



Figure 2. Virtual role play lets learners practice high-stakes interactions in a safe environment.

The following are some techniques for employing virtual role play that maximize its effectiveness. Authoring tools and technologies for virtual role play should support these techniques to help developers and trainers make best use of this innovative instructional technology.

Intelligent tutoring technology, in the form of virtual coaches, can monitor learner performance in role-play simulations, provide feedback, and ensure that learners draw the right lessons from the practice experience. Figure 3 illustrates the VCAT's Virtual Coach, Erika, in action. In this example, the learner has expressed dislike for a dish that sounded unappealing, namely, sea cucumber. The Virtual Coach advises the learner to show appreciation and interest in the dishes that his host has offered. Such feedback can be very important in cross-cultural communication, where learners sometimes are not even aware when they make cultural mistakes.



Figure 3. A Virtual Coach provides scaffolding and feedback on the learner's performance.

When tasks become particularly complex, involving a variety of skills, it can be beneficial to break a task a part into component skills and role play them separately in a part-task training approach. VCATs and other Alelo courses use this approach to reinforce individual communication skills, as shown in Figure 4. Here the learner is practicing offering compliments to his host. Learners can practice individual responses by selecting from menus of options, as in this case, or speaking their response into a microphone.

 A screenshot of the VCAT (Virtual Cultural Awareness Trainer) software interface. The title bar says "VCAT VIRTUAL CULTURAL AWARENESS TRAINER" and "Taiwan". Below the title is "Practice 8". On the left is a large video frame showing a close-up of a man's face. On the right is a text box with instructions: "Compliment the host on his choice of restaurant by saying, 'Very beautiful.'". It lists four options with radio buttons:

- Hěn piāoliàng!
- Hěn mǎn yòng.
- Qǐng mǎn yòng.
- Qǐng piāoliàng.

 Below the text box is a smaller video frame showing a woman with a pink scarf, with the text "Outstanding!" above her. At the bottom are navigation buttons: "LESSONS", "INDEX", "PREV", "NEXT", and a central button with a book icon.

Figure 4. Learners can practice individual communication skills in a part-task training approach.

To encourage ongoing practice and provide an appropriate level of challenge, simulations can be made to vary both in terms of amount of scaffolding and degree of difficulty of the interactions. The Tactical

Interaction Simulator (TI Simulator) (Emonts et al., 2012) illustrates both dimensions of variability, as shown in Figure 5. The avatar in these examples is an Australian soldier on a peacekeeping mission in East Timor. The screenshots in the figure illustrate two different simulations of a clearance operation, in which the learner is supposed to keep civilians clear of hazardous areas. In the left screenshot, the learner is provided with a high degree of scaffolding, including a transcript of the dialogue, possible courses of action, and possible ways of expressing these courses of action in Tetum (the language spoken in East Timor). In the example on the right, the scaffolding is removed and the learner is expected to engage in conversation unassisted.



Figure 5. The Tactical Interaction Simulator can be played at a low level of difficulty and a high level of scaffolding (left), or a high level of difficulty and a low level of scaffolding (right).

The left example, in which the civilians are hostile, is at a low level of communicative difficulty all the learner can do in this case is to tell the civilians to calm down and call the police. The right example, in which the civilian is initially cooperative, is linguistically more difficult the learner must explain calmly why the civilian cannot enter the restricted zone and avoid raising tensions. These examples illustrate how virtual role-play simulations, if designed properly, can support learners at a variety of skill levels and encourage learners to practice and try alternative courses of action until they have fully mastered the target skills.

These examples also illustrate that virtual role play involves nonverbal communication as well as verbal dialogue. The body language of the virtual role players can communicate their emotions and attitudes in ways that their verbal responses may not. Conversely, virtual role play can enable learners to practice their nonverbal communication and use of body language. If the computing device has suitable sensors, it can track the learner's body language directly. If not, the learner can use menus or interface gestures to control the body movements of his avatar.

Virtual role-play simulations can serve multiple purposes and phases of training: walkthroughs, practice, and assessment. In walkthrough scenarios, the learner may have little or no mastery of the target skills and so the system provides a high degree of scaffolding and helps the learner walk through the scenario to get a feel for how to perform the task. The left screenshot in Figure 4 is an example of such a walkthrough

one doesn't need to know much Tetum to complete this simulation, although the score one receives depends upon how much Tetum is used. Practice simulations help learners develop their skills and involve progressively less amounts of scaffolding and higher levels of difficulty. In assessment simulations, scaffolding is withheld and learners must demonstrate that they can complete the task unassisted.

In summary, below is a list of desirable characteristics for virtual role play, as illustrated in these examples:

- Engaging, immersive experiences that simulate real-world interactions.
- Support for multiple computing devices and interface modalities.
- Support for speech recognition and other sensors for more realistic interaction.
- Nonverbal as well as verbal communication.
- Alternative courses of action, to promote replayability.
- Support for walkthroughs, practice, and assessment.
- Virtual coaching support.
- Part-task training of component skills.
- Varying levels of scaffolding.
- Varying levels of difficulty.

Role-Play Training and Scenario-Based Training

Virtual role-play training is related to scenario-based training. Scenarios and stories are used widely in training, and authoring tools are available to support their development. However, scenarios in general are much simpler than virtual role-play simulations, and so are the authoring tools used to create them.

Figure 6 shows an example scenario created by Van Nice (2014), created using Articulate Storyline (Articulate Global, 2015). In this approach to scenario-based training, each character in the scenario appears as a drawn or photographic character, in a sequence of still poses. The non-player character poses a question, presented on the screen. The learner chooses from a small set of multiple-choice answers. The non-player character then responds to the learner's choice, and the system gives feedback on that choice.

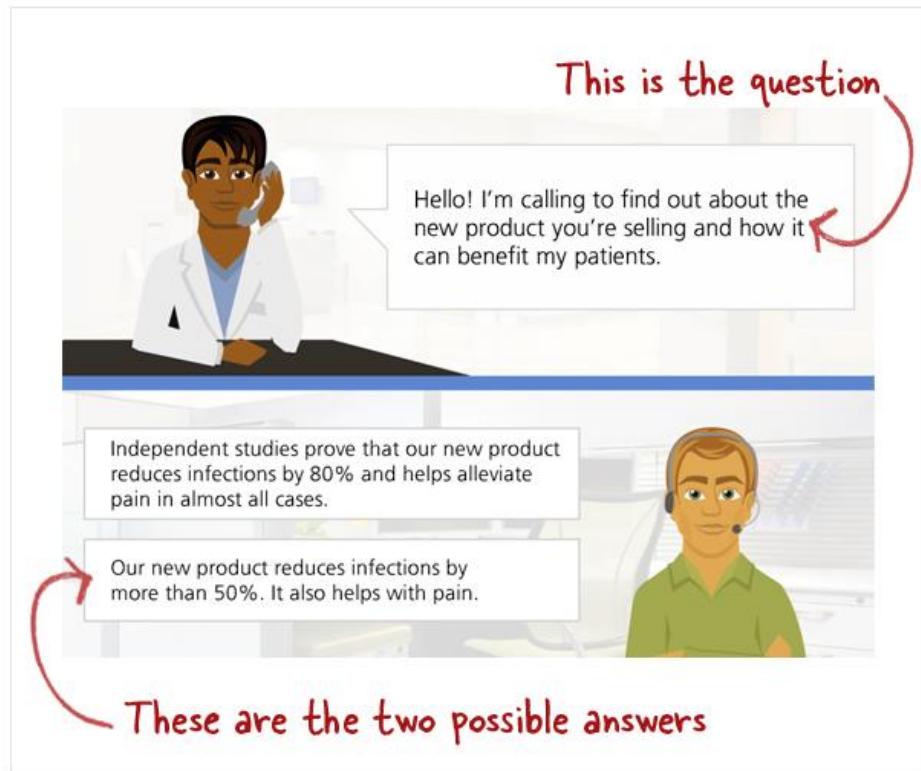


Figure 6. This example scenario was created using the Articulate Storyline authoring tool.

Scenarios such as this are useful for some purposes such walkthroughs. Current authoring tools make it possible to create such scenarios without any programming. However, they lack many of the characteristics discussed in the previous section, and this limits their utility. In particular, scenarios tend to limit learners to a small set of choices, as in this example. They are limited to a single question-response pair, as in this case, or a linear sequence of inputs and responses. This limits their replayability. Simulations in contrast support a range of possible inputs, responses, and outcomes, and so are more suitable for ongoing practice and sustainment. The challenge for role-play authoring tools is to make it easy to create such simulations with little or no programming.

Authoring Processes

Authoring virtual role play is not simply the application of a tool; it is a process. It can involve multiple stages, with different participants involved at each stage. This is true for any significant intelligent tutoring development effort, but it is especially true for virtual role-play authoring, because it can involve people with different skill sets. Authoring tools must be designed to support the intended process, participants, and roles.

Figure 7 shows one example development process, used to develop VCAT courses. Development proceeds in six distinct phases, from background sociocultural research through instructional design, scenario authoring, media production, and quality assurance. Each phase of authoring involves distinct activities and skill sets, and consequently, different authoring capabilities. The course also goes through an approval process with the client, which also involves multiple phases. Authoring tool features can vary depending upon the stage.

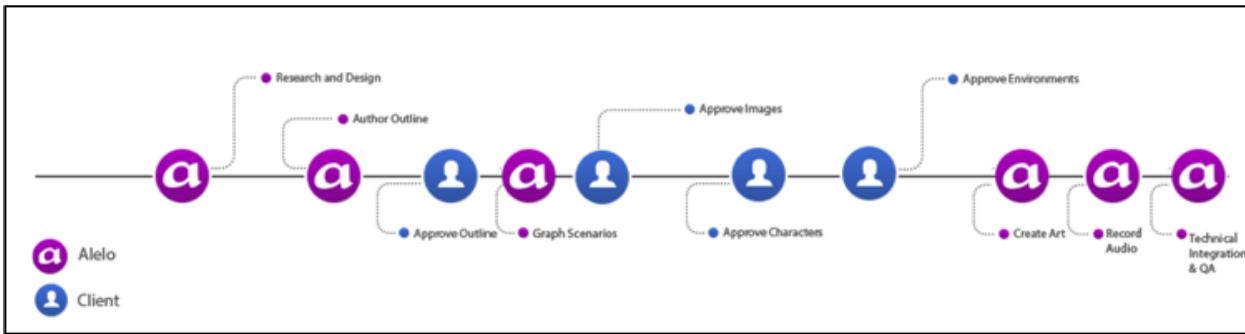


Figure 7. This example authoring process involves multiple phases and roles, both for the system developer and for the client.

Below are examples of some process issues that a good virtual role-play authoring toolset should support in order to create product-quality virtual role-play training systems:

- **Domain model validation.** The role-play simulation must reflect an accurate understanding of how the target skills are performed in real life. This is important when the training author and the subject matter experts are different people, or when multiple subject matter experts are required. Otherwise there is a risk that the course author will create content that appears to be correct but in fact is inaccurate. This is a critical issue for cultural awareness courses such as VCATs, which incorporate expertise in culture as well as military operations. For VCATs, we cross-validate cultural content from multiple sources to ensure that the final content correctly reflects the target culture.
- **Team collaboration and workflow.** Role-play simulation development often requires multidisciplinary teams. Authoring tools should support sharing among team members.
- **Courseware quality assurance.** The tools should support thorough testing and validation to ensure that the resulting content is free of mistakes. Again, VCATs provide a good case in point. Errors can creep in in the domain model, the instructional design and content, the artwork, and the interaction behavior.

VIRTUAL ROLE-PLAY AUTHORING TOOLS

Currently, few authoring tools are generally available for creating virtual role-play simulations. Virtual role-play developers such as SIMmersion (2013) and Kognito Interactive (Boyd, 2015) create simulations using in-house tools and methods; they do not make these tools available to others and publish few details about them. Scenario editors such as Articulate Storyline (Articulate Global, 2015) and Video RolePlay (Rehearsal Video Role-Play, 2015) make it easy to create simple scenarios but are not designed to support the creation of rich role-play simulations.

Page-based Authoring Tools

Most scenario authoring tools use a page metaphor, similar to slides in PowerPoint. The author creates a set of pages, where the virtual role player and learner's dialogue choices are bits of artwork embedded in the page. The dialogue progresses by jumping from page to page.

SkillStudio, the authoring toolset offered by Skillsoft, has support for creating role-plays (Skillsoft Ireland Limited, 2013). SkillStudio does not give users the option of creating new role-plays, but it permits users to edit existing role-plays developed by Skillsoft.

Skillsoft role-plays are composed of pages showing an image of a character saying something to the learner and a list of multiple-choice responses to select from, similar to the example in Figure 6. SkillStudio supports single-path role-plays and multiple-path role-plays. In single-path role-plays, there is only one correct choice in each turn of the role-play, and learner is constrained to follow the correct path. In multiple-path role-plays, each choice leads to a new dialogue page, each of which, in turn, leads to a set of successor pages. This results in a tree of pages. Skillsoft role-plays can be played in either Explore Mode or Summary Mode. Explore Mode is a kind of walkthrough mode in which the learner can explore the outcome of each option before making a choice. Summary Mode is a kind of assessment mode, in which the learner must make an immediate choice at each step in the role-play. Learners receive a cumulative score based on number of correct choices they make over the course of the role-play.

One limitation of the Skillsoft approach is that it offers the learner a limited range of options at each decision point. Each learner action is selected from a small list of choices, so learners learn to recognize appropriate responses instead of coming up with their own responses. Single-path role-plays constrain learners to follow a linear script. Multiple-path role-play trees offer more options, but they are not scalable. The number of pages is exponential in the depth of the tree. Realistic role-plays involving a series of conversational turns and a range of options become very large and time-consuming to produce.

ZebraZapps (Lee, 2013) is a more recently released authoring tool that supports the creation of role-plays as well as other interactive eLearning media. As in SkillStudio authors can author role-plays by creating a set of pages showing a picture of a character saying something and a set of multiple-choice options. The author can specify go-tos between pages, so that when the learner selects a choice it causes the course to jump to another page. The properties of graphical objects in the page, as well as the go-tos between pages, are presented in a table to facilitate editing.

ZebraZapps role-play applications do not require quite as many pages as SkillSoft role-plays, since authors can use go-tos to merge paths and share pages across paths. But since each simulation state is a separate page, dynamic simulations inevitably require large numbers of pages. Large numbers of go-tos result in complex control structures that are hard to follow and difficult to maintain.

Dialogue Authoring Tools

Dialogue authoring tools differ from the above tools in that there is an explicit model of the dialogue that the character is engaging in, independent of the screen artwork. Dialogue authoring tools are designed to enable authors to define complex dialogues with interactive characters. Some dialogue authoring tools are emerging that are designed specifically to create role-play simulations.

ChatMapper (Urban Brain Studios, 2014) is a general-purpose authoring tool for nonlinear dialogue. Authors can create dialogue trees and specify conditions under which branches are activated. It can thus be used to create complex simulations. Dialogues are compiled into the Lua scripting language (Lua, 2014), a commonly used scripting language in games. The ChatMapper editor has a built-in conversation simulator, which makes it easy for developers to test dialogues as they are developing them. Although ChatMapper is very flexible, it only takes care of authoring dialogue logic. Constructing complete role-play simulations with capabilities listed above, such as spoken dialogue, scaffolding, etc., inevitably requires additional Lua scripting and programming.

The USC Institute for Creative Technologies (ICT) has developed a series of experimental authoring tools for role-play development. For example the **Tactical Questioning authoring tool** (Gandhe et al., 2009) been used to create virtual role players for a system that trains tactical questioning skills. It supports a model of dialogue in which the virtual role player responds to questions posed by the trainee, and sometimes engages in subdialogues to negotiate with the trainee for compensation in return for the release of information. In this approach, the author creates a model of information that the virtual role player knows and can talk about. This includes information about objects, people, and places. The author then defines dialogue acts that the player and virtual role player can engage in concerning this information. Dialogue acts include questions, assertions, offers, threats, offers, and insults, as well greetings and closings to start and end the conversation. Dialogue moves are specified as state transition networks, in which the author can specify conditions under which transitions may occur. Conditions may include the emotional state of the character and character's willingness to comply and cooperate, which, in turn, are influenced by what the learner has said previously in the dialogue. The system uses statistical language processing techniques for natural language understanding as well as natural language generation to map between English text utterances and dialogue acts. The authoring tool enables the author to train the natural language processor by selecting which dialogue act to map to a given text utterance. Ghandhe et al. (2009) report that the developers used the Tactical Questioning authoring tool to create the first character, Hassan, after which two subject matter experts without previous experience building dialogue systems used the tool to author dialogue for two additional characters.

More recent ICT authoring tool named Situated Pedagogical Authoring (**SitPed**) uses the ChatMapper tool to create branching dialogue and incorporates a character simulator so that authors can test and annotate dialogue as they create it (Lane et al., in press). It also provides authors a tool for annotating dialogue texts to indicate how well they exhibit the skills being taught in the simulation. An evaluation of SitPed was conducted in 2014, and at the time of this writing, the results of this evaluation are still being analyzed.

Alelo has a suite of tools for creating training content employing virtual role play (Johnson & Valente, 2008). Alelo uses these in house and also makes them available to third parties. For example, the Danish Simulator (Dansksimulatoren, 2015), an award-winning game for learning Danish language and culture, was developed using Alelo's tools and platform. The toolset supports development teams throughout the authoring process, from background sociocultural research through building complete training systems. The tools and supporting methodology have enabled Alelo to deliver a wide range of effective culture and language training courses, which have a consistently high level of quality.

The core tools in the Alelo authoring toolset are **Xonnet** and **Tide**. Xonnet supports web-based authoring by teams of authors, operating on content stored in a central learning content management system. It provides content management functions necessary for collaborative authoring such as checking in and checking out of content. Tide is used to design and construct the virtual role-play content elements within each course. Other tools in the toolset edit and manage the media assets comprising simulations, such as character animations and voice recordings. Content is specified in a device-agnostic fashion so that it can run on personal computers and mobile devices, in web browsers, immersive games, mixed-reality environments, and even mobile robots. For each hardware/software configuration, Alelo has developed a content player capable of delivering content on that device and software platform.

To understand how authoring works one needs to know something about how the Alelo architecture controls the behavior of virtual role players (Johnson et al., 2012). Each virtual role player has a "brain" (decision engine) that controls a "body" (character persona and sensing-action layer) that operates within the simulated world or real-world environment. When the virtual role player is interacting with a trainee, the sensing-action layer receives inputs from the speech recognizer, user interface, other sensors, and the virtual-world simulation, and relays them to the decision engine to determine what the character should

do in response. The decision engine interprets the inputs in the context of the culture, current situation, and dialogue history to determine what *act* the trainee is performing. Acts are similar to the dialogue acts in Ghandhe et al.'s (2009) formulation, but also subsume nonverbal communication and other actions. For example in the VCAT Taiwan simulations the trainee's avatar might extend his hand in order to share hands or raise his glass to offer a toast. The decision engine interprets such behaviors as acts with communicative intent and chooses an action to perform in response. The decision engine is able to recognize a variety of possible acts, affording the trainee a range of possible courses of action. The decision engine then chooses what action to perform in response, and realizes that as a combination of speech and gesture for the sensing-action layer to perform.

Each virtual role-player model can incorporate a set of dynamic variables that represent the attitudes of the virtual role player toward the trainee. Trust and rapport are typically the most important variables. These can change over the course of the encounter in reaction to the trainee's actions and can influence what actions the virtual role player will take. In many of the simulations Alelo creates the trainee must first establish trust and rapport in order to accomplish the mission.

The job of Tide is to enable authors to create content that conforms to this architecture, enables the virtual role player to interpret the trainee's actions, and responds accordingly. For each encounter or scene, authors create an act library, which is the inventory of acts that the trainee or the virtual role player may perform during the encounter or scene. These can vary from simulation to simulation, but in practice authors reuse elements of previous act libraries when developing new act libraries. Authors also create utterance libraries, which consist of example utterances that express the meaning of the acts in the target language. To increase the coverage of utterances in the utterance library, authors can use a templatizer tool, based on the work of Kumar et al. (2009), to generalize utterances into utterance patterns that match a variety of utterances.

Tide provides an interactive diagramming tool for specifying interactive dialogues. Dialogues are depicted as directed acyclic graphs containing nodes representing acts, utterances, and nonverbal behaviors. Transitions may be conditioned on certain predicates becoming true, e.g., a character's trust level exceeding a certain threshold. Authors can also create subdialogues that are activated and deactivated during the course of the dialogue. Through these simple mechanisms authors can create complex dialogues with a variety of alternative paths. A testing function enables authors to execute a dialogue within the editor to validate the dialogue logic. This helps with the problem of quality assurance of the simulation content.

As authors create dialogues they incorporate assessment and feedback. Learner responses are scored and contribute to an overall assessment of the trainee's performance in the simulation. Some feedback, what we call *organic feedback*, is incorporated into the responses of the virtual role player and thus becomes an organic part of the simulation. For example, the virtual role player might take offence at the trainee's statement or display facial expressions that indicate discomfort or disapproval. Such feedback is powerful and effective because learners can immediately see the consequences of their actions. Other feedback takes the form of corrective and explanatory feedback to be provided by the Virtual Coach. The author supplies the feedback at authoring time, and it is up to the run-time content player to determine whether to present that feedback to the learner, based upon the chosen level of scaffolding or upon learner request.

Alelo tools are used to create role-play simulations that serve as walkthroughs, practice sessions, or assessments. They include single conversational turns for part-task training, as well as extended exchanges of several minutes in duration. Hundreds or even thousands of simulations have been authored to date using these tools.

Empowering Trainers Using Role-Play Configuration Tools

Current dialogue authoring tools reduce the amount of programming required to create role-play simulations. However to promote adoption of the virtual role-play approach at a really large scale, it is important that we empower trainers so that they can create their own virtual role-play simulations. This goal of empowering trainers is one of the next big challenges for adaptive intelligent tutoring systems (ITSs) generally, including the tools described in this volume. Visionaries such as Sottilare (2013) have called for interfaces to ITSs that teachers and instructors can use. However, there are just a few instances to date, such as ASSISTments (Heffernan & Heffernan, 2014) that teachers or trainers have used to any significant extent to create their own content. Alelo has developed a new product named VRP® MIL (Stuart, 2014) that is specifically designed to meet this need in the area of virtual role play.

VRP MIL was developed to meet the needs of military training organizations that wish to organize training exercises for their units at simulation training centers. Simulation training centers are equipped with computers for virtual training and staffed with personnel who are skilled in running training exercises using this equipment. The simulation center staff is permanently resident at the training center, while the units continually rotate through the center as part of their preparation for deployment.

When a unit wishes to organize a training program, the training officer associated with the unit typically works with the simulation center staff to define a series of training exercises for the unit to perform. The training officers are experts in training but may have little knowledge of simulation technology. It is up to the simulation center staff to quickly put together training simulations that meet the training officer's requirements. A common request from the training officer is training scenarios at varying levels of difficulty. The training officer might start with a training exercise at a high level of difficulty knowing that the trainees will likely fail the exercise in order to motivate the trainees to improve. The trainer will then undertake another exercise at a low level of difficulty, in which the trainees will likely succeed. They then undertake additional exercises at progressively higher levels of difficulty until the exercises again reach a high level of difficulty. By this point, the trainees have progressed to the point where they can successfully complete the mission with full confidence in their skills.

When the training is preparation for overseas deployments, a key challenge is providing training that accurately reflects the culture of the region of deployment. Unfortunately, the training officers and simulation staff may not have detailed knowledge of the target culture. Cultural subject matter experts, if available, may not have much knowledge of military missions or simulation technology. Moreover, if they are available they may not have accurate knowledge of the culture of the specific region; if they have been out of the country for an extended period, their knowledge may not be up to date.

VRP MIL helps trainers and simulation staff to overcome these challenges and quickly create training simulations that are culturally accurate and appropriate for the intended training objectives. It provides trainers with a library of reusable virtual role players, each intended to perform a designated role in training simulations. Example roles include local leaders, guards and sentries, shopkeepers, and passers-by on the street. Instead of authoring content from scratch using authoring tools, trainers populate the virtual training world with virtual role players and configure them to meet their needs. The behavior of each virtual role player has been validated beforehand as culturally accurate, ensuring that the resulting training simulation is also culturally accurate. VRP MIL is built as a plug-in that integrates into the popular VBS simulation-based training tool (Bohemia Interactive Simulations, 2015), which already provides users with tools for constructing virtual worlds and populating them with buildings, vehicles, and other entities.

We have developed the VRP MIL framework and a basic library of virtual role players (VRPs), and now plan to extend it with form-based interfaces for providing the necessary configuration parameters.

Configuration parameters will include the level of difficulty interaction with the VRP, as well as specific topics that the VRP is prepared to discuss with the trainee. This fits well with the way the military currently defines roles for live role players in training exercises. These configuration parameters will then be automatically inserted into the dialogue model to generate the target behavior. Authoring tools will still be used to create the VRP models, but this way each VRP model will undergo much broader use. Simulation center staff will have the option to use the authoring tools themselves to add adapt and extend the VRP library.

VRP MIL underwent a successful trial evaluation in February 2015 at the NATO Joint Force Training Centre in Bydgoszcz, Poland, with NATO units preparing to travel to Afghanistan on training and support missions. From there, we anticipate its adoption by NATO member nations and allied nations preparing for overseas coalition operations.

Conclusions and Future Directions

Virtual role play is becoming an increasingly important training method for intelligent learning environments. It is being applied to an ever-broadening range of education and training applications, particularly for cross-cultural communication. Progress in authoring tool development for this class of applications has made this possible. Emerging developments such as role-play configuration tools are likely to further accelerate the expansion and large-scale adoption of this technology.

Dialogue authoring tools for role-play simulations are in some ways similar to tutorial dialogue authoring tools such as AutoTutor's authoring tools (Nye et al., 2014) or TuTalk (Jordan et al., 2007), and there is much that we can learn from these tools. However role-play simulations have their own unique characteristics that warrant their own class of authoring tools.

Role-play authoring tools have been most successful when they take into account the tasks and roles of the people using the tools, and the processes by which content is developed. This is an important general lesson for authoring tools for adaptive ITSs. The clearer understanding we have of our intended users the better a job we can do of addressing their needs.

As we have seen, existing page-based authoring tools are quite capable of creating simple role-play scenarios. These tools are very widely available, and many training developers are familiar with their use. Virtual role-play and associated tools are most likely to be adopted when they offer clear and compelling advantages over existing methods, especially in skill development, authentic assessment, and promoting behavior change. There is a general lesson here for authoring tools for the adaptive ITSs of GIFT. Researchers in adaptive ITSs often wonder why their technologies are not being adopted more widely. Existing authoring tools are quite capable of creating simple versions of various types of learning environments, and trainers are unlikely to switch to new tools if they do not see a compelling advantage.

The general architecture for GIFT, as described in Sottilare (2012), needs to be clarified so that it accommodates the instructional interaction typical of virtual role-play simulations. According to the GIFT architecture the tutor-user interface and the training app client are separate, and interact with users separately. However, as we have seen, assessment and feedback are often tightly integrated into virtual role-play simulations, and feedback is an organic part of virtual-role-player behavior. If the GIFT architecture is to support virtual role play it should support such integrated interaction.

Virtual role-play systems can collect valuable, accurate data about trainee performance. There is an opportunity to capture and exploit this data as part of the GIFT architecture. One way of doing this via the TinCan API. Once data are captured via TinCan and stored in a Learner Record Store (LRS), it is possible

to analyze these data and develop more granular models of learner skills, which in turn can be used to tailor training. If these are integrated with job performance data, it would provide a method for providing just-in-time training and promoting behavior change on the job.

There is a need in virtual role-play systems for flexible domain models of dialogue that can be used in a variety of ways. In live role-play training exercises, it can be useful to switch roles, so that the trainee can better understand the perspective of the other person. For virtual role-play systems to have similar flexibility, they require dialogue models that capture the interaction while being agnostic as to which roles are played by the learners and which are played by the virtual role players. This is very consistent with the GIFT approach of modeling domain expertise independent of specific instructional use.

Looking ahead, speech recognition will continue to improve. Sensor and interface technologies will increase in performance and reduce in cost. This will make it easier to deliver virtual role-play training and assessment in a wider range of domains, to a wider range of organizations. Techniques that have been developed and proven in military training can be applied to a wide range of domains in training, development, and behavior change for a wide range of organizations. Many of these currently rely on traditional methods and informal observation of performance. There are many opportunities to achieve radical improvements in training and performance development, through virtual role-play methods that employ realistic models of skill and provide accurate assessments of performance.

References

- Articulate Global (2015). Storyline 2: Create interactive e-learning, easily. Retrieved Feb. 19, 2015 from <https://www.articulate.com/products/storyline-why.php>.
- Barrows, H.S. (1993). An overview of the uses of standardized patients for teaching and evaluating clinical skills. *Academic Medicine, (1993)*, 443-451.
- Bohemia Interactive Simulations (2015). VBS3: The future battlespace. Retrieved Feb. 19, 2015 from www.bisimulations.com/virtual-battlespace-3.
- Boyd, P. (2015). Dooplo - Kognito's human interaction platform. Retrieved Feb. 19, 2015 from <http://patboyd.com/site/projects/kognito-platform>.
- DanskSimulatoren (2015). DanskSimulatoren revolutionizing language learning. Retrieved Feb. 19, 2015 from www.danskSimulatoren.dk.
- Emonts, M., Row, R., Johnson, W.L., Thomson, E., Joyce, H. de S., Gorman, G. & Carpenter, R. (2012). Integration of social simulations into a task-based blended training curriculum. In *Proceedings of the 2012 Land Warfare Conference*. Canberra, AUS: DSTO.
- Gandhe, S., Whitman, N., Traum, D. & Artstein, R. (2009). An integrated authoring tool for tactical questioning dialogue systems. In *6th Workshop on Knowledge and Reasoning in Practical Dialogue Systems*, Pasadena, California. 2009. Retrieved Feb. 19, 2015 from <http://people.ict.usc.edu/~traum/Papers/krp09authoring.pdf>.
- Heffernan, N. & Heffernan, C. (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.
- Johnson, W.L. (2010). Serious use of a serious game for language learning. *International Journal of Artificial Intelligence in Education*, 20(2), 175-195.
- Johnson, W.L., Friedland, L., Schrider, P., Valente, A. & Sheridan, S. (2011). The Virtual Cultural Awareness Trainer (VCAT): Joint Knowledge Online's (JKO's) solution to the individual operational culture and language training gap. In *Proceedings of ITEC 2011*. London: Clarion Events.
- Johnson, W.L., Friedland, L., Watson, A.M. & Surface, E.A. (2012). The art and science of developing intercultural competence. In P.J. Durlach & A.M. Lesgold (Eds.), *Adaptive Technologies for Training and Education*, 261-285. New York: Cambridge University Press.
- Johnson, W.L. & Lester, J.C. (in press). Twenty years of face-to-face interaction with pedagogical agents. *International Journal of Artificial Intelligence in Education*.

- Johnson, W.L. & Valente, A. (2008). Collaborative authoring of serious games for language and culture. *Proceedings of SimTecT 2008*.
- Jordan, P., Hall, B., Ringenberg, M., Cue, Y. & Rosé, C. (2007). Tools for authoring a dialog agent that participates in learning studies. In R. Luckin et al. (Eds.), *Artificial Intelligence in Education*, 43-50. Amsterdam: IOS Press.
- Kim, J.M., Hill, R.W. Jr., Durlach, P.J., Lane, H.C., Forbell, E., Core, M.G., Marsella, S., Pynadath, D.V. & Hart, J. (2009). BiLAT: A game-based environment for practicing negotiation in a cultural context. *International Journal of Artificial Intelligence in Education*, 19, 289-308.
- Lane, H.C., Core, M.G. & Goldberg, B.S. (in press). Lowering the skill level requirements for building intelligent tutors: A review of authoring tools. In R. Sottilare, A. Graesser, Xiangen Hu & K. Brawner (Eds.), *Design Recommendations for Adaptive Intelligent Tutoring Systems: Authoring Tools (Volume 3)*. Orlando, FL: U.S. Army Research Laboratory.
- Lee, S. (2013). Build a role play in a day with ZebraZapps. Retrieved Feb. 19, 2015 from <http://vimeo.com/80417830>.
- Loke, S.-K., Blyth, P. & Swan, J. (2012). Student views on how role-playing in a virtual hospital is distinctly relevant to medical education. *Proceedings of ascilite 2012*. Retrieved Feb. 19, 2015 from <http://www.ascilite.org/conferences/Wellington12/2012/pagec16a.html>.
- Lua (2014). Lua: The programming language. Retrieved Feb. 19, 2015 from www.lua.org.
- 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 (2014), 427-469.
- Radecki, L., Goldman, R., Baker, A., Lindros, J. & Boucher, J. (2013). Are pediatricians “game”? Reducing childhood obesity by training clinicians to use motivational interviewing through role-play simulations with avatars. *Games for Health Journal*, 2(3), 174-178.
- Rehearsal Video Role-Play (2015). Rehearsal features. Retrieved Feb. 19, 2015 from <http://www.videoroleplay.com/features>.
- Robinson, L.J.B. (1987). Role playing as a sales training tool. *Harvard Business Review*, May-June 1987, No. 87310. Cambridge, MA: Harvard Business Publishing.
- Sandler Training (2014). A better way to role play. Retrieved on Feb. 19, 2015 from <http://www.sandler.com/blog/a-better-way-to-role-play/>.
- SIMmersion (2013). Technology: Ground-breaking technology lets SIMmersion deliver effective communication training to learners of all kinds. Retrieved Feb. 19, 2015 from <http://simmersion.com/Technology.aspx>.
- Simmons, T.G. (2010). Using virtual role-play to solve training problems: How do you train employees to think on their feet? *eLearn magazine*, June 2010. Retrieved Feb. 19, 2015 from <http://elearnmag.acm.org/archive.cfm?aid=1821985>.
- Skillsoft Ireland Limited (2013). Roleplays. Retrieved Feb. 19, 2015 from http://documentation.skillsoft.com/en_us/sstudio/index.htm#17853.htm.
- Sottilare, R.A. (2012). A modular framework to support the authoring and assessment of adaptive computer-based tutoring systems. Paper presented at the Interservice/Industry Training, Simulation & Education Conference (I/ITSEC), Orlando, FL.
- Sottilare, R.A. (2013). Pushing and pulling toward future ITS learner modeling concepts. In R. Sottilare, A. Graesser, X. Hu & H. Holden (Eds.), *Design recommendations for intelligent tutoring systems*, 195-198. Orlando, FL: U.S. Army Research Laboratory.
- Stuart, S. (2014). Using video games to prepare for the culture shock of war. *PC.com*, Nov. 24, 2014. Retrieved Feb. 19, 2015 from <http://www.pc当地.com/article2/0,2817,2472395,00.asp>.
- Urban Brain Studios (2014). *Chat Mapper 1.7 documentation*. Retrieved Feb. 19, 2015 from <http://www.chatmapper.com/documentation>.
- USC ICT (2013). Situated pedagogical authoring for virtual human-based training. Retrieved on Feb. 19, 2015 from http://ict.usc.edu/wp-content/uploads/overviews/Situated%20Pedagogical%20Authoring_Overview.pdf.
- Van Nice, J. (2014). Toolbox Tip: Creating scenarios in Articulate Storyline — No programming necessary. Retrieved Feb. 19, 2015 from <https://www.td.org/Publications/Blogs/Learning-Technologies-Blog/2014/08/Creating-Scenarios-in-Articulate-Storyline>.
- Wilcox, A. (2012). Somali-Americans assist reserve Marines with pre-deployment training. *The Daily News Jacksonville, NC*, Dec. 13.

Chapter 19 Emerging Trends in Automated Authoring

Andrew M. Olney¹, Keith Brawner², Phillip Pavlik¹, Kenneth R. Koedinger³

¹University of Memphis; ²US Army Research Laboratory; ³Carnegie Mellon University

Introduction

Traditional intelligent tutoring systems (ITS) are specialized feats of engineering: they are custom-made to implement a theory of learning, in a particular domain, within a specific computer environment. There are many ways to describe or categorize authoring tools used to make ITSs (Murray, 2004). This chapter considers authoring tools primarily in terms of intelligent tutor paradigms. Three popular ITS paradigms are dialogue-based tutors (Nye, Graesser & Hu, 2014), constraint-based tutors (Mitrovic, 2012), and model-tracing tutors (Anderson et al., 1995). These paradigms may be distinguished along two abstract axes, as shown in Figure 1. The axes reflect how the learning task is defined and how student progress in the task is measured.

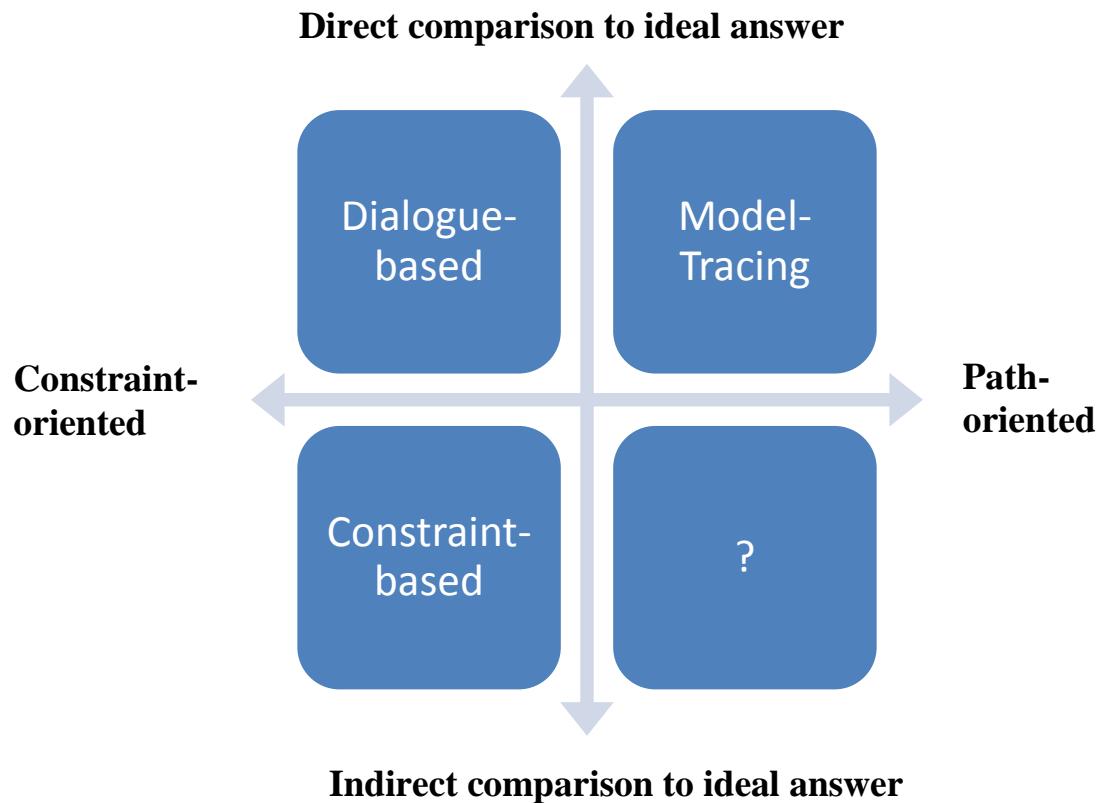


Figure 6. Tutoring paradigms arranged by orientation (path vs. constraint) and comparison to ideal answer (direct vs. indirect).

The horizontal axis indicates whether the paradigm is *primarily* path-oriented or constraint-oriented. A path-oriented paradigm conceives the learning task as a sequence of steps that lead to a solution. For example, the instructional theory behind model-tracing tutors can be expressed within the knowledge-

learning-instruction (KLI) framework (Koedinger, Perfetti & Corbett, 2012). Critical to the KLI framework are the ideas that (a) most of our knowledge in any area of expertise (e.g., grammar, algebra, design) is in the form of procedural skills, which are learned by induction from experience and feedback, and (b) two forms of instruction that best facilitate such learning are problem solving practice with as-needed feedback on student errors and as-needed examples of correct behavior (next step hints). The key is to engage the learner in the process of doing, that is, engaging in the target activity, and provide personalized tutoring support for the learner that adapts to their particular needs. Tutoring support is achieved by the use of a model of desired or correct performances and of particularly common undesired or incorrect performances. This *direct* comparison against a model (which includes ideal answers) also situates model-tracing on the vertical axis. Each student's actions are traced against this model such that feedback can be generated when undesired performance is observed and next-step hints can be generated when students are stuck. In both cases, the emphasis is on minimal intervention (Anderson et al., 1995) in order to maximize student active and constructive involvement in the thinking and learning process.

Conversely, a constraint-oriented paradigm conceives of the learning task as attaining a solution state irrespective of the path that led to it. Both dialogue-based and constraint-based paradigms share this property but they differ in many respects, most notably in how they represent knowledge and compare the student answer to an ideal answer. Dialogue-based tutors are typically frame-filling systems (McTear, 2002) that fill slots in a frame in any order. For example, given the physics question, "If a lightweight car and a massive truck have a head-on collision, upon which vehicle is the impact force greater, and why?", a dialogue-based system might have the slots [The magnitudes of the forces exerted by A and B on each other are equal] and [If A exerts a force on B, then B exerts a force on A in the opposite direction]. If a user says, "the forces are equal," the system would recognize that the first slot is filled and follow up with a question to fill the second slot like, "What can you say about the direction of the forces?" The slots are known as expectations, or expected components of the ideal answer (Graesser, D'Mello, et al., 2012), and the follow-up questions used to fill out the frame are aligned with models of naturalistic human tutoring (D'Mello, Olney & Person, 2010; Graesser & Person, 1994; Graesser, Person & Magliano, 1995; Person, Graesser, Magliano & Kreuz, 1994) plus ideal pedagogical strategies in some versions. Dialogue-based systems determine whether a slot, or expectation, is filled by *directly* comparing the student's answer to an ideal answer, typically using methods like latent semantic analysis (Landauer, McNamara, Dennis & Kintsch, 2007) and other semantic matching algorithms. Dialogue-based tutors can be considered as a very narrow form of constraint-based tutors where the constraints are defined by whether all slots are filled, i.e., all expectations are met.

Constraint-based tutors operationalize constraints as consisting of a relevance condition (R) and a satisfaction condition (S) (Ohlsson, 1992). The constraint is only applicable when the relevance condition is met, at which point the satisfaction condition defines what conditions the student's solution must meet in order to be correct. Only solutions that violate no constraints are correct. Constraints therefore do not specify a path or set of paths to a solution but rather define a space of correct solutions (Ohlsson & Mitrovic, 2007). For example, a constraint for fraction addition might have the relevance conditions *problem statement*: $a/b + c/d$ and *student solution*: $(a+c)/n$ with satisfaction condition $b=d=n$: the solution is correct only when the denominators of the problem statement and student solution are equal (Ohlsson, 1992). Constraints are well suited for design tasks and tasks that are ill defined precisely because they allow solutions to be recognized without requiring them to be enumerated by the author. With regard to the vertical axis, constraint-based tutors do not directly compare a solution to an ideal solution but instead compare *indirectly* via preserved and violated constraints.

The characterization presented in Figure 1 is undoubtedly an over-simplification of the differences between these tutoring paradigms because they differ on so many other dimensions. Moreover, they share more features than Figure 1 represents, because path-oriented tutors can relax ordering restrictions and constraint-oriented tutors can incorporate path-like ordering restrictions. However, from an authoring tool

standpoint, the above depiction highlights some of the key authoring problems faced by each paradigm. Model-tracing tutors require a model to trace, commonly in the form of sequences of steps and production rules that require next-step hints. Dialogue-based tutors require a set of expectations and associated follow-up questions (e.g., hints or prompts) when the expectations are unfulfilled. Constraint-based tutors require a set of constraints and associated feedback for their violations.

This chapter discusses several emerging approaches to ITS authoring that attempt to go beyond the typical human-created practice and automate more of the authoring process than has been previously attempted. Efforts are currently being undertaken in order to ease this burden from the authors in the form of programming by tutoring, automated concept map generation, metadata tagging, extensive content reuse, and continual refinement. With respect to Figure 1, the chapter emphasizes automated authoring in the model-tracing and dialogue-based traditions (see Mitrovic et al., 2006 and Mitrovic et al., 2009 for discussion of automated authoring of constraint-based tutors).

Related Research

Advanced Authoring for Model-Tracing Tutors

This section focuses on authoring tools for model-tracing tutors. The instructional approach in such tutors is to provide students with one-on-one tutoring support as they work on problem or activity scenarios of varying complexity. They do so within rich interface tools or simulation environments, for example, solving a physics problem using tools for drawing and annotating a free body diagram, and for writing and solving equations (e.g., VanLehn, 2006); solving a real-world quantitative reasoning problem (e.g., which cell phone plan to choose) using tools for creating tables, graphs, and equations (e.g., Koedinger et al., 1997); designing an efficient system using a thermodynamics simulation (see Fig. 26 in Aleven et al., 2009); and making an English grammar choice using a pop-menu (Wylie, Koedinger & Mitamura, 2010).

Effective and efficient authoring depends on how *completely, accurately, and quickly* an author can specify a sufficiently complete set of desired and common undesired student actions. This set of reasonable actions was traditionally specified in a general artificial intelligence (AI) rule-based system (cf., Anderson et al., 1995). For example, in a production system, each production rule is annotated with instructional messages, such as (a) next-step hints in the case of productions that represent desirable student actions and (b) error feedback messages in the case of productions that represent common student errors or underlying misconceptions. One successful alternative to production system authoring is to concretely enumerate, for each problem scenario, every action along all reasonable solution paths. This is the “example-tracing” approach taken in the Cognitive Tutor Authoring Tools (CTAT) (Aleven, McLaren, Sewall & Koedinger, 2009), a complete tutor-authoring suite that has been used to create several dozen ITSs.

A second alternative to hand-authoring production systems is to have the author tutor a machine learning system that learns the production system (largely) from scratch. This is the approach taken by SimStudent (Matsuda, Cohen & Koedinger, 2015). SimStudent learns problem-solving skills from the two kinds of instruction that are arguably the most powerful in human skill acquisition: learning from examples and learning from (feedback on) doing (e.g., Gick & Holyoak, 1983; Roediger & Butler, 2011; Zhu & Simon, 1987). Figure 2 shows an example of SimStudent being tutored on algebra equation solving. An example of an acquired production rule (in the JESS language) from the first author demonstration is shown on the right. SimStudent has three online learning mechanisms that focus on learning (1) information retrieval paths (clauses of the IF-part of the production that identifies where in the interface relevant information may lie), (2) preconditions on actions (clauses of the IF-part that constrain when the production is appropriate), and (3) action plans (compositions of functions that compute appropriate actions). The

newest addition to SimStudent is a representation learning mechanism that learns the general structure of declarative memory structures, which are the basis for both the operation and learning of production rules (Li, Matsuda, Cohen & Koedinger, 2015).

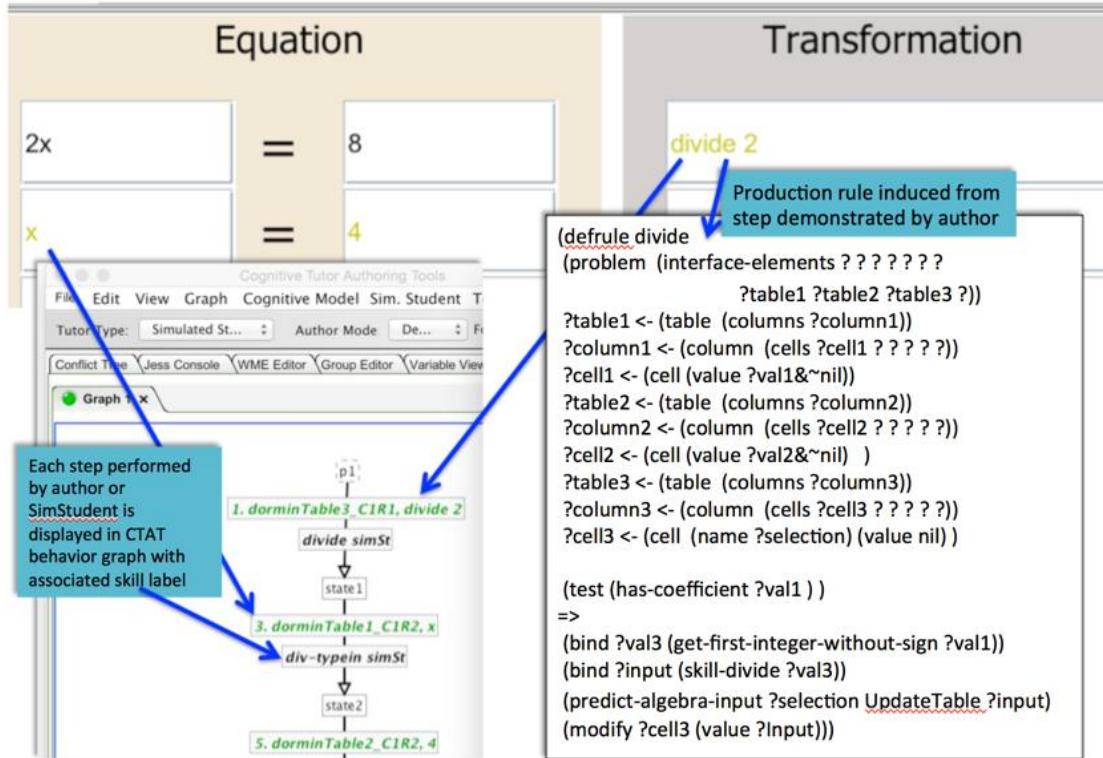


Figure 2. After using CTAT to create an interface (shown at top) and entering a problem (“ $2x=8$ ”), the author begins teaching SimStudent either by giving yes-or-no feedback when SimStudent attempts a step or by demonstrating a correct step when SimStudent cannot (e.g., “divide 2”). SimStudent induces production rules from demonstrations (example shown on right) for each skill label (e.g., “divide” or “div-typein” shown on left). It refines productions based on subsequent positive (demo or yes feedback) or negative (no feedback) examples.

The use of SimStudent as authoring tool is still experimental, but there is evidence that it may accelerate the authoring process and that it may produce more accurate cognitive models. In one demonstration, Matsuda et al. (2015) explored the benefits of a traditional *programming by demonstration* approach to authoring in SimStudent versus a *programming by tutoring* approach, whereby SimStudent asks for demonstrations only at steps in a problem/activity where it has no relevant productions and otherwise it performs a step (firing a relevant production) and asks the author for feedback as to whether the step is correct/desirable or not. They found that programming by tutoring is much faster, 13 productions learned with 20 problems in 77 minutes versus 238 minutes in programming by demonstration. They also found that programming by tutoring produced a more accurate cognitive model whereby there were fewer productions that produced overgeneralization errors. Programming by tutoring is now the standard approach used in SimStudent and its improved efficiency and effectiveness over programming by demonstration follow from having SimStudent start performing its own demonstrations. Better efficiency is obtained because the author need only respond to each of SimStudent’s step demonstrations with a single click, on a yes or no button, which is much faster than demonstrating that step. Better effectiveness is obtained because these demonstrations better expose overgeneralization errors to which the author

responds “no” and the system learns new IF-part preconditions to more appropriately narrow the generality of the modified production rule.

In a second demonstration of SimStudent as an authoring tool, MacLellan, Koedinger & Matsuda (2014) compared authoring in SimStudent (by tutoring) with authoring example-tracing tutors in CTAT. Tutoring SimStudent has considerable similarity with creating an example-tracing tutor except that SimStudent starts to perform actions for the author, which can be merely checked as desirable or not, saving the time it otherwise takes for an author to perform those demonstrations. That study reported a potential savings of 43% in authoring time by using SimStudent to aid in creating example-tracing tutors. A third demonstration by Li, Stampfer, Cohen, and Koedinger (2013) evaluated the empirical accuracy of the cognitive models that SimStudent learns as compared to hand authored cognitive models. The accuracy of a cognitive model in this demonstration was measured by the so-called “smooth learning curve” criteria (Martin, Mitrovic, Mathan & Koedinger, 2011; Stamper & Koedinger, 2011) that tests how well a cognitive model predicts student performance data over successive opportunities to practice and improve. Across four domains (algebra, fractions, chemistry, English grammar), Li et al.(2013) found that the cognitive model acquired by SimStudent produced cognitive models that typically produced better predictions of learning curve data (in 3 of 4 cases). More ambitious attempts to improve and evaluate SimStudent as a tutor authoring aid are underway. SimStudent and other means for AI-driven enhancement of ITSs, including data-driven hint generation and Markov decision process algorithms to optimize tutor action choices, are discussed in Koedinger et al. (2013).

Advanced Authoring for Dialogue-Based Tutors

In dialogue-based ITSs (Graesser et al., 2005; Olney et al., 2012, Rus et al., 2014), the computer attempts to tutor the student by having a conversation with them. These ITSs present similar challenges in ITS authoring as those without natural language dialogue, but there are greater dialogue-authoring demands than typical ITSs. The dialogue is typically authored by a subject matter expert (Graesser et al., 2004), though attempts have been made to semi-automate the process by automatically generating questions and representations that a subject matter can select or modify (Olney, Cade & Williams, 2011; Olney, Graesser & Person, 2012). However, both manual and semi-automated approaches have a common weakness: a shortage of motivated experts. In other words, experts are scarce, and it is uncommon for experts to volunteer their time to author ITS content. Without willing experts to use an authoring tool, an authoring tool will remain unused.

Our recent work addresses the shortage of motivated experts by considering expertise and motivation independently. Expertise may be approximated by allowing novices to do the authoring but then having other novices check the work to ensure quality. Motivation may be addressed by disguising the authoring task as another task in which novices are already engaged. We combine these two approaches in the BrainTrust system. In order to enhance motivation, BrainTrust leverages out-of-class reading activities, specifically online reading activities using eTextbooks through providers like CourseSmart¹, as opportunities for ITS authoring. As students read online, they work with a virtual student on a variety of educational tasks related to the reading. These educational tasks are designed to both improve reading comprehension and contribute to the creation of an ITS based on the material read. After reading a passage, the human student works with the virtual student to summarize, generate concept maps, reflect on the reading, and predict what will happen next. The tasks and interaction are inspired by reciprocal teaching (Palincsar and Brown, 1984), a well-known method of teaching reading comprehension strategies. Thus the key strategies to enhance motivation are leveraging a reading task to which the user has already committed, a teachable agent that enhances motivation (Chase et al., 2009), and a collaborative dialogue that increases arousal (D’Mello et al., 2010).

¹ <http://www.coursesmart.com/>

The virtual student's performance on these tasks is be a mixture of previous student answers and answers dynamically generated using AI and natural language processing techniques. As the human teaches and corrects the virtual student, they, in effect, improve the answers from previous sessions and author dialogues and a domain model for the underlying ITS. The process of presenting previously proposed solutions to a task for a new set of users to improve upon has been called "iterative improvement" in the human computation literature (von Ahn, 2005; Chklovski, 2005; Cycorp, 2005). These methods often use a simple heuristic that if the majority of evaluating users agrees a solution is correct, then the solution is correct, a process sometimes referred to as "majority voting." However, even simple tasks, such as determining if an image includes the sky, can have non-agreeing "schools of thought" who systematically respond in opposing ways (Tian & Zhu, 2012). Therefore it is preferable to use Bayesian models of agreement jointly to determine the ability of the user (and their trustworthiness as teachers) as well as the difficulty of the items they correct (Raykar et al., 2010). Although Bayesian approaches of this kind are an emerging research area, they are being actively pursued by the massive open online course (MOOC) community, because peer-grading is an important component of scaling MOOCs to many thousands of students (Piech et al., 2013).

Because BrainTrust activities are designed to facilitate learning (both of the content and of reading comprehension strategies) while preserving motivation, not all of the BrainTrust activities directly relate to the authoring of an ITS. In fact, the primary task that relates to ITS authoring is the construction of concept maps, as show in Figure 2. Concept maps can be used to generate exercises and questions in a dialogue-based ITS (Olney, Cade & Williams, 2011; Olney, Graesser & Person, 2012). They can also be used to generate rather trivially direct instruction, e.g., "attitudes are made of emotions and beliefs," as in Figure 2. With a small amount of additional information, such as the overall gist of a text passage, concept maps can also be used to generate larger summaries or "ideal answers" (Graesser et al., 2005). In the example given, the gist is "attitudes and attitude change," and using this gist, a concept-map driven summary can be topicalized so that "attitudes" are the key concept rather than another node. Topicalization is important because non-hierarchical concept maps can be read in any order, but a given text passage can only be read in the linear order in which it was written.

S vvmLight1TestPage.html?assignmentId=test&CondOrder=Read-Hi-Lo&TopOrde C Google

Social Cognition: Attitudes and Attitude Change-Copping a 'Tude

Journey Question 14.3 How are attitudes acquired and changed?

What is your attitude toward affirmative action, euthanasia, environmental groups, the situation in the Middle East, the death penalty, legalized abortion, junk food, psychology? Your answers, which are often influenced by social situations, can have far-reaching effects on your behavior. Attitudes are intimately woven into our actions and views of the world. Our tastes, friendships, votes, preferences, goals, and behavior in many other situations are all touched by attitudes (Baumeister and Bushman, 2011). Let's see how attitudes are formed and changed.

What specifically is an attitude? An attitude is a mixture of belief and emotion that predisposes a person to respond to other people, objects, or groups in a positive or negative way. Attitudes summarize your evaluation of objects (Bohner and Dickel, 2010). As a result, they predict or direct future actions.

Alright. So the important things to remember are Attitudes are made of beliefs Attitudes are made of emotions Attitudes affect actions Does that sound right?

```

graph TD
    attitudes[attitudes] -- "affect" --> actions[actions]
    attitudes -- "made of" --> emotions[emotions]
    attitudes -- "made of" --> beliefs[beliefs]
    
```

Submit

Figure 7. BrainTrust during a concept map activity

Advanced Component-Based Authoring

The previous sections describe efforts to automate authoring of a particular ITS component, such as the model in model-tracing tutors. However, there are also emerging technologies that facilitate the reuse and dynamic configuration of existing components, which allow for a different kind of automated authoring. Instead of authoring the components, these technologies attempt to dynamically assemble components for a particular learning objective. An outline of these technologies is presented in this section as a possible path forward to completely remove ITS expertise required in component-based authoring. In short, the steps to this process, addressed in further detail below, are the following:

- (1) Gather content.
- (2) Make the content discoverable.
- (3) Make the content customizable.
- (4) Generate additional tutoring-type information.
- (5) Perform delivery for both information and practice sessions.
- (6) Perform ITS-standard tasks (learner modeling, experience tracking, etc.); not discussed here.
- (7) Repeat: perform pedagogical selection/adaption (steps 1–6).

With regards to gathering (1) of content, the Internet presents a wealth of information, but little of it is relevant to educational goals. There are a few such efforts that attempt to make learning-specific resources available: the Learning Registry (Jesukiewicz & Rehak, 2011), Gooru Learning (GooruLearning, 2014) and, to a lesser extent, the Soldier-Centered Army Learning Environment (Mangold, Beauchat, Long & Amburn, 2012). Fundamentally, each of these has faced the problem of indexing learning content for gathering purposes. The Learning Registry adopts a solution of maintaining separately developed, but interlinked, content repositories while making indexing information available. Gooru instead allows for a centrally managed cloud of content, indexed in the same fashion as a search engine.

Continuing with the Internet analogy, search engines make content discoverable through indexing and cross-referencing. Each of the two main learning architectures must make the content discoverable (2) for a given topic in order for it to be used. Gooru Learning takes a traditional web approach of using community-curated metadata tags, while the Learning Registry has a project to automate the generation of these tags using a project called “Data for Enabling Content in Adaptive Learning Systems (DECALS)” (Veden, 2014). Both approaches make use of metadata-based descriptions of the content in order to drive content selection, in approaches inspired by search engines. The content is made customizable (3) through the editing access from the Gooru platform, or a Sharable Contend Object Reference Model (SCORM) editing and packaging standard (Initiative, 2001) is made available through the Re-Usability Support System for eLearning (RUSSEL) for management of repurposing courses, documents, and multimedia (Eduworks Corporation, 2014). Such systems allow content to be found via search of metadata attributes (e.g., reading level, interactivity index, etc.) and customized for the user.

Generating tutoring-type information (4) is discussed in the previous sections, but simply involves the supplementation of a piece of content with learning-relevant information. One such example of a process involves the generation of a concept map of the key topics contained within the indexed material. Such a concept map can then be used grouping of learning content, with underlying content used for the supplementation of additional learning-relevant items. Examples of such machine-generated learning-relevant information include topic sequencing (Robson, Ray & Cai, 2013), question generation for learning assessment (Olney, Graesser & Person, 2012), hint generation for student help during learning, or other supplemental information.

Content delivery (5) is both an easy and a difficult problem. Both Gooru and the Generalized Intelligent Frameworks For Tutoring (GIFT) support delivery via a web browser, which can easily deliver the majority of modern content. Difficulty stems from more complex SCORM objects, executable programs, 3D simulations, or other items. RUSSEL makes use of human authoring of Gagné’s 9 events (Gagné & Gagné, 1985), while GIFT automates the process of authoring through the Rule/Example/Recall/Practice quadrants of Merril’s Component Display Theory (Wang-Costello, Tarr, Cintron, Jiang & Goldberg, 2013). The potential to automate the delivery of content based on searchable metadata parameters is one of the key services missing from most of the instructional architectures, but has great potential for content to reach a wide audience quickly.

With the difficult problem of content delivery, the user may be given a simulated environment to practice their obtained knowledge. The integration of intelligent tutoring technologies into systems of practice is not an easy problem, but it is one that is commonly addressed. This integration is a frequent and standard use of the GIFT and Cognitive Tutor systems (e.g., Aleven et al., 2009; Ritter & Koedinger, 1997). However, the current adaptive content (hints, prompts, pumps, etc.) is hand-generated. It may be possible to use the generated tutoring-style information from the previous sections of this work to assist within the practice environment, and eschew the need for expert authoring. As an example, the ordering-based hint “you should multiply before you add” can be generated from the content and used to populate the practice environment.

The above sequence of technologies has the potential to create an adaptive learning system without human intervention. Even substantially diminishing the human workload required would represent a significant savings of time. As part of this overall vision, learning content can be found on the Internet, indexed and sorted into repositories, tagged with searchable metadata information, supplemented with tutoring information, and delivered via browser. The combination of these technologies can allow an instructional system to use an instructional template (e.g., “Rule” content), define user characteristics (e.g., low motivation), match it with intended metadata (e.g., animated/interactive), query a learning system for the appropriate content on the subject (e.g., 4th grade history), and deliver it to the student. Such a combination averts the problem of authoring by reusing existing ITS components for a particular learning objective.

Closing the Authoring Loop: Continuous Feedback and Improvement

Many ITSs have a number of free parameters that must be fixed during the authoring process. For example, in dialogue-based systems, the author must decide how “correct” a student answer should be in order to be counted correct, e.g., must it be exactly the same as the ideal answer or can it be “close enough.” Once fixed, these parameters usually remain fixed until the ITS is overhauled or re-parameterized with new data.

However, ideally, we would close the loop and an ITS would be “self-updating” such that the parameters of the theory of learning would be automatically adjusted to be more optimal as more students used the system. While automated authoring of content involves creating exercises for students to interact with, automated improvement in the pedagogical interactions means modifying the learner model used for pedagogical decision making. For example, a self-updating system may be able to make use of information on population dynamics to provide a “best guess” for model parameters of an unseen student. Such guesses could be updated based upon their effect on learning among groups, allowing broader applicability of the ITSs.

To do such continuous improvement will require a flexible model that characterizes the student learning in the domain. Flexible implies that the model will behave in multiple different ways, depending on how it is configured with parameters or mechanisms. For example, a model might characterize the different outcomes for the student from success and failure with a practice problem. This model can be flexible in its representation of the effect of the success and failure if the model allows this difference to vary, for example, by quantifying success and failure effects numerically. Similarly, a model might characterize forgetting, but again a flexible representation of forgetting would specify that it might range from none at all to very fast depending on some numerical parameter. Again, the model allows for continuous variation in the model space.

Given such a flexible model, one can configure a system with only preliminary settings for the different flexible mechanisms. Following this initial cold start, the system would be designed to be self-tuning, such that the model continuously improves both for groups of students and individual students connected in a server-client architecture. While the network communication and mathematical complexity of this proposal makes it challenging, the possibility for better effectiveness with students in ITSs may also be large. It should also be noted that similar, but conceptually much simpler, A/B tests are now commonly used in industry (e.g., deciding how many search results to put on a page). In the next few paragraphs, we sketch the outlines for such a system.

The system would be controlled by a central server that receives data from the individual clients in order for the server to reestimate parameters. These group estimates of parameters would then be offered to existing and new clients. This system would allow for all the students’ data to be quickly analyzed by the

server to see if the default parameters resulted in a good fit or if they needed to be adjusted. Adjustments would be gradual. Default parameters would thus incrementally evolve on the server for the task, depending on the clients. These default parameters mean that the system will adapt to different contexts of use. For example, a poor performing school district might give rise to parameters that reflect higher forgetting than a better funded district. An adapted system would be expected to promote better learning, since the accuracy of the model effects the accuracy of pedagogical decision making.

In addition to this tracking at the server level, the individual student models would be adjusted at the client level as well. For example, a low performing student might find the task hard, since the system would have adapted to the average student. The client level tracking would correct this inaccuracy very quickly, since the client data would weigh heavily on the model parameters provided by the server. In fact, the server level tracking would function more as a seed for new students than having active effects on a student, minute to minute, supplied by the client level model.

Such capabilities are currently possible but have not been explored greatly. Some systems have been constructed which illustrate this client level tracking of the student model. For example, in the FaCT system experimental software (Pavlik Jr. et al., 2007), the model that controls student actions can be configured to automatically take optimization steps every N number of student practices. After N practices, any particular parameter can be optimized one step either up or down by a specific increment (the step size). This is accomplished by computing the log-likelihood of model fit for the parameter above, below and at the current value. The step size can be specified to determine how fast adaptation occurs. If adaptation occurs too quickly with too little data, pedagogical decisions may fluctuate too wildly.

One problem with this strategy is that often there is not enough variability in the data due to the consistency of the pedagogical decisions. For example, the FaCT system tries to balance correctness and spacing, and generally recommends practice at around 95% correct. Unfortunately, this means that there is little variability in the conditions of the data collection with which to improve the model. One solution to this is to embed small experiments in the tutored practice in order to better measure the parameters in the individual student's model. These embedded randomized trials might be delivered at much wider spacing than the tutor selected items, making them more difficult. Efforts like this to create varying conditions in the tutor data may be necessary to make sure that automated adjustment systems have some variability in the data in order to identify the parameters being optimized as being unique from the other parameters.

Discussion

The new approaches to authoring discussed in this chapter overlap in the problems they are trying to solve. Firstly, there must be content that the user will interact with, whether it is digital characters, simulated environments, or static webpages. This content is usually authored by a subject matter expert (SME) or an instructional design expert (ISD). SimStudent, BrainTrust, and component-based authoring all try to ease the burden of authoring content while still keeping a human “in the loop.”

Secondly, adaptive tutoring systems must have something to adapt to, usually through the modeling of expert and learner knowledge. While this content has traditionally been authored by an SME, SimStudent and BrainTrust speed the authoring of both these models simultaneously, because they compare student actions to an ideal, or expert, answer. Component-based authoring can make use of diverse student models, and a system with continuous improvement and feedback re-parameterizes the student model making best use of the learner data collected to date.

Thirdly, adaptive tutoring systems must contain instruction and feedback to give to the student when diagnosed with a deficiency in a proficiency. These items consist of hints, scenario adaptations, texts, summaries, or other items in response to student actions. There are several efforts at authoring tools which attempt to automate this process. SimStudent uses author demonstrations and collects feedback from the author on the steps SimStudent performs to learn production rules that can be employed to check student solution progress and to generate next step hints when a student is stuck. BrainTrust uses the question-generation technology previously developed for Guru and uses concept maps to generate various kinds of question, e.g., hints, prompts, verification, as well as direct instruction. Similar reusable components are being developed for concept maps and question generation (Robson, Ray & Cai, 2013).

Naturally, all of the items of the ITS system must be delivered through an actual system, which is usually developed through the programming of a simulation or conversational interaction. Both SimStudent and BrainTrust assume specific systems, namely, they produce *expert models* to be used in existing model-tracing and dialogue-based systems. To create other modules (e.g., interface and tutoring modules), other tools (e.g., CTAT) or other system-level programming may be necessary. In contrast, component-based authoring addresses programming more comprehensively by attempting to dynamically assemble systems out of existing components with no additional programming.

The above discussion is summarized in Table 1. Both SimStudent and BrainTrust address the majority of authoring needs but do not squarely address system-level programming. If SimStudent is used as a module within CTAT, then systems-level programming support is provided through the remainder of the CTAT suite. CTAT provides non-programmer authoring tools for interface development and algorithms (model tracing and knowledge tracing) that provide adaptive tutoring support when given the production rules that SimStudent automatically learns. Component-based approaches and continuous improvement, as presented in this chapter, most directly address the authoring needs of programming and assessment, respectively. However, each approach is quite general and could be applied to other authoring needs.

Table 1 Authoring roles addressed by emerging approaches discussed in this chapter.

Authoring Need	Human role	SimStudent	BrainTrust	Component-based	Continuous-improvement
Content	SME & ISD	✓	✓		
Assessment	SME & ISD	✓	✓		✓
Instruction/Feedback	SME & ISD	✓	✓		
Programming	Programmer			✓	

Note. SME: Subject Matter Expert; ISD: Instructional Design Expert

As this chapter has focused on emerging areas of research, it is perhaps no surprise that these areas are operating somewhat in their own silos, motivated by authoring problems in their own ITS traditions. Perhaps a total integration of these approaches may not be possible, given the differences discussed in the introduction and depicted in Figure 1. To that end, it may be preferable for these emerging areas to continue to develop following their own needs, but also more broadly to the needs of their tutoring paradigm, namely, model-tracing, dialogue-based, and constraint-based. If general tools can be made for these quadrants, then in time it may be possible to assemble an integrated suite of tools that, once a paradigm has been selected, afford the greatest degree of automation possible so that ITS learning objectives may be authored completely, accurately, and quickly.

Finally, Figure 1 has an empty quadrant corresponding to path-oriented tutors that indirectly compare student activities to an ideal answer. Whether existing ITS research can properly be located in this quadrant is unclear, but there are several possibilities that may have implications for tutoring in ill-defined domains (Fournier-Viger, Nkambou & Nguifo, 2010; Lynch et al., 2006). In particular, it may be that tutors using case-based reasoning (CBR), such as those used for tutoring the law (Aleven, 2003), fall into this quadrant, because they are both path-oriented and only indirectly compare student input to an expert solution. CBR compares the current situation to previous situations (i.e., cases) and adapts solutions from previous situations to the current problem (Leake, 1996). From this standpoint, CBR may be viewed as representing solution paths in cases, but these paths are ultimately fragments that might be generalized or recombined in a new situation. Comparison to an ideal answer may be indirect because comparison may apply not only to the final solution (as in constraint-based tutoring), but also to whether the solution made use of the same cases and in the same way. If so, then CBR tutors may be an area of research that is currently underdeveloped and amenable to further research in automated authoring.

Recommendations and Future Research

Based on our findings, we can make several recommendations for GIFT and future ITSs. First, the four quadrants of ITS research described in Figure 1 should continue to be developed, with an end goal that the resulting authoring tools may ultimately form a suite of tools that could generally be applied to any problem in their respective tutoring paradigm. As discussed above, however, these approaches are largely building models and do not implement systems-level programming. To assemble new systems from scratch, GIFT should also encompass component-based authoring. This implies that tools operating in the four quadrants should output reusable components, but it further implies that these components must be discoverable and customizable. Finally, we argue that all future ITSs should implement continuous improvement so that the tutor can better adapt to an individual or specific population. As described in this chapter, continuous improvement best aligns with improving of learner models based on interaction data, but it is also conceivable to implement continuous improvement generally for content, assessment, and instruction.

Very impressive performance support tools for ITS authoring already exist (Aleven et al., 2009) and the research described in this chapter does not propose to replace these tools in the near future. Instead, we recommend that such tools continue to incorporate improvements in automated authoring in the research we describe, so that ITS learning objectives may be authored completely, accurately, and quickly. Indeed, it may be the case that some tasks supported by such performance support tools, such as drag-and-drop editors for building ITS graphical interfaces, may never be completely automated. We anticipate that the current generation of ITS authoring tools will instead continue to be enriched by new advances in automated authoring, which will ultimately lower the cost and increase the adoption of ITS.

References

- Aleven, V. (2003). Using background knowledge in case-based legal reasoning: A computational model and an intelligent learning environment. *Artificial Intelligence*, 150(1–2), 183–237. doi:10.1016/S0004-3702(03)00105-X
- Aleven, V., McLaren, B. M., Sewall, J. & Koedinger, K. R. (2009). A new paradigm for intelligent tutoring systems: Example-tracing tutors. *International Journal of Artificial Intelligence in Education*, 19(2), 105-154.
- 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.
- Chase, C.C., Chin, D.B., Oppezzo, M.A. & Schwartz, D.L. (2009). Teachable Agents and the Protégé Effect: Increasing the effort towards learning. *Journal of Science Education and Technology*, 18(4), 334-352.
- Chklovski, T. (2005). Collecting Paraphrase Corpora from Volunteer Contributors. In *Proceedings of the 3rd International Conference on Knowledge Capture* (pp. 115–120). New York, NY, USA: ACM. doi:10.1145/1088622.1088644
- Cycorp. (2005). Factory. <http://game.cyc.com/>. Accessed: 7/23/12.
- D'Mello, S. K., Hays, P., Williams, C., Cade, W., Brown, J. & Olney, A. M. (2010). Collaborative Lecturing by Human and Computer Tutors. In *Intelligent Tutoring Systems* (pp. 178–187). Berlin: Springer.
- Eduworks Corporation. (2014). Re-Usability Support System for eLearning (RUSSEL). from <https://github.com/adlnet/RUSSEL>
- Fournier-Viger, P., Nkambou, R. & Nguifo, E. M. (2010). Building Intelligent Tutoring Systems for Ill-Defined Domains. In R. Nkambou, J. Bourdeau & R. Mizoguchi (Eds.), *Advances in Intelligent Tutoring Systems* (pp. 81–101). Springer Berlin Heidelberg. Retrieved from http://link.springer.com/chapter/10.1007/978-3-642-14363-2_5
- Gagné, R. M. & Gagné, R. M. (1985). *Conditions of learning and theory of instruction*. New York: Holt, Rinehart and Winston.
- Gick, M.L. & Holyoak, K.J. (1983). Schema induction and analogical transfer. *Cognitive Psychology*, 15, 1-38.
- GooruLearning. (2014). <http://www.goorulearning.org>. Retrieved 10/6/2014, 2014
- Graesser, A. C., Chipman, P., Haynes, B. & Olney, A. M. (2005). AutoTutor: An Intelligent Tutoring System with Mixed-Initiative Dialogue. *IEEE Transactions on Education*, 48(4), 612–618.
- Graesser, A. C., D'Mello, S. K., Hu, X., Cai, Z., Olney, A. & Morgan, B. (2012). AutoTutor. In P. McCarthy & C. Boonthum-Denecke (Eds.), *Applied Natural Language Processing: Identification, Investigation, and Resolution*. (pp. 169–187). Hershey, PA: IGI Global.
- Graesser, A. C. & Person, N. K. (1994). Question Asking during Tutoring. *American Educational Research Journal*, 31, 104-137.
- Graesser, A. C., Person, N. K. & Magliano, J. P. (1995). Collaborative dialogue patterns in naturalistic one-to-one tutoring. *Applied Cognitive Psychology*, 9, 1-28.
- Initiative, A. D. L. (2001). Sharable Content Object Reference Model (SCORM™). *Advanced Distributed Learning*, <http://www.adlnet.org>.
- Jesukiewicz, P. & Rehak, D. R. (2011). The Learning Registry: Sharing Federal Learning Resources. Paper presented at the Interservice/Industry Training, Simulation & Education Conference, Orlando, FL.
- Koedinger, K. R., Anderson, J. R., Hadley, W. H. & Mark, M. A. (1997). Intelligent tutoring goes to school in the big city. *International Journal of Artificial Intelligence in Education*, 8, 30-43.
- Koedinger, K.R., Brunskill, E., Baker, R.S.J.d., McLaughlin, E.A., Stamper, J. (2013). New potentials for data-driven intelligent tutoring system development and optimization. *AI Magazine*, 34(3).

- Koedinger, K.R., Corbett, A.C. & Perfetti, C. (2012). The Knowledge-Learning-Instruction (KLI) framework: Bridging the science-practice chasm to enhance robust student learning. *Cognitive Science*, 36 (5), 757-798.
- Landauer, T. K., McNamara, D. S., Dennis, S. E. & Kintsch, W. E. (2007). *Handbook of latent semantic analysis*. Lawrence Erlbaum Associates Publishers.
- Leake, D. B. (1996). *Case-Based Reasoning: Experiences, Lessons and Future Directions* (1st ed.). Cambridge, MA, USA: MIT Press.
- Li, N., Matsuda, N., Cohen, W. & Koedinger, K.R. (2015). Integrating representation learning and skill learning in a human-like intelligent agent. *Artificial Intelligence*.
- Li, N., Stampfer, E., Cohen, W. & Koedinger, K.R. (2013). General and efficient cognitive model discovery using a simulated student. In M. Knauff, N. Sebanz, M. Pauen, I. Wachsmuth (Eds.), *Proceedings of the 35th Annual Conference of the Cognitive Science Society*. (pp. 894-9) Austin, TX: Cognitive Science Society.
- Lynch, C., Ashley, K., Aleven, V. & Pinkwart, N. (2006). Defining ill-defined domains; a literature survey. In *Proceedings of the Workshop on Intelligent Tutoring Systems for Ill-Defined Domains at the 8th International Conference on Intelligent Tutoring Systems* (pp. 1–10). Retrieved from <http://people.cs.pitt.edu/~collinl/Papers/Ill-DefinedProceedings.pdf#page=7>
- MacLellan, C.J., Koedinger, K.R., Matsuda, N. (2014) Authoring Tutors with SimStudent: An Evaluation of Efficiency and Model Quality. *Proceedings of the 12th International Conference on Intelligent Tutoring Systems*. Honolulu, HI. June 5-9, 2014.
- Mangold, L. V., Beauchat, T., Long, R. & Amburn, C. (2012). An Architecture for a Soldier-Centered Learning Environment. Paper presented at the Simulation Interoperability Workshop.
- Martin, B., Mitrovic, T., Mathan, S. & Koedinger, K.R. (2011). Evaluating and improving adaptive educational systems with learning curves. *User Modeling and User-Adapted Interaction: The Journal of Personalization Research (UMUAI)*, 21(3), 249-283. [2011 James Chen Annual Award for Best UMUAI Paper]
- Matsuda, N., Cohen, W. W. & Koedinger, K. R. (2015). Teaching the Teacher: Tutoring SimStudent leads to more Effective Cognitive Tutor Authoring. *International Journal of Artificial Intelligence in Education*, 25, 1-34.
- McTear, M. F. (2002). Spoken dialogue technology: enabling the conversational user interface. *ACM Computing Surveys (CSUR)*, 34, 90-169.
- Mitrovic, A. (2012). Fifteen years of constraint-based tutors: what we have achieved and where we are going. *User Modeling and User-Adapted Interaction*, 22, 39-72.
- Mitrovic, A., Suraweera, P., Martin, B., Zakharov, K., Milik, N., Holland, J. (2006). Authoring constraint-based tutors in ASPIRE. In Ikeda, M., Ashley, K., Chan, T.-W. (eds.), *Proceedings of ITS 2006*. LNCS, vol. 4053, pp. 41–50.
- Mitrovic, A., Martin, B., Suraweera, P., Zakharov, K., Milik, N., Holland, J., McGuigan, N. (2009). ASPIRE: an authoring system and deployment environment for constraint-based tutors. *International Journal of Artificial Intelligence in Education*, 19(2), 155–188.
- Nye, B. D., Graesser, A. C. & Hu, X. (2014). AutoTutor and Family: A Review of 17 Years of Natural Language Tutoring. *International Journal of Artificial Intelligence in Education*, 24(4), 427–469. doi:10.1007/s40593-014-0029-5
- Ohlsson, S. (1992). Constraint-based student modelling. *International Journal of Artificial Intelligence in Education*, 3, 429-447.
- Ohlsson, S. & Mitrovic, A. (2007). Fidelity and Efficiency of Knowledge Representations for Intelligent Tutoring Systems. *Technology, Instruction, Cognition and Learning (TICL)*, 5, 101-132.
- Olney, A. M., Graesser, A. C. & Person, N. K. (2012). Question generation from concept maps. *Dialogue & Discourse*, 3(2), 75-99.
- Olney, A. M., Cade, W. & Williams, C. (2011). Generating Concept Map Exercises from Textbooks. In *Proceedings of the Sixth Workshop on Innovative Use of NLP for Building Educational Applications* (pp. 111–119). Portland, Oregon: Association for Computational Linguistics. Retrieved from <http://www.aclweb.org/anthology/W11-1414>
- Olney, A. M., D'Mello, S., Person, N., Cade, W., Hays, P., Williams, C., Graesser, A. (2012). Guru: A Computer Tutor That Models Expert Human Tutors. In S. Cerri, W. Clancey, G. Papadourakis & K. Panourgia (Eds.), *Intelligent Tutoring Systems* (Vol. 7315, pp. 256–261). Springer Berlin / Heidelberg.
- Olney, A. M., Person, N. K. & Graesser, A. C. (2012). Guru: Designing a Conversational Expert Intelligent Tutoring System. In P. McCarthy, C. Boonthum-Denecke & T. Lamkin (Eds.), *Cross-Disciplinary Advances in Applied Natural Language Processing: Issues and Approaches* (pp. 156–171). Hershey, PA: IGI Global.
- Palinscar, A. S. & Brown, A. L. (1984). Reciprocal teaching of comprehension-fostering and comprehension-monitoring activities. *Cognition and instruction*, 1(2), 117-175.

- Pavlik Jr., P. I., Presson, N., Dozzi, G., Wu, S.-m., MacWhinney, B. & Koedinger, K. R. (2007). The FaCT (Fact and Concept Training) System: A new tool linking cognitive science with educators. In D. McNamara & G. Trafton (Eds.), *Proceedings of the Twenty-Ninth Annual Conference of the Cognitive Science Society* (pp. 1379–1384). Mahwah, NJ: Lawrence Erlbaum.
- Person, N. K., Graesser, A. C., Magliano, J. P. & Kreuz, R. J. (1994). Inferring what the student knows in one-to-one tutoring: The role of student questions and answers. *Learning and Individual Differences*, 6, 205–229.
- Piech, C., Huang, J., Chen, Z., Chuong Do, Andrew Ng & Daphne Koller. (2013). Tuned Models of Peer Assessment in MOOCs. In D'Mello, S. K., Calvo, R. A. & Olney, A. (Eds.), *Proceedings of the 6th International Conference on Educational Data Mining* (pp. 153–160).
- Raykar, V. C., Yu, S., Zhao, L. H., Valadez, G. H., Florin, C., Bogoni, L. & Moy, L. (2010). Learning From Crowds. *Journal of Machine Learning Research*, 11, 1297–1322.
- Ritter, S. & Koedinger, K. R. (1996). An architecture for plug-in tutoring agents. In *Journal of Artificial Intelligence in Education*, 7 (3/4), 315–347. Charlottesville, VA: Association for the Advancement of Computing in Education.
- Robson, R., Ray, F. & Cai, Z. (2013). Transforming Content into Dialogue-based Intelligent Tutors. Paper presented at the The Interservice/Industry Training, Simulation & Education Conference (I/ITSEC), Orlando, FL.
- Roediger, H.L. & Butler A.C. (2011). The critical role of retrieval practice in long-term retention. *Trends in Cognitive Science* 15:20–27
- Rus, V., Stefanescu, D., Niraula, N. & Graesser, A. C. (2014). DeepTutor: Towards Macro- and Micro-adaptive Conversational Intelligent Tutoring at Scale. In *Proceedings of the First ACM Conference on Learning @ Scale Conference* (pp. 209–210). New York, NY, USA: ACM. doi:10.1145/2556325.2567885
- Stamper, J.C. & Koedinger, K.R. (2011). Human-machine student model discovery and improvement using data. In G. Biswas, S. Bull, J. Kay & A. Mitrovic (Eds.), *Proceedings of the 15th International Conference on Artificial Intelligence in Education*, pp. 353–360. Berlin: Springer.
- Tian, Y. & Zhu, J. (2012). Learning from Crowds in the Presence of Schools of Thought. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 226–234). New York, NY, USA: ACM. doi:10.1145/2339530.2339571
- VanLehn, K. (2006). The behavior of tutoring systems. *International Journal of Artificial Intelligence in Education*, 16(3), 227–265.
- Veden, A. (2014). Data for Enabling Content in Adaptive Learning Systems (DECALS). from <https://github.com/adlnet/DECALS>
- Von Ahn, L. (2005). Human Computation (Doctoral thesis). Carnegie Mellon University.
- Wang-Costello, J., Tarr, R. W., Cintron, L. M., Jiang, H. & Goldberg, B. (2013). *Creating an Advanced Pedagogical Model to Improve Intelligent Tutoring Technologies*. Paper presented at the The Interservice/Industry Training, Simulation & Education Conference (I/ITSEC).
- Wylie, R., Koedinger, K. R. & Mitamura, T. (2010). Analogies, explanations, practice: Examining how task types affect second language grammar learning. In V. Aleven, J. Kay & J. Mostow (Eds.), *Proceedings of the International Conference on Intelligent Tutoring Systems* (pp. 214–223). Heidelberg, Berlin: Springer.
- Zhu, X. & Simon, H. A. (1987). Learning mathematics from examples and by doing. *Cognition and Instruction*, 4(3), 137–166.

CHAPTER 20 Developing Conversational Multimedia Tutorial Dialogues

Wayne Ward^{1,2} and Ron Cole¹

¹ Boulder Language Technologies; ² University of Colorado

Introduction

This chapter describes an approach to authoring intelligent tutoring systems (ITSs) used in My Science Tutor (MyST). This virtual science tutor engages children in spoken dialogues in which they learn to construct explanations of science phenomena presented in illustrations, animations, and interactive simulations. Tutorials are developed through an iterative process of recording, annotating, and analyzing logs from sessions with students, and then updating tutor models. This approach has been used to develop over 100 tutorial dialogue sessions, of about 15 minutes each, in 8 areas of elementary school science. Summative evaluations indicate that students are highly engaged in the tutoring sessions and achieve learning outcomes equivalent to expert human tutors (Ward et al., 2011; 2013).

This chapter describes the process of developing conversational science tutors that use visual media and the infrastructure supporting the development. A particular focus is the development of models for representing and extracting the semantics that provide the basis for selecting tutor actions based on interpretations of student answers. While initial evidence suggests that MyST tutorials can improve students' motivation and science learning (Ward et al., 2011; 2013), the potential of these systems to transform learning and education is limited by the amount of effort required to develop them. A major focus of our current research, discussed in this chapter, is to motivate and demonstrate the feasibility of an approach to authoring conversational tutoring systems that substantially reduces the effort and data required to develop dialogues for each new science domain.

Related Research

Research in ITSs addresses a critical need to provide teachers and students with accessible, inexpensive and reliably effective tools for improving young learners' interest in science, as well as their ability to learn science and participate productively in classroom science activities. The 2009 National Assessment of Educational Progress (NAEP 2009) reports that fewer than 2% of 4th, 8th, and 12th grade students demonstrated advanced knowledge of science, and over two-thirds of all students in these grades were scored as *not proficient in science*. Analyses of NAEP scores in reading, math, and science over the past 20 years indicate that this situation is not improving, and is actually worsening. The gap between English learners and English-only students, which is over one standard deviation lower for English learners, has increased rather than decreased over the past 20 years.

ITSs aim to enhance learning by providing students with individualized and adaptive instruction similar to that provided by a knowledgeable human tutor. These systems support conversational interaction with users through either typed or spoken input with the system presenting prompts and feedback via text, human voice, or an animated pedagogical agent (Graesser et al., 2001; D'Mello et al., 2011; Rus et al., 2013; Graesser et al., 2014). Advances in ITSs during the past 15 years have resulted in systems that produce learning gains equivalent to human tutoring, which is widely regarded as the most efficient and effective form of learning. A review by Van Lehn (2011) compared learning gains with human tutoring

and ITSs that required students to engage in problem solving and construct explanations. When compared to students who did not receive tutoring, the effect size of human tutoring across studies was $d=0.79$ whereas the effect size of tutoring systems was $d=0.76$. Van Lehn concluded that ITSs “are nearly as effective as human tutoring systems.” (Van Lehn, 2011, pg. 197). A recent meta-analysis by Ma et al. (2014) indicated that ITSs produce significant effects across a wide range of subjects at all education levels relative to large group instruction, non-ITS computer-based instruction, or textbook or workbooks, and no differences between human tutoring and learning using ITSs (Ma, et al., 2014).

Research in argumentation and collaborative discourse acknowledges the strong influence of the theories of Vygotsky (1978, 1987) and Bakhtin (1975; 1986), who argue that all learning occurs in and is shaped by the social, cultural, and linguistic contexts in which they occur. Roth (2013, 2014) provides an excellent integration of Vygotsky’s and Bakhtin’s theories and their relevance to research on collaborative discourse. He argues that, when considered in the context of the basic tenets of their theories, “currently available analyses of science classroom talk do not appear to exhibit sufficient appreciation of the fact that words, statements, and language are living phenomena, that is, they inherently change in speaking” (Roth, 2014). Vygotsky argued that scientific vocabulary and concepts could only be learned through deliberate instruction in an academic setting, as opposed to the more ad-hoc manner in which vocabulary and concepts are learned in everyday conversation. Consistent with this view, the 2007 NRC report emphasizes that *scientific inquiry and discourse is a learned skill*, so students need to be involved in activities in which they learn appropriate norms and language for productive participation in scientific discourse and argumentation (Duschl et al., 2007).

The past decade has seen a remarkable growth in publications investigating scientific discourse and argumentation. Kuhn (2010) notes that argumentation has become widely advocated as a framework for science education. The idea that argumentation has become both a reform movement and framework for science education is supported by growing evidence of substantial benefits of explicit instruction and practice on the quality of students’ argumentation and learning (Chin & Osborne, 2010; Kulatunga & Lewis, 2013). Evidence from these studies indicates that argumentation can be improved by providing professional development to teachers or knowledgeable students (Bricker & Bell, 2009; Bricker & Bell, 2014; deJong, 2013; Berland, 2009), explicitly teaching students the structure of good arguments, and providing students with scaffolds during argumentation that helps them provide evidence for their own arguments and critiquing other’s arguments (Kulatunga et al., 2013; Kulatunga and Lewis, 2013).

In the remainder of this chapter, the type of interaction used by MyST is described along with the semantic representation used to support the interaction. The process for developing tutorials is explained with a focus on creation and refinement of the model for extracting semantic representations from spoken student responses. A new approach is then presented for developing more robust semantic parsers for the domain with significantly reduced developer effort.

Discussion

The Nature of Tutorial Dialogues between Students and Marni in MyST

Since 2007, our research has focused on development of MyST, an ITS designed to improve science learning of 3rd, 4th and 5th grade children through spoken dialogues with Marni, a virtual science tutor. Because many elementary school children have difficulty reading at grade level, we decided to develop tutoring systems in which students use speech to converse with a virtual tutor. Students in our study received eight to ten weeks of classroom instruction in one of four areas of science—measurement, water, magnetism and electricity, or variables—using the Full Option Science System (FOSS, 2014). Over the course of each FOSS module instruction, students conducted 16 science investigations in small groups.

Students made written entries and drawings in science notebooks about their predictions, observations and explanations of the science encountered in each investigation. Shortly after each investigation, students engaged in spoken dialogues for 15 to 20 minutes with the virtual tutor Marni or with an expert human tutor. In these dialogues, the human or virtual tutors asked open-ended questions about the science encountered in the classroom science investigations. The tutors asked students questions about science presented in illustrations, animations, or interactive simulations to scaffold learning and help them construct accurate and complete explanations. Analyses of dialogues indicate that, during a dialogue of about 15 minutes, tutors and students produced about the same amount of speech, around 5 minutes each. The main result of the summative evaluation was that, relative to students in classrooms who did not receive supplemental tutoring, students who were tutored by Marni and by human tutors achieved equivalent learning gains, with moderate to strong effect sizes. Surveys indicated that over 70% of students tutored by Marni reported that they were more excited about studying science in the future. Details of these experiments are reported in Ward et al. (2011, 2013).

It is noteworthy that tutoring by both human and virtual tutors produced significant learning gains, relative to students who did not receive tutoring, given that all students in the study received classroom instruction using a highly respected inquiry-based learning program (FOSS, 2014) that is used by over 1 million K-8 students annually in the US. These results are consistent with a meta-analysis by Chi (2009), which indicates that students whose instruction involves *interactive tasks* that include collaborative discourse and argumentation learn more than students whose learning involves *constructive tasks*, (e.g., classroom investigations and written reports) or *active tasks* (e.g., classroom Science Investigations). Chi's synthesis of research indicates the critical importance of having students talk about and explain science to optimize learning in inquiry-based programs.

When using MyST, the student's computer shows a full screen window that contains the virtual tutor Marni (a 3D character), a display area for presenting information and a display button that indicates the listening status of the system. The agent's lips and facial movements are synchronized with her speech, which is recorded by an experienced science tutor, the voice talent whose phrasing and prosody imbues Marni with the personality of a sensitive and supportive tutor. Spoken dialogues involve Marni asking open-ended questions about science presented in illustrations, silent animations and interactive simulations. Interactive simulations allow students to use a mouse to manipulate variables and observe the effects, such as adding additional winds of wire to an electromagnet core and observing the effect on the number of washers picked up. The pedagogical role of these media types are discussed in detail in Ward et al. (2011). Figure 1 shows a screen shot of the student's screen for the example interactive.

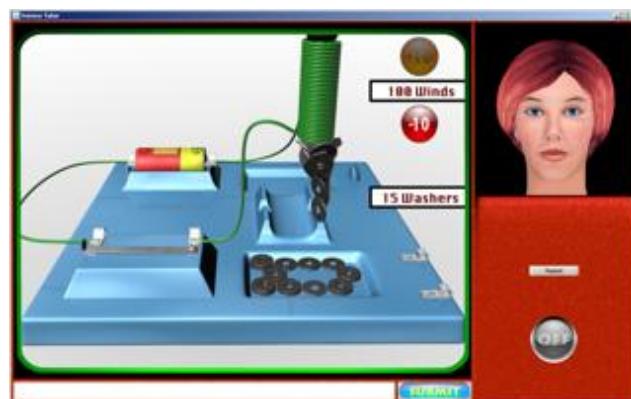


Figure 1: The student screen contains the avatar Marni, a display area, and a listening indicator.

A typical sequence of actions for the tutor would be to introduce a Flash animation (“Let’s look at this.”), display the animation, and then ask a question (“What’s going on there?”). Depending on the nature of the question and the media, the student may interact with content in the display area, watch a movie, or make passive observations. Students wear high quality headphones with a noise-cancelling microphone. When ready to speak, the student holds down the space bar. As the student speaks, the audio data are sent to the speech recognition system. When the space bar is released, the word string produced by the speech recognizer is parsed to produce a set of semantic parses. The set of parses is pruned using session context information to a single best interpretation. The new information is added to the session context and a new set of tutor actions is generated. The actions are executed and the system again waits for a student response.

The focus of the MyST system is to elicit explanations of science concepts from students. Each 15 to 20 minute MyST dialogue session functions as an independent learning activity that provides, to the extent possible, the scaffolding required to stimulate students to think, reason, and talk about science during spoken dialogues with the virtual tutor. The goal of these *multimedia dialogues* is to help students construct explanations that express their ideas. The dialogues are designed so that over the course of the conversation with Marni, the student is able to reflect on their explanations and refine their ideas in relation to the media they are viewing or interacting with, leading to a deeper understanding of the science they are discussing. It is necessary to design dialogues that (1) engage students in conversations that provide the system with the information needed to identify gaps in knowledge, misconceptions, and other learning problems; and (2) guide students to arrive at correct understandings and accurate explanations of the scientific processes and principles. A related challenge is to decide when students need to be provided with specific information (e.g., a narrated animation) in order to provide the foundation or context for further productive dialogue. Students sometimes lack sufficient knowledge to produce satisfactory explanations, and must therefore be presented with information that provides a supporting or integrating function for learning, such as brief multimedia presentation that explains the key concepts the student was attempting to explain.

MyST tutorials are characterized by two key features: the inclusion of media throughout the dialogue and the use of open-ended questions related to the phenomena and concepts presented via the media. Follow-on questions attempt to build on things the student said. For example, an initial classroom investigation about magnets has students move around the classroom exploring and writing down what things do and do not stick to their magnets. The subsequent multimedia dialogue with Marni begins with an animation that shows a magnet being moved over a set of identifiable objects, which picks up some of the objects but not others. Marni then says: “What’s going on here?” If the student says: “The magnet picked up some of the objects,” Marni might say: “Tell me more about the types of objects magnets pick up.”

Each tutorial session in MyST is designed to cover a few main points (typically two to four) in a 15 to 20-minute session with a student. The tutorial dialogue is designed to get students to articulate concepts and be able to explain processes underlying their thinking. Tutor actions are designed to encourage students to share what they know and help them articulate why they know what they know. For the system (Marni), the goal of a tutorial session is to elicit responses from students that show their understanding of a specific set of points, or more specifically, to *entail a set of propositions*. Marni attempts to elicit the points by encouraging self-expression from the student. Many dialogue moves are adapted from principles of questioning the author (QtA) (Beck & McKeown, 2006). Much use is made of open-end questions such as “What do you think is going on here?” One of the developers of QtA, Margaret McKeown, worked closely with our development team during development of MyST dialogues. Dr. McKeown analyzed annotations of sessions with human tutors trained in QtA dialogue moves, and provided feedback that were used to improve subsequent dialogues. Analysis of MyST dialogues (Ward et al., 2011; 2013) reveals that concepts expressed by students are recognized at about 85% accuracy. The

system fails to recognize about 15% of the concepts correctly expressed by the student. MyST does not tell students that they are wrong, but simply moves on to other propositions if the student expressed understanding, or continues to discuss the current topic otherwise. This strategy provides for graceful dialogues when concept recognition errors occur.

Semantic Representation

The MyST dialogue model is based on representing what students are saying about attributes of entities and how entities and events in the domain are related. MyST uses the Phoenix system for natural language processing and generating tutor moves. Phoenix represents the propositions being discussed as semantic frames with role labels similar to other semantic parsing systems such as FrameNet (Baker et al., 1998) and PropBank (Palmer et al., 2005), but uses role labels specific to the domain of Science. Roles represent how entities are related to each other and to predicates (usually a verb or nominalization). Semantic frames are used to represent role sets important for the domain. The following example of a statement describing movement would be extracted as follows:

- *Electricity flows from the negative terminal through the bulb and to the positive terminal.*
 - Frame: DescribeMovement
 - Predicate: Move
 - Theme: Electricity
 - Source: Terminal.negative
 - Goal: Terminal.positive
 - Path: Bulb

Other examples of frames important in science discourse are the following:

- *Grass is a producer.*
 - Frame: ClassMembership
 - Member: Grass
 - Class: Producer
- *The bulbs are not shining because the pathway for electricity to flow has been broken.*
 - Frame: CausalRelation
 - Result:
 - Theme: Bulb
 - State: Off
 - Cause:

- Predicate: Interrupted
- Theme: Pathway

Student responses are extracted by the system into semantic frames. Tutor next moves are selected by comparing the frames extracted from student responses to reference frames representing correct role assignments. The following sections explain how role extraction is accomplished and how the extracted frames are used in generating tutor moves.

Defining and Extracting Semantic Frames

The first step in developing a MyST tutorial dialogue is to define the topics to be covered. The specification of tutorial semantics begins with creating a narrative. The tutorial narrative is a set of natural language statements that express the concepts to be discussed in as simple a form as possible. These do not represent the questions that the system asks, but are the set of points that the student should express. The narrative represents what an ideal explanation from a student would look like. The narrative statements are manually annotated to reflect the desired semantic parse. An example annotation is as follows:

- *The current flows from the minus terminal to the plus.*
 - Theme: [Electricity] (The current)
 - Predicate: [Move] (flows)
 - Source: from the [_negative] (minus terminal)
 - Goal: to the [_positive] (plus)
 - Which results in the extracted frame:
 - Theme: Electricity
 - Predicate: Move
 - Source: negative
 - Goal: positive

These parsed statements define the domain of the tutorial. After enumerating the concepts to be discussed, the visuals to be used to illustrate scientific vocabulary, materials, and phenomena and are defined. A short narrative is written and parsed for each of the media files to be used in the tutorial. The Phoenix compiler is used to compile the annotated narratives into recursive transition networks that are used by the parser to extract text into semantic frames.

Student responses are also parsed into the same semantic representations as the narratives. The initial patterns are created from the narratives and have all of the roles and entities that will be discussed, but only a few ways of expressing them. Over the course of development, the patterns must be expanded to cover the various ways students articulate their understandings of the science concepts. In developing the MyST system, project tutors were asked to type simulated student input. These inputs were annotated and added to the training data for the extraction patterns. Once the initial components for a tutorial have been

specified, the task becomes to obtain coverage in the extraction patterns of all of the ways in which the semantics are expressed by students. As the system is used, it logs all transactions and records student speech. When tutorials are deployed for live use, all session data are uploaded to a server each night. The data are processed automatically to assess system confidence in the interpretation of student responses. Using an active learning paradigm, low confidence sessions are selected for transcription and annotation. Once annotated, the data are added to the training set and system models (acoustic models, language models and extraction patterns) are retrained. Periodically, data are sampled for test sets and a learning curve is plotted for each module. All elements of this process are automatic except for transcription and annotation.

Generating Tutor Moves

The virtual tutor has a set of resources to conduct the session dialogue; synthesized prompts, recorded prompts, narrations, static visuals, silent animations, narrated animations, and interactive simulations. The tutor model controls how the resources for each tutor turn are selected. Features used for move selection include a semantic representation of the last prompt, whether the student reply was responsive to the prompt, and a comparison between the extracted representation from student responses and the reference representation from the narrative. These features generally express whether each target frame role (a) hasn't been addressed, (b) has been prompted for but not answered, (c) has been expressed incorrectly, or (d) has been expressed correctly. Boolean expressions of features are used to select the next tutor move. Tutor moves are sequences of the basic tutor actions: speak(play a recorded audio file), synthesize(a specified word string), flash(execute Flash application), and play(static media file or recorded video). Production rules in the form of Boolean expressions of features are associated with a sequence of actions to be taken by the tutor if the rule evaluates true. Some example pattern-action rules are as follows:

```
# last student response indicated boredom
Response == "boredom"
    Action: "synth(So, I have to be entertaining every minute? You try it some time.)"

# Got it all right, give positive feedback and re-state
Origin == Reference:Origin AND Destination == Reference:Destination
    Action: "synth(Excellent observations!");
        synth(So, electricity is flowing from the negative end of the battery
              and back to the positive end of the battery)"

# origin wrong
Origin != Reference:Origin
    Action: "synth(Let's take a look at something together. Look at the flow of electricity.
              What do you notice about which end the electricity is flowing away from?)"
```

Templates are created for interaction types to make authoring of dialogue interactions more efficient. For example, when discussing word definitions, set membership, and causal relations, very similar dialogue sequences are used regardless of specific content. This is especially true of the introductory parts of each concept, where very open-ended prompts are used. *Tell* types of moves introduce a concept and present a narrated animation. *Elicit* type moves might make an opening statement to segue into a concept, present a silent animation and ask “What’s going on here?” Elicitation of explanation of a causal relationship might use a scenario using and interactive simulation. Ask “What do you think would happen if …”, then have the student try it in the simulation and then explain their observation.” The specific predicates and entities are different, but the interaction pattern is very similar.

During initial development and testing of dialogues, synthetic speech is used in the virtual tutor to allow easy modification. The application could use synthesis in field use, but we generally choose to have prompts recorded by a voice talent before students engage with Marni. This is a viable option since prompts for a session are known in advance and we have an efficient procedure for recording them. System tools generate the set of sentences to be recorded and a recording application is provided to efficiently manage recording and verifying each prompt, as well as the accuracy of the alignment of the speech to the movements of Marni's lips and associating each audio file with the word string. The tools also automatically produce a task control file where all synth(word string) actions have been replaced with play(recorded file) actions.

Summary of Current MyST Tutorials Dialogue Development Process

The primary activities involved in the development of MyST tutorial sessions are developing Flash media; authoring feature expressions and associated action sequences; and annotating data for extracting semantic representations. Templates of interaction types are used to reduce the effort of creating new tutor models. An efficient process is in place for collecting and annotating data and re-training system models. Fifty tutorial sessions were developed in four months by a small team (one project manager, two digital artists, and two linguistics students).

That optimistic assessment notwithstanding, substantial effort is required to develop and tune multimedia conversational tutorials. Less expensive media can be substituted for Flash animations, but the media is so integral to the presentation that we feel the expense justified. The other labor-intensive effort is the annotation of extraction patterns. The next section details a proposal for reducing the data and effort required for training the semantic extraction model.

Applying Linguistic Resources to Semantic Extraction

One of the more costly and time-consuming aspects of developing a tutorial with this model is achieving good coverage in the extraction patterns used in parsing. The semantics of the domain are constrained, but student responses can vary greatly in the ways they choose to express concepts and terms. An efficient process is in place for collecting data and training the system, but the first time the system sees a construct it has not seen before, it does not extract it correctly. It still takes time, effort, and data to get good coverage of student responses.

The patterns are used to extract (and normalize) entities into semantic roles, and thus represent both patterns for entity recognition and higher-level patterns assigning the entities to roles. Entity patterns represent the set of phrases considered to be an acceptable synonym for a term. Electricity could be expressed as *electricity*, *energy*, *power*, *current*, or *electrical energy*. Coverage of term synonyms from annotated data is achieved fairly quickly and easily and can be done by most anyone familiar with the domain. The larger problem is the patterns discriminating between possible role assignments. Not only is there more disfluency and variability here, annotating them is a more difficult task for someone not trained to do it.

One possibility for increasing robustness of extraction patterns and reducing data (and effort) needed to achieve coverage for role assignment is to use output from a domain-independent semantic role labeling (SRL) system to help with role assignment. The Proposition Bank (PropBank) provides a corpus of sentences annotated with domain-independent semantic roles (Palmer, et al.). PropBank has been widely used for the development of machine learning based SRL systems. Pradhan et al. (2005) used the representation in open domain question answering and Albright et al. (2013) extended PropBank for processing clinical narratives. The idea is not to try to use PropBank output directly to produce the

extracted representations, but to map PropBank SRL output onto MyST frames domain-specific entity patterns will still need to be applied to produce the canonical extracted form, but this is a much simpler task than role assignment and one more suited to non-linguists.

An initial investigation has been conducted to examine how well the semantic frames used in MyST can be produced from PropBank roles. Many of the roles can be mapped directly, such as class membership. In some cases, such as causal relations between two events, several PropBank predicates are involved in producing the MyST frame. PropBank parses are oriented around a predicate and separate parses are produced for each predicate. These need to be unified to produce the MyST frame. An example of a Propbank parse that maps directly is as follows:

All metals are conductors

PropBank	MyST
Predicate: are	Frame: ClassMembership
A1: metals	Member: metals
A2: conductors	Class: conductors

And an example of one that is not so direct is:

When the switch is closed electricity flows

PropBank	MyST
Predicate: flow	Frame: CausalRelation
A1: electricity	Cause: SwitchState: closed
TMP: when the switch is closed	Result: ElectricalFlow: on

The MyST patterns produce the *SwitchState: closed* and *ElectricalFlow: on* elements. The mapping issue is that Propbank treats *When the switch is closed* as a temporal expression while the MyST frame treats it as a pre-condition (the *Cause* role covers both cause and pre-condition concepts). As the number of frames in a MyST tutorial is small, generally less than 20, rule based mapping of Propbank predicates and roles to MyST frames seems feasible.

In MyST, many different related predicates share the same frame. Students could say electricity *flows*, *goes*, *runs*, *races*, *zooms*, or *circles*, and the important elements are *what* is moving, *from where*, *to where*, irrespective of the choice of verb. The goal is to map PropBank predicates that share similar role sets onto a common MyST frame to provide general ways of talking about the event participants e.g., a set of patterns for talking about roles in motion events. The following two sentences describe motion in two very different domains, but use the same semantic frame for representing the meanings:

Electricity is flowing from the negative terminal to the positive.

Predicate: Move
Theme: Electricity
Source: from the negative terminal
Goal: to the positive

The clouds are blowing from the west to the east.

Predicate: Move
Theme: clouds
Source: from the west
Goal: to the east

In MyST, the recognition and clustering of predicates is done by the extraction patterns. As an example, the predicate term *Move* might have synonyms, move, flow, and circle around. This gives no guidance of what to do when a new predicate is encountered. For example, suppose a student says *Electrons are zipping around in a circle*, and the system has never encountered the word *zipping*. The extraction patterns do not indicate that *zipping* is a form of movement. A saving grace of the system is that a predicate is not required to extract into a frame. The system produces the set of possible extracted frames and uses context to disambiguate between competing alternatives. As long as the role assignments are not ambiguous (as in Source and Goal) it is often able to perform the semantic frame extraction correctly. Sometimes however, extraction patterns for roles do not cover the construction used by the student. Incorporating PropBank parses offers the possibility to save considerable annotation effort by doing role assignment in a domain-independent way so that extraction patterns are mostly only required to add structure to and normalize entities. It is expected that some MyST frames might not have a useful mapping from PropBank roles and will still require extraction patterns, but that most can be mapped from PropBank. At the current time, there is no quantitative data to support this, only a pilot investigation.

Adapting PropBank to Domain and Genre

Even though PropBank uses a domain-independent representation, machine learning based systems trained on it will necessarily be learning aspects of the topic and genre used in the training data. Initial PropBank training data were sentences taken from the Wall Street Journal and the Brown Corpus, both fluent written text. When PropBank-trained SRL systems were applied to clinical narratives in the medical domain, both the genre of dictated notes and idiosyncratic word usage in the medical domain were very different from the original training data, which lowered performance (Albright et al., 2013). Parser performance was enhanced significantly by annotating a modest amount of data in the new domain with PropBank labels.

None of the available PropBank corpora are a good match to either topic or genre for children's conversational speech on science. There currently is no large corpus available that is appropriate for training PropBank parsers for spoken dialogue based science tutorials for children. Boulder Language Technologies is beginning the work of annotating data collected in the MyST project to provide such a resource, representing over 1000 hours of speech from over 1200 elementary school students.

Recommendations and Future Research

While most of the mechanisms in the MyST framework are similar to capabilities that are already contained in the Generalized Intelligent Frameworks For Tutoring (GIFT), we believe that the extraction and use of domain-specific semantic roles can provide complementary information to the current set of features being used. The functions for annotating data, training extraction patterns, and extracting semantic frames could easily be integrated into the GIFT framework and the features derived from them made available as additional information within the current framework. The tools for selecting data for new annotations to add to the training data and evaluating component performance can be used to expand the representation as the systems evolve over time.

Boulder Language Technologies will make all of the components of the MyST system available for research use, including the Bavieca Automatic Speech Recognition engine, Phoenix Natural Language Processing engine, and a character animation system. Many of these components are trained from data, and both supervised and unsupervised training can improve the models. Many projects have benefitted from the sharing of data within a research community. An example is the Linguistic Data Consortium, which serves as a repository and distribution center for corpora. The availability of corpora reduces the entry barrier to new research efforts to improve the technology. When corpora are available, common

tasks can be defined and common evaluations conducted to accelerate progress in the field. The availability of data tends to attract new researchers. We recommend that providing methods for sharing data by GIFT users, including common annotation guidelines and assessment conventions, be considered.

References

- Albright, D., Lanfranchi, A., Fredriksen, A., Styler, W., Warner, C., Hwang, J., Choi, J., Dligach, D., Nielsen, R., Martin, J., Ward, W., Palmer, M., Savova, G. (2013). Towards comprehensive syntactic and semantic annotations of the clinical narrative. *JAMIA*, 20(5), 922-930.
- Baker, C., Fillmore, C. & Lowe, J. (1998). The Berkeley FrameNet project. In Proceedings of the COLING-ACL, 86-90.
- Bakhtin, M. (1975). *The dialogic imagination*. Austin, TX, University of Texas Press.
- Bakhtin, M. (1986). *Speech genres and other late essays*. Austin, Tx, University of Texas Press.
- Beck, I. & McKeown, M. (2006). *Improving comprehension with Questioning the Author: A fresh and expanded view of a powerful approach*. New York: Scholastic.
- Berland, L. & Reiser, B. (2009). Making sense of argumentation and explanation. *Science Education*(93), 26.
- Bricker, L. & Bell, P. (2009). Conceptualizations of argumentation from science studies and the learning sciences and their implications for the practices of science education. *Science Education*, 82, 473-498.
- Bricker, L. A. & Bell, P. (2014). What comes to mind when you think of science? The perfumery!: Documenting science-related cultural learning pathways across contexts and timescales. *Journal of Research in Science Teaching*, 51(3), 260-285. doi: 10.1002/tea.21134.
- Chi, M.T.H. (2009) Active-contructive-interactive: a conceptual framework for differentiating learning activities. *Topics in Cognitive Science*, 1:73-105.
- Chin, C. & Osborne, J. (2010). Students' questions and discursive interaction: Their impact on argumentation during collaborative group discussions in science. *Journal of Research in Science Teaching*, 47(7), 883-908. doi: 10.1002/tea.20385.
- deJong, L., Zacharia (2013). Physical and virtual laboratories in science and engineering education. *Science*, 340(305).
- Duschl, R. (2008). Science education in three-part harmony: Balancing conceptual, epistemic, and social learning goals. *Review of Research in Education*, 32, 268-291.
- Duschl, R., Schweingruber, H. & Shouse, A. (2007). *Taking science to school: Learning and teaching science in grades K-8*: National Academy Press.
- Erduran, S. & Aleixandre, M. (2008). *Argumentation in science education: perspectives from classroom-based research*: Springer.
- Graesser, A. C., VanLehn, K., Rosé, C. P., Jordan, P. W. & Harter, D. (2001). Intelligent tutoring systems with conversational dialogue. *AI Magazine*, 22(4), 39-51.
- Kelly, G., Regev, J. & Prothero, W. (2008). Analysis of lines of reasoning in written argumentation. In S. Erduran & M. P. Jimenez-Aleixandre (Eds.), *Argumentation in science education: Perspectives from classroom-based research*. New York: Springer.
- Kuhn, D. (1993). Science as argument: Implications for teaching and learning scientific thinking. *Science Education*, 77(3), 319-337.
- Kuhn, D. (2010). Teaching and learning science as argument. *Science Education*, 94, 810–824. doi:10.1002/sce.20395.
- Kulatunga & Lewis. (2013). Exploration of peer leader verbal behaviors as they intervene with small groups in college chemistry. *Chemistry Education Research and Practice*, 14, 576-588.
- Kulatunga, U., Moog, R. S. & Lewis, J. E. (2013). Argumentation and participation patterns in general chemistry peer-led sessions. *Journal of Research in Science Teaching*, 50(10), 1207-1231. doi: 10.1002/tea.21107
- Lehrer, R., Schauble, L. & Lucas, D. (1998). Supporting development of the epistemology of inquiry. *Cognitive development of mental representation - theories and applications*, 23, 512-529.
- Lehrer, R., Schauble, L. & Petrosino, A. J. (2001). Reconsidering the role of experiment in science education. In K. Crowley, C. Schunn & T. Okada (Eds.), *Designing for science: Implications from everyday, classroom, and professional settings* (pp. 251-277). Mahwah, NJ: Erlbaum.

- Lester, J. C., Converse, S. A., Kahler, S. E., Barlow, S. T., Stone, B. A. & Bhogal, R. S. (1997). *The persona effect: affective impact of animated pedagogical agents*. Paper presented at the Proceedings of the SIGCHI conference on Human factors in computing systems, Atlanta, Georgia.
- Ma, W., Adesope, O., Nesbit, J. & Liu, Q. (2014) Intelligent tutoring systems and learning outcomes: A meta-analysis. (2014). *Journal of Educational Technology*, 106, 901-918.
- McNeill, K. L. (2011). Elementary students' views of explanation, argumentation, and evidence, and their abilities to construct arguments over the school year. *Journal of Research in Science Teaching*, 48(7), 793-823. doi: 10.1002/tea.20430
- McNeill, K., Lizotte, D., Krajcik, J. & Marx, R. (2006). Supporting students' construction of scientific explanations by fading scaffolds in instructional materials. *Journal of the Learning Sciences*, 15(2), 153-191.
- Mostow, J. & Aist, G. (2001). Evaluating tutors that listen: an overview of project LISTEN. In K. Forbus & P. Felтовich (Eds.), *Smart machines in education* (pp. 169-234). MIT Press.
- NAEP (2009), National and state reports in science *The nations report card: National assessment of educational progress* from <http://nces.ed.gov/nationsreportcard>
- Naylor, S., Keogh, B. & Downing, B. (2007). Argumentation and primary science. *Research in Science Education*, 37(17), 39.
- Nussbaum, E., Sinatra, G. & Poliquin, A. (2008). Role of epistemic beliefs and scientific argumentation in science learning. *International Journal of Science Education*, 30, 1977-1999.
- Osborne, J., Erduran, S. & Simon, S. (2004). Enhancing the quality of argumentation in school science. *Journal of Research in Science Teaching*, 41(10), 994-1020.
- Palmer, M., Gildea, D. & Kingsbury, P. (2005). The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1), 71-106.
- Pradhan, S., Hacioglu, K., Krugler, V., Ward, W., Martin, J., & Jurafsky, D. (2005). Support vector learning for semantic argument classification. *Machine Learning*, 60(1), 11-39.
- Roth, W.-M. (2013). An integrated theory of thinking and speaking that draws on Vygotsky and Bakhtin/Vološinov. *Dialogical Pedagogy*, 1, 32-53.
- Roth, W.-M. (2014). Science language *Wanted Alive*: Through the dialectical/dialogical lens of Vygotsky and the Bakhtin circle. *Journal of Research in Science Teaching*, 51, 1049-1083. DOI: 10.1002/tea.21158
- Sampson, V. & Clark, D. (2008). Assessment of the ways students generate arguments in science education: Current perspectives and recommendations for future directions. *Science Education*, 92(3), 447-472.
- Sampson, Grooms, J. & Walker, J. (2009). Argument-Driven Inquiry: A way to promote learning during laboratory activities. *The Science Teacher*, 76(7), 42-47.
- Schworm & Renkle. (2007). Learning argumentation skills through the use of prompts for self-explaining examples. *Journal of Educational Psychology*, 99(2), 285-296.
- Simon, S., Erduran, S. & Osborn, J. (2006). Learning to teach argumentation: Research and development in the science classroom. *International Journal of Science Education*, 235-260.
- VanLehn, K. (2011). The relative effectiveness of human tutoring, intelligent tutoring systems and other tutoring systems. *Educational Psychologist*, 46(4), 197-221.
- Voss, J. & Means, M. (1991). Learning to reason via instruction in argumentation. *Learning and instruction*, 1(337-350).
- Vygotsky, L. S. (1978). *Mind in society: The development of higher psychological processes*. Cambridge, MA: Harvard University Press.
- Vygotsky, L.S. (1987) Thinking and Speech. In R.W. Rieber & A.S. Carton (Eds.) *The collected works of L.S. Vygotsky, Vol. 1, Problems of general psychology*. (N. Minick, Trans.) (pp.39-285) New York: Plenum Press.
- Ward, W., Cole, R., Bolanos, D., Buchenroth-Martin, C., Svirsky, E., van Vuuren, S., Weston, T. & Zheng, J. (2011), My science tutor: A conversational multi-media virtual tutor for elementary school science. *ACM Transactions on Speech and Language Processing*, 7(4).
- Ward, W., Cole, R., Bolaños, D., Buchenroth-Martin, C., Svirsky, E. & Weston, T. (2013), My science tutor: A conversational multimedia virtual tutor. *Journal of Educational Psychology*, 105, 1115-1125.doi: 10.1037/a0031589
- Wise, B., Cole, R., Van Vuuren, S., Schwartz, S., Snyder, L., Ngampatipatpong, N., Pellom, B. (2005). Learning to read with a virtual tutor: foundations to literacy. In C. Kinzer & L. Verhoeven (Eds.), *Interactive Literacy Education*, Lawrence Erlbaum, Mahwah, NJ
- Zohar, A. & Nemet, F. (2002). Fostering students' knowledge and argumentation skills through dilemmas in human genetics. *Journal of Research in Science Teaching*, 39(1), 35-62. doi: 10.1002/tea.10008.

SECTION V

INCREASING INTEROPERABILITY AND REDUCING WORKLOAD AND SKILL REQUIREMENTS FOR AUTHORIZING TUTORS

Robert Sottilare, Ed.

CHAPTER 21 Approaches to Reduce Workload and Skill Requirements in the Authoring of Intelligent Tutoring Systems

Robert A. Sottilare
US Army Research Laboratory

Introduction

The effectiveness of intelligent tutoring systems (ITSs) as an instructional tool makes them an attractive choice for one-to-one instruction as compared to traditional classroom training (VanLehn, 2011; VanLehn, et al., 2005; Lesgold, Lajoie, Bunzo & Eggan, 1988). Limiting factors in their adoption are workload and skill requirements. Even for well-defined domains, the authoring process for ITSs is both complex and time consuming. A major goal for the Generalized Intelligent Framework for Tutoring (GIFT; Sottilare, Brawner, Goldberg & Holden, 2012; Sottilare, Holden, Goldberg & Brawner) is to integrate tools and methods that reduce the time/cost, workload, and skill requirements to author adaptive tutoring systems.

The ITS community has identified several goals associated with ITS authoring processes (Murray, 1999; Murray, 2003; Sottilare and Gilbert, 2011; Sottilare, Goldberg, Brawner, and Holden, 2012; Sottilare, 2013; and Sottilare, 2015). We have organized these goals into four key categories. The chapters in this section reinforce these goals across various authoring systems and various ITS genres. Research is needed to discover and innovate authoring tools and methods to accomplish the following:

- decrease the effort required by the author
- decrease the knowledge required by the author
- support the organization of domain knowledge
- enable rapid evaluation of prototypes

Tools and Methods to Decrease Authoring Burden

Aleven, McLaren, Sewall, and Koedinger (2006) asserted that it takes approximately 200–300 hours of development time to author 1 hour of adaptive instruction. Sottilare (2015) indicated that the progress of authoring system capabilities may have reduced this burden to about 100–200 hours, but this is still far from being practical for teachers/instructors and course managers who may need to develop new content on a weekly or perhaps daily basis. To be agile in meeting changing demands to update domain knowledge, the goal for authoring 1 hour of adaptive instruction should be about 4 hours (threshold) with an objective of 1 hour.

To meet this lofty goal, we have identified two supporting objectives:

- create community-based standards for interoperability

- create tools to automate large portions of the authoring process and remove the human author from the process

Creating Community-Based Standards for Interoperability

By either creating or adopting existing interoperability standards for ITSs, we will increase opportunities for reuse of essential ITS elements and drive the community’s need for authoring down, thereby reducing the authoring burden. In previous sections of this volume, ITS genres (e.g., model tracing, agent-based, and dialogue-based) are examples authoring tools with academic, commercial, and governmental origins. While there are many more authoring tools, below are four toolsets with active user bases:

- ***Cognitive Tutor Authoring Tools (CTAT)*** produces cognitive modeling and example-tracing tutors; developed by Carnegie-Mellon University.
- ***Authoring Software Platform for Intelligent Resources in Education (ASPIRE) Authoring Tools*** produces constraint-based online tutors; developed by the University of Canterbury (New Zealand).
- ***AutoTutor Script Authoring Tools (ASAT)*** produces dialogue-based tutors; developed by the University of Memphis.
- ***Generalized Intelligent Framework for Tutoring (GIFT)*** produces various types of tutors; developed by the US Army Research Laboratory (ARL).

We recommend decreasing the effort to author ITSs by establishing and documenting standards for processes, tools, and integration of components. Features for some existing authoring tools such as those listed above may be ready-made candidates for ITS standards. Templates for the development of domain models and content may also reduce the effort required to author ITSs.

While we may never reach a single standard ITS, it is possible and beneficial for the community to rally around interoperability standards for integration, modular components, and metadata. Interoperability standards will support rapid integration of standalone training and education platforms (e.g., serious games, virtual simulations, presentation content, and other domain knowledge) with ITSs to promote multi-domain training platforms with tailored tutoring. Interoperability standards may also allow for movement of modular domain knowledge from one tutoring platform to another. Metadata standards may also allow for easier curation (search, retrieval, and archiving) of domain knowledge.

Finally, we advocate the use of web services as a standard to support integration with external capabilities. Web service calls are data driven and therefore largely domain-independent. For example, in recent releases of GIFT, ARL implemented calls to external AutoTutor web services. Web services available through GIFT support AutoTutor dialogue-based tutoring including latent semantic analysis (LSA) of text to support near-real-time analysis of learner essay responses; conversational dialogues based on LSA assessments; interfaces to animated agents (e.g., commercial virtual humans); and various other tutoring and delivery style mechanisms. The use of web services reduces the author’s workload by reducing integration effort to service calls by the ITS.

Automating the Authoring Process

By understanding, modeling, and then automating authoring processes, we can lower the authoring load and knowledge required to author ITSs. A design goal for GIFT is to be able to provide authoring tools

suitable for domain experts who may lack computer programming and instructional design skills. Two emerging technologies include *automated integration for serious games and ITSs* and *automated authoring of expert models*.

As mentioned previously, the opportunity to automate the integration of games and tutors will combine the higher levels of engagement found in serious games with the effective instructional techniques found in ITSs. The goal is to reduce authoring by automating the process of developing middleware to link serious games and ITSs. Since games can be used across multiple scenarios and training domains, providing an integrated game-based tutor will increase reuse and reduce authoring load.

A second category of emerging technologies is data-mining tools to develop an ITS expert model based on the analysis of text-based sources (e.g., how to manuals or web content). These tools reduce time and skill, and thereby the cost to develop domain models, an essential part of the ITS domain, without human knowledge of the domain. The accuracy of these current data-mining tools is a limiting factor with respect to the amount of authoring saved.

Chapter 27 (Domeshek, Jensen, and Ramachandran) discusses the concept of *bootstrapping* to support automated authoring. Bootstrapping includes “incremental rule condition generalization and student action templates created by demonstration and generalization.” An example of bootstrapping includes SimStudent (MacLellan, Stampfer Wiese, Matsuda & Koedinger, 2015), which collect learner behaviors and trends to support development of automated learner analytics (e.g., misconception libraries and expert models).

Decreasing the Skill Requirements for Authoring

Today, ITSs are built by highly skilled, multidisciplinary teams, which may include computer scientists, instructional designers, human factors psychologists, learning specialists, and domain experts. In order to reduce the skills required by ITS authors, some of the knowledge, skills, and best practices of these interdisciplinary team members must be represented in the authoring process by artificial intelligence methods. Default decisions are represented in the authoring process to accommodate novices or more experienced author preferences. For example, a novice may author a problem-based course and the selection of problems may be random and driven by metadata representing each problem’s complexity. During instruction this can be used by the domain model to select problems of appropriate complexity without the author’s specification of problem order.

Another authoring tool design goal is to create artificially intelligent job aids (e.g., TurboTax) to guide the author through the process and thereby reduce their cognitive load during authoring. For example, authors who move from model-tracing to dialogue-based tutors might have a job aid to support their transition. Authoring tool user interfaces must be able to recognize the level of experience in using the tool and how long since they last used it. Decreasing levels of scaffolding by the job aid should be experienced by the author as they become more knowledgeable.

Regardless of the approach, usability is a key to supporting efficient authoring. Understanding the capabilities and limitations of authors is vital. In Chapter 22, Aleven, Sewall, Popescu, van Velsen, and Demi advocate a *use-driven development* process consistent with human-computer interaction and user-centered design principles. In this approach, user experiences drive development priorities. In Chapter 23, Sinatra, Holden, Ososky, and Berkey discuss usability considerations and the effect of user roles. Chapter 24 (Gilbert and Blessing) examines user experience to describe the design of authoring tools including the need for multiple representations of domain knowledge to align with the mental models of the author.

Popular approaches to reducing required knowledge for authoring are reviewed in Chapter 25 (Lane, Core, and Goldberg). These include such as programming by demonstration, visualization tools, and what you see is what you get (WYSIWYG) authoring. Chapter 27 (Domeshek, Jensen, and Ramachandran) discusses user-friendly tools that allow subject matter experts or instructional designers to create complex knowledge components.

Supporting the Organization of Domain Knowledge

While it may not be feasible to have a totally generalized set of authoring tools for all disciplines, it may be possible to tailor authoring tool interfaces to meet the needs of specific user disciplines (e.g., instructional designers, course managers, researchers, and domain experts) and authoring tasks (e.g., domain knowledge organization, development of directed graphs for course, and assessments). Tools to aid the user in organizing their knowledge for quick recall and application can result in large authoring time savings. Authoring tools to support curation, which includes the search, retrieval, organization, and storage of domain knowledge, are critical to efficient development of ITSs. The ability to add metadata tags to knowledge components will aid in their organization and retrieval.

A significant element of domain knowledge is formed by defining objectives, measures, standards, and assessments for each concept to be learned. Chapter 26 (Goldberg, Hoffman, and Tarr) examines processes in GIFT to author adaptation through a data-driven approach which requires significant domain knowledge. As tutors expand into new domains (e.g., psychomotor and social domains), the challenge will be to organize domain knowledge for efficient authoring. Chapter 28 (Sotilare, Ososky, and Boyce) provides insight into the development of measures and challenges to authoring in the psychomotor domain (e.g., sports and marksmanship).

Enabling Rapid Evaluation of Prototypes

Our fourth goal is to enable rapid prototyping of adaptive tutoring systems and allow for rapid design/evaluation cycles of prototype capabilities. Decreasing the time required to evaluate prototypes will result in a more efficient model-test-model cycle and support more efficient authoring of new system capabilities (Murray, 1999; Murray, 2003; Sotilare, 2015). To this end, we recommend development of a standard testbed methodology as designed in GIFT (Sotilare, Goldberg, Brawner & Holden, 2012). The designers of GIFT have adapted their testbed methodology from Hanks, Pollack, and Cohen (1993). Elements, models, and methods within the learner module (e.g., transient states, cumulative states, and enduring traits), pedagogical module (e.g., instructional strategies), domain module (e.g., instructional tactics), and user interface (e.g., source of feedback) may be used to compare and contrast effect with alternatives.

Challenges and Best Practices

A model of *domain knowledge complexity* and its significant dimensions are needed to compare and contrast authoring tools and their performance. It is currently difficult to compare authoring systems of different genres (e.g., dialogue-based tutors vs. cognitive tutors) based on differences and overlapping functions within these ITS genres. It is also difficult to compare 1 hour of adaptive instruction when the density of adaptive strategies and tactics needed varies from domain to domain. Finally, it is essential to expand ITS domains beyond problem-centric tutors to more situated tutoring domains (e.g., scenario-based instruction). Authoring in various task domains (cognitive, affective, psychomotor, and social) also presents challenges in comparing the efficiency of authoring. Until a community-based standard

definition of domain knowledge complexity is agreed upon, we should restrict our comparisons to authoring systems within the same genre and task domain.

In response to this need, we put forward a recommended best practice for authoring comparison to identify *domain knowledge density*. We see domain knowledge density as a function of the number learning concepts, their associated measures and assessments, and the number of problems and their adaptations (e.g., problem steps) or in the case of situated tutors, scenario variables. Scenario variables include components within the scenario that can change in response to learner needs (e.g., boredom may require an increase in challenge level). Scenario density contributes to domain knowledge complexity and all density factors should be normalized to a one hour scale. Other suggested best practices are called out in subsequent chapters in this section which will allow us to compare authoring capabilities.

References

- Aleven, V., McLaren, B., Sewall, J. & Koedinger, K. (2006). "The Cognitive Tutor Authoring Tools (CTAT): Preliminary Evaluation of Efficiency Gains" In Proceedings of the 8th International Conference on Intelligent Tutoring Systems, 2006, 61-70.
- Hanks, S., Pollack, M.E. and Cohen, P.R. (1993). Benchmarks, Test Beds, Controlled Experimentation, and the Design of Agent Architectures. AI Magazine, Volume 14 Number 4.
- Lesgold, A.M., Lajoie, S., Bunzo, M. & Eggan, G. (1988). Sherlock: A coached practice environment for an electronics trouble shooting job. LRDC Report. Pittsburgh, PA: University of Pittsburgh, Learning Research and Development Center.
- MacLellan, C., Stampfer Wiese, E., Matsuda, N., and Koedinger, K. (2015). SimStudent: Authoring Expert Models by Tutoring. In R. Sotilare (Ed.) 2nd Annual GIFT Users Symposium (GIFTSym2), Pittsburgh, Pennsylvania, 12-13 June 2014. Army Research Laboratory, Orlando, Florida. ISBN: 978-0-9893923-4-1.
- Murray, T. (1999). Authoring intelligent tutoring systems: An analysis of the state of the art. International Journal of Artificial Intelligence in Education, 10(1):98–129.
- Murray, T. (2003). An Overview of Intelligent Tutoring System Authoring Tools: Updated analysis of the state of the art. Authoring tools for advanced technology learning environments. 2003, 491-545.
- Sotilare, R. and Gilbert, S. (2011). Considerations for tutoring, cognitive modeling, authoring and interaction design in serious games. Authoring Simulation and Game-based Intelligent Tutoring workshop at the Artificial Intelligence in Education Conference (AIED) 2011, Auckland, New Zealand, June 2011.
- Sotilare, R., Brawner, K., Goldberg, B. & Holden, H. (2012). The Generalized Intelligent Framework for Tutoring (GIFT). US Army Research Laboratory.
- Sotilare, R., Goldberg, B., Brawner, K. & Holden, H. (2012). A modular framework to support the authoring and assessment of adaptive computer-based tutoring systems (CBTS). In Proceedings of the Interservice/Industry Training Simulation & Education Conference, Orlando, Florida, December 2012.
- Sotilare, R., Holden, H., Goldberg, B. & Brawner, K. (2013). The Generalized Intelligent Framework for Tutoring (GIFT). In Best, C., Galanis, G., Kerry, J. and Sotilare, R. (Eds.) Fundamental Issues in Defence Simulation & Training. Ashgate Publishing.
- Sotilare, R. (2015). Examining Opportunities to Reduce the Time and Skill for Authoring Adaptive Intelligent Tutoring Systems. In R. Sotilare (Ed.) 2nd Annual GIFT Users Symposium (GIFTSym2), Pittsburgh, Pennsylvania, 12-13 June 2014. Army Research Laboratory, Orlando, Florida. ISBN: 978-0-9893923-4-1.
- VanLehn, K., Lynch, C., Schulze, K., Shapiro, J. A., Shelby, R., Taylor, L., et al., (2005). The Andes physics tutoring system: Lessons learned. International Journal of Artificial Intelligence and Education, 15(3), 147–204.
- VanLehn, K. (2011): The Relative Effectiveness of Human Tutoring, Intelligent Tutoring Systems, and Other Tutoring Systems, Educational Psychologist, 46:4, 197-221.

Chapter 22 Reflecting on Twelve Years of ITS Authoring Tools Research with CTAT

Vincent Aleven, Jonathan Sewall, Octav Popescu, Martin van Velsen, Sandra Demi, and Brett Leber
Human-Computer Interaction Institute, Carnegie Mellon University

Introduction

In this chapter, we reflect on our 12+ years of experience developing and using the Cognitive Tutor Authoring Tools (CTAT), by now a mature and widely used suite of ITS authoring tools. A key reason to create ITS authoring tools is to make ITS development easier, easier to learn, and more cost-effective, so that, ultimately, more ITSs can help more students learn. CTAT is no exception; it was created with these goals in mind. It has gone far in meeting these goals (for a recent update, see Aleven et al., under review), even if there is also substantial room for next steps, greater generalization, and a wider use base. Our reflections center around generalized architectures for tutoring systems –architectures that support relatively easy plug-and-play compatibility of ITS components or whole ITSs.

We identify eight themes that emerge from our experience with CTAT. We expect our reflections on these themes will have relevance to a substantial range of ITS authoring tools and generalized architectures, not just CTAT and the Generalized Intelligent Framework for Tutoring (GIFT) (Sottilare, Brawner, Goldberg, and Holden, 2012). These themes touch on issues such as use-driven development of authoring tools to make sure they address users' goals and needs, the importance of describing ITSs in terms of their tutoring behaviors, advantages of supporting both programmer and non-programmer options within a single ITS authoring tool suite, the versatility of solution space graphs within the process of authoring an ITS, three aspects of interoperability that an ITS authoring tools or a generalized ITS architecture should support, and finally, a discussion of how different classes of likely authors of ITSs in the near future might have different goals and needs, and what this implies for tool development. Along the way, we reflect on the degree to which CTAT could be viewed as a generalized architecture for tutoring and how it might be generalized further. We hope our thoughts can inform useful discussion within the field regarding ITS authoring tools and generalized ITS architectures.

A reader wanting get a quick summary might read the section “Overview of CTAT” and then the brief summaries marked “Recommendations for GIFT.” at the end of each section. These suggestions are meant to be relevant not just to GIFT, but to a wide range of ITS authoring tools and ITS architectures.

Overview of CTAT

CTAT is a suite of ITS authoring tools and, at the same time, a factored architecture for developing and delivering tutors. Tutors built with CTAT provide various forms of step-level guidance for complex problem solving activities as well as individualized task selection based on a Bayesian student model. CTAT supports multiple ways of authoring tutors, with multiple technology options for the tutor front-end and the same for the tutor back-end. CTAT supports deployment of tutors in a wide range of configurations and delivery environments. To support this range of authoring and delivery options, it has aspects of a generalized tutoring architecture, which we highlight below.

CTAT is a key component of a more encompassing infrastructure for ITS research and development, together with two other main components, the TutorShop and DataShop (Koedinger et al., 2010). In this infrastructure, CTAT provides tools for authoring tutor behavior as well as run-time support for tutors.

The TutorShop is a web-based learning management system geared specifically toward tutors. Besides offering learning management options (e.g., reports presenting tutor data to teachers), it supports a number of ways of deploying tutors on the Internet. DataShop is a large online repository for educational technology data sets plus a broad suite of analysis tools, designed for use by researchers and geared towards data-driven refinement of knowledge component models underlying tutors (Aleven & Koedinger, 2013).

CTAT supports two tutor technologies: Using CTAT, an author can create an example-tracing tutor using non-programmer methods (Aleven, McLaren, Sewall, & Koedinger, 2009) or can build a rule-based Cognitive Tutor either through AI programming (Aleven 2010) or using a non-programmer module called SimStudent (Matsuda, Cohen, & Koedinger, 2005, 2015). In a nutshell, an author starts by identifying an appropriate task domain and appropriate problem types for tutoring, carries out cognitive task analysis to understand the concepts and skills needed for competence in this task domain as well as how students learn them, designs and builds a problem-solving interface for the targeted problem type, and authors a domain model for the given tutor, either in the form of generalized examples (for an example-tracing tutors) or a rule-based cognitive model (for a Cognitive Tutor). An author can build a tutor interface using an off-the-shelf tutor interface builder (for Flash, Java, or HTML5) combined with tutor-enabled components that come with CTAT. Once a tutor interface is ready, an author creates and edits the domain knowledge that the tutor will help students learn, using a variety of tools, depending on the tutor type. Obtaining the desired tutor behavior across a range of tutor problems and solution strategies is usually an iterative process with multiple edit-test-debug cycles. An easy way to deploy CTAT tutors is to upload them to the TutorShop. This makes them available via the Internet, where they can be used in conjunction with the learning management facilities of the TutorShop. Other delivery options are available as well. Among CTAT tutors, example-tracing tutors are by far the more frequently authored tutor type.

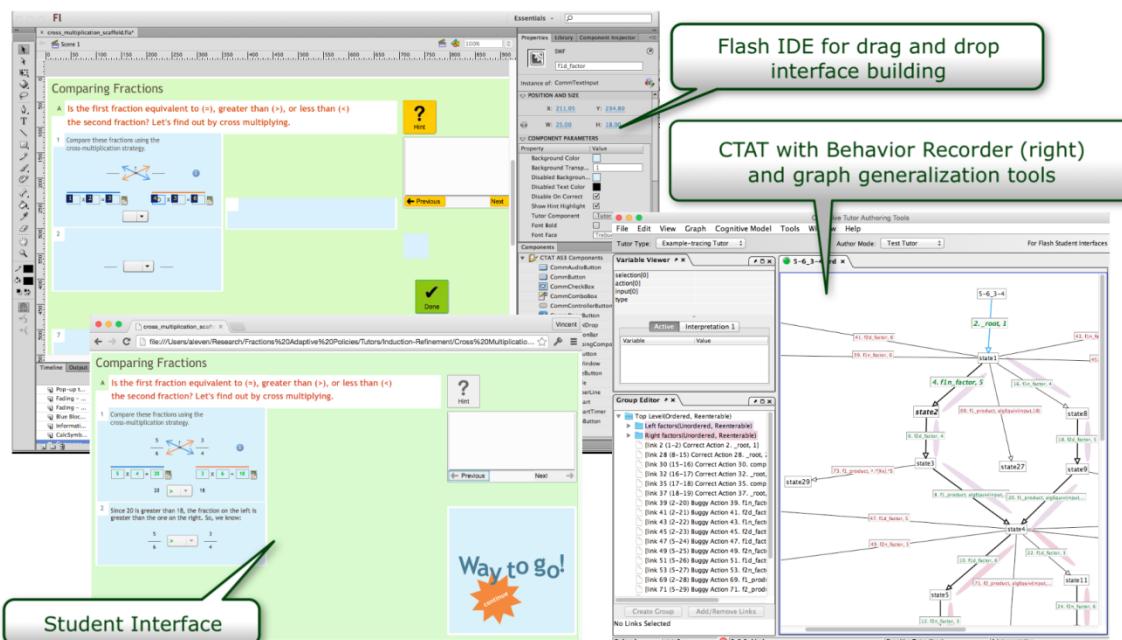


Figure 1: Authoring an example-tracing tutor with CTAT

When authoring an example-tracing tutor, an author edits the tutor's domain knowledge using a tool called the Behavior Recorder, shown in Figure 1 (Aleven et al., 2009; Koedinger, Aleven, Heffernan, McLaren, & Hockenberry, 2004; Koedinger, Aleven, & Heffernan, 2003). This knowledge takes the form of generalized examples captured as behavior graphs, with multiple strategies and common errors recorded as paths in the graph. The Behavior Recorder provides many options for creating, editing and generalizing a behavior graph, so it supports the desired tutor behavior. It also lets an author attach hints, error messages, and knowledge component labels. In addition it supports a variety of useful tutor-general functions (i.e., functions shared between example-tracing tutors and Cognitive Tutors), such as cognitive task analysis, solution space navigation, and semi-automated regression testing. These domain-general functions are discussed further below.

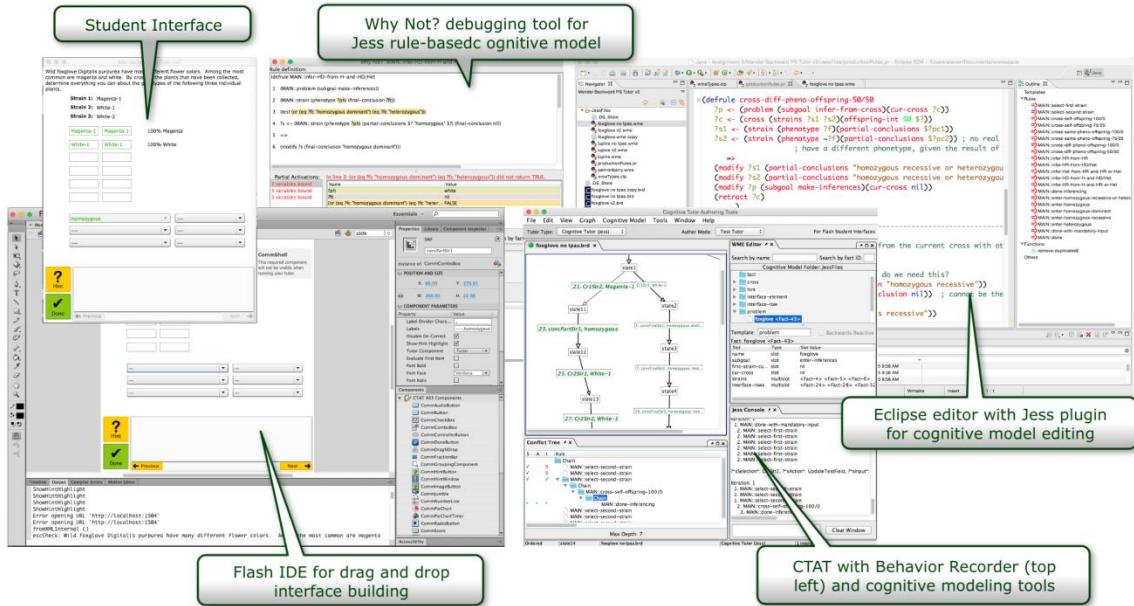


Figure 2: Authoring a rule-based Cognitive Tutor with CTAT

When authoring a rule-based model for a Cognitive Tutor, the second type of tutor that CTAT supports, an author uses tools for editing, testing, and debugging a cognitive model (Aleven 2010), as illustrated in Figure 2. These models are written in Jess, a standard production rule language (Friedman-Hill 2003). The tools used include an external editor (Eclipse with a plug-in for Jess editing) as well as the following CTAT tools: the Behavior Recorder for cognitive task analysis, solution space navigation, and testing, a working memory editor for inspecting/editing the contents of working memory, two diagnostic tools for debugging cognitive models, the conflict tree and why not window, and a Jess Console that provides a low-level command-line interface to the Jess interpreter. Most of these tools are specific to CTAT and are not available in standard production role developments. A controlled evaluation study shows these tools can substantially reduce the number of edit-test-debug cycles needed for cognitive model development (Aleven, McLaren, Sewall, & Koedinger, 2006). SimStudent, a machine learning module integrated with CTAT, supports a second, non-programmer, way of authoring a rule-based cognitive model for use in a Cognitive Tutor (MacLellan, Koedinger, & Matsuda, 2014; Matsuda et al., 2005, 2015). It supports programming-by-tutoring, in which it automatically induces rules from author-provided examples (behavior graphs) and author feedback. In this chapter, however, we focus primarily on example-tracing tutors.

A key difference between example-tracing tutors and model-tracing tutors is that example-tracing tutors are practical only for problem types that have no more than a moderately-branching solution space¹, whereas rule-based cognitive tutors can handle problems even with very large solution spaces (e.g., Waalkens, Aleven, & Taatgen, 2013). In practice, we have found that this constraint is met often, although not always (Aleven et al., under review). For example, model tracing may be more appropriate for computer programming and equation solving. There can be other reasons as well to prefer a rule-based tutor to an example-tracing tutor. With a rule-based tutor, it can be easier to create small variations of the same tutoring behavior within a problem, as might be useful in a research study. Also, sometimes the development team may include one or more people who are very facile with production rule writing.

It is important to point out, however, that in task domains where both approaches are applicable (i.e., no more than a moderately-branching solution space), example-tracing tutors and Cognitive Tutors can support the exact same tutoring behavior. If that seems a bold claim, consider that when the tutor interface for a certain problem type is kept constant, the tutor author has no choice but to author a domain model that captures all reasonable student strategies within the given interface, whether it be rule-based or example-based domain model. Otherwise, the tutor may flag as incorrect certain correct student behavior, namely, correct behavior not captured in the domain model, clearly an undesirable situation. Given a domain model that captures the same solution paths, the same essential tutoring behaviors are supported within the system's "inner loop" (VanLehn 2006), namely, immediate feedback, on-demand next step hints, error-specific feedback messages. In the outer loop, the system supports individualized task selection through Bayesian Knowledge Tracing and Cognitive Mastery (Corbett, McLaughlin, & Scarpinatto, 2000; Corbett & Anderson, 1995).

CTAT strengths are that it is a mature set of ITS authoring tools that support both non-programmer and programmer options to tutor authoring. The non-programmer approach is easy to learn and has turned out to be useful in a wide range of domains. It may be fair to say that a wider range of tutors has been built with CTAT than with any other ITS authoring tool. It appears to make tutor authoring 4-8 times as cost-effective (Aleven et al., 2009). CTAT's programmer approach can be used to build tutors for task domains in which CTAT's non-programmer approach is infeasible. CTAT-built tutors support complex problem solving with the full range of step-by-step guidance and problem selection options identified by VanLehn (2006) in his thoughtful cataloging of tutor behaviors². It builds on and generalizes from the experience of developing Cognitive Tutors. CTAT has been shown to be quite general, with tutors built for many domains covering a range of pedagogical approaches, including guided invention (Chase, Marks, Bennett, & Aleven, under review; Roll, Aleven, & Koedinger, 2010), collaborative learning (Olsen et al., 2014), simulation-based learning (Aleven, Sewall, McLaren, & Koedinger, 2006; Borek, McLaren, Karabinos, & Yaron, 2009), learning from erroneous examples (McLaren et al., 2012), and game-based learning (Forlizzi et al., 2014). Some of these systems required custom modifications of the tool, which does not, however, undercut the usefulness of having a tool. Many of these tutors been used in classrooms and other educational settings, as evidence of their robustness. Of these tutors, Mathtutor (Aleven, McLaren, & Sewall, 2009) and the Genetics Tutor (Corbett, Kauffman, McLaren, Wagner, & Jones, 2010) have seen substantial use over the years. In many studies, CTAT tutors were shown to help students learn including the Fractions Tutor (Rau, Aleven, & Rummel, 2013, 2015; Rau, Aleven, Rummel, & Pardos, 2014) and Lynnette (Long & Aleven, 2013, 2014; Waalkens et al., 2013), each of which been used in an elaborate sequence of research studies. CTAT has been used primarily by researchers, including researchers from outside of our own institution. A Google users group sometimes helps each other on the forum – members of the CTAT staff also answer queries.

¹ Sometimes, example-tracing tutors can handle a large solution space by collapsing multiple paths into a single one, using CTAT's formula language to express how steps depend on other steps.

² There is one exception: CTAT does not support end-of-problem reflective solution review.

CTAT's limitations are that it has a built-in pedagogical model for step-level problem-solving support, with little to no support for authoring custom tutoring strategies. Even so, a number of tutors have been developed with CTAT that support pedagogical approaches other than step-based problem solving, as mentioned above. Further, CTAT does not support natural language interactions. So far, CTAT have been primarily researchers; CTAT has not been used by teachers to author tutors for their students. Although many tutors have been built, only a small number of CTAT tutors are in regular, widespread use in schools or other educational institutions. As a final limitation, there is room for improvement, for example, if the authoring of very simple interactive activities were simpler and CTAT offered more options for interoperability.

Themes Regarding ITS Authoring Tools

In the remainder of this chapter, we highlight a number of key themes that emerge from the work on CTAT that might be applicable to other ITS authoring tool projects, including GIFT.

Use-driven Development of ITS Authoring Tools

A key goal for projects that create ITS authoring tools is to create useful, usable, and efficient tools that authors can use to create effective, efficient, and enjoyable learning experiences for students. To achieve this goal within the CTAT project, we have consistently followed a use-driven design approach, in line with many approaches to Human-Computer Interaction and User-Centered Design. Given that it is difficult to know up front exactly what tool functionality will be most useful and how users will use the tools, it is important to learn from users and to let the needs and experiences of actual tool users guide tool development priorities. To this end, we have promoted use of CTAT from day one and worked hard to hear about and learn from the experiences and needs of actual tool users. We then applied what we learned as we iteratively improved and extended the tools. This approach has been exceedingly helpful, while also being humbling at times.

The use-driven design approach requires having a base of motivated users. We have invested in building a community of users; in part this can be achieved by making sure the tool is genuinely useful for a wide range of users. In addition, we have worked hard to help CTAT users and potential CTAT users in all stages of their projects, through workshops, summer schools, a website with tutor examples, documentation and tutorials (<http://ctat.pact.cs.cmu.edu>), and an online user group (<https://groups.google.com/group/ctat-users>), and also by having an organization that can provide consulting, can host tutors on the Internet (e.g., in the TutorShop), and sometimes even can make custom changes to the tools. We have oftentimes “eaten our own dog food,” meaning that as tool developers we have used CTAT in tutor development projects such as Mathtutor, the Fractions Tutor, AdaptErrEx, the Genetics Tutor, and a host of others (Aleven et al., 2009; Corbett et al., 2010; McLaren et al., 2012; Rau et al., 2013; Rau et al., 2014).

We have found it to be important to take active steps to gather information from users, such as wish lists and feedback regarding their experiences with the tools. To do so, we have frequently polled users, closely consulted with users, and solicited feature requests when planning new CTAT releases. We frequently asked users about their likes and dislikes in questionnaires at the end of courses or summer schools and summer schools on ITSs that we taught. At times, we have also solicited feedback from experienced users. Second, we have often consulted with external researchers who used the tools in their research projects and learned from this experience. Oftentimes, these researchers got started with the tools when they visited our summer school. Finally, when planning releases, we often solicited requests for new features from researchers we knew were using the tools.

We made sure that the information we gained from users actually influenced our tool development agenda, both when planning new releases and when implementing new features out-of-cycle, in-between CTAT releases. For example, user requests for new tool features or tool modifications were the main way in which we prioritized new development. This kind of principled prioritization is very important, because we always had many more ideas for possible tool extensions than we had capacity to implement. Such is the nature of ITS authoring tools. Another useful criterion for feature prioritization was affordability: How often will authors use the new feature in the process of creating a tutor and how much more efficient will it make their work?

If there is a downside to use-driven tool development, it may be that it could inadvertently lead to gearing the tool towards the needs of certain groups of users at the expense of others, for example, groups that are more easily accessible or are more vocal. CTAT and the TutorShop have been honed as they were used to support ITS research in schools and other educational settings. Thus, they are somewhat geared towards educational researchers as a target group. Possibly, working with other potential users, such as teachers, instructors, and professional e-learning developers, may turn up additional requirements for the tool that would make it more useful for those groups. In spite of this caveat, the use-driven tool development approach has served CTAT well.

Recommendations for GIFT: Practice use-driven tool development by promoting tool use, learning from user experiences, and letting user experiences be a key factor when setting development priorities. Since different users have different needs and these needs are not always what the tool developers assume they are, this approach may help substantially in creating authoring functionality that is well aligned with users' needs and supports cost-effective tutor development.

Discuss ITSs and authoring tools in terms of their tutoring behavior

In discussions about ITS authoring tools, the question often comes up, *what kind of tutor* can authors build with this particular toolkit? Oftentimes, the answer is couched in terms of the underlying technology, for example, constraint-based tutors v. example-tracing tutors v. model-tracing tutors. As a contrasting perspective, following VanLehn (2006) we have often found it useful to focus on the *behavior of tutoring systems*. VanLehn cogently argues that, although there is a bewildering variety in ITS architectures, there is much greater regularity in the *behavior* of systems. In this behavior, we can discern meaningful dimensions for comparison. For example, many ITSs can fruitfully be viewed as providing step-level guidance as students work through challenging problem scenarios; step-level guidance can be broken down further, as illustrated in VanLehn's cataloguing of inner loop behaviors. This viewpoint extends to authoring tools: It is fruitful to discuss and compare ITS authoring tools in the first place in terms of the tutoring behaviors that can be created with these tools. VanLehn provides a foundational taxonomy of tutoring behaviors that is helpful in this regard.

This is not to say that ITS architecture is unimportant and should not be discussed. Quite the contrary, architecture is important, in particular in discussions about component re-use and interoperability. Ultimately, however, architecture is a means to an end; it is about how to realize particular tutoring behaviors in software that support effective student experiences and/or learning outcomes. Discussions about technology or architecture can be more focused when it is clear what tutoring behaviors are supported and whether the systems or architectures being compared support the same set of behaviors or different sets of behaviors. Sometimes, the same behaviors can be realized with different technologies. For example, among CTAT-using projects we know of two instances where tutors originally developed as rule-based Cognitive Tutors were later re-implemented, without loss of tutor behavior, as example-tracing tutors: the Mathtutor and Genetics Tutor projects, discussed above (Aleven et al., under review). At other times, more complex behaviors require a more complex architecture (e.g., Waalkens et al., 2013).

Sometimes, connections between architecture and behavior are not dictated by the architecture. For example, Cognitive Tutors typically provide instant feedback on problem-solving steps (Anderson, Corbett, Koedinger, & Pelletier, 1995) and constraint-based tutors typically provide feedback on demand (Mitrovic et al., 2006). It appears, however, that this difference is not dictated by their main architectural difference, the use of a rule-based model versus a constraint-based model. It is a pedagogical choice by their developers.

In comparisons of tutor types (e.g., Kodaganallur, Weitz, & Rosenthal, 2005; Mitrovic, Koedinger, & Martin, 2003), for example regarding authoring efficiency, it is important to take into account differences in tutor behaviors. Suppose it was found in a well-executed empirical study that building tutors with tool A is more efficient than with tool B. What do we conclude? It depends on the sets of tutoring behaviors that the respective tools support. Maybe tool A supports only simple tutors, whereas B supports many sophisticated tutoring behaviors. Conversely, perhaps tool A is more efficient *and* supports a wider range of tutoring behaviors than B.

While VanLehn's (2006) taxonomy of tutoring behaviors is very useful, it may be time to update it with a fine-grained set of categories. We see some further distinctions both for the inner loop (i.e., within-problem guidance offered by the tutoring system) and the outer loop (i.e., between-problem pedagogical decisions). For example, inner loops can differ vastly in their complexity (e.g., in whether they can recognize alternative strategies for solving the same problem or even in whether they support multi-step problems or only single-step problems). These distinctions are not made in VanLehn's taxonomy, but are likely to be useful in comparing tutors and tools and in thinking about ITS architectures (e.g., (Waalkens et al., 2013)). Also, tutors both in their inner loop and outer loop also differ substantially in the range of student characteristics they adapt to (student knowledge growth, affect, and so forth). VanLehn's taxonomy is agnostic to these differences as well. A slightly more fine-grained taxonomy may be helpful especially as the number of tools for building online instruction, including "learning by doing" or "activities," is likely to grow enormously in the years to come.

Recommendations for GIFT: Clarify the tutoring behaviors supported in GIFT tutors in terms of VanLehn's (2006) taxonomy; this clarification will make it easy to understand for ITS researcher and developers to understand what kinds of tutors can be built with GIFT and where the strengths and limitations are of the GIFT authoring environment. Indicate where VanLehn's taxonomy needs to be extended or needs to be more fine-grained, based on the experience developing GIFT and developing tutors with GIFT. That is, clarify what features authorable within GIFT tutors are missing from VanLehn's (2006) taxonomy or are more naturally described at a slightly finer grain size than is currently done in this taxonomy. A more elaborate, fine-grained taxonomy will benefit the GIFT and ITS communities as it may help support meaningful discussion characterizing tutors and authoring tools in terms of behavior.

Support Both Non-Programmer and Programmer Options for Authoring

The experience with CTAT provides a perspective on the value of integrating, within a single ITS authoring tool suite, multiple authoring methods and ITS technologies – and by extension, of the value of certain ways of making tools interoperable. As mentioned, CTAT supports both programmer and non-programmer options to tutor authoring. Example-tracing tutors can be authored without actual coding, through drag-and-drop interface building and programming-by-demonstration. By contrast, creating rule-based cognitive models requires AI programming in the Jess production rule language³. As an additional

³ As mentioned, with SimStudent, rule-based models can be created without programming, an approach to tutor authoring that may well have a bright future.

way in which programming can be used when creating a CTAT tutor, an author can program custom behaviors either in the tutor interface, the tutor's domain model, or in the tools themselves. For example, when using a Flash interface, an author could program custom behaviors in Actionscript without changing CTAT. When building an example-tracing tutor, the author can write functions in Java or Javascript to augment CTAT's formula language, without changing CTAT proper. Likewise, when creating a rule-based model, an author can write functions in the Jess language, again without altering CTAT proper.

Providing multiple tutor paradigms within a single tool suite has substantial benefits. It makes it easier for an author, authoring team, or organization to select the option (i.e., tool and tutoring paradigm) that best fits the requirements for a given project, without having to switch to a different tool suite. The simpler and more efficient paradigm (e.g., example-tracing tutors) could be used for simpler tutor problems, the more complex paradigm (e.g., Cognitive Tutors) only when the situation calls for it. Alternatively, the simpler and more efficient paradigm could be used to create quick prototypes, say with less than complete within-problem guidance, and the more complex technology for the full-blown tutor. Even within a given tutor development project, a different technology could be used for different tutor units or tutor problems. Also, in the course of a project, it may sometimes be useful to switch from one technology to the other, for example, as the project team's understanding of the task domain for which tutoring is provided evolves. (Examples are discussed below.) What stays constant across these options, ideally, is not just the tool suite but also the delivery options. To the extent that the different tutoring options share tools within the tool suite, authors need to learn fewer tools (e.g., in CTAT, the interface builder and some of the Behavior Recorder functionality are shared across tutor types). Similarly, to the extent that the tools have been designed to support a similar tutor development process across tutor technologies, there is less to learn for an author. Finally, it may be substantially easier, given a single tool suite, to create tutors with a consistent look and feel and interaction style, even when – within the tool suite – a different technology is used for the tutor front end or back end. The different tutor types may be consistent in other ways as well, such as the logging format, the information they communicate to the LMS, and the underlying implementation language (useful when authors want to modify the tool itself).

These potential advantages have been illustrated in CTAT to a degree. In practice, CTAT's non-programmer option (example-tracing tutors) has been far more popular than we anticipated. By now many more example-tracing tutors have been built with CTAT than rule-based tutors (perhaps 50 times as many). The relative popularity of example-tracing tutors may reflect the fact they are easier to learn and more cost-effective (4-8 times more so, compared to historic estimates of tutor development (Aleven et al., 2009). It may also reflect the fact that CTAT's primary audience so far has been educational researchers or researchers in the learning sciences, who often are not programmers. Finally, in many domains, there is a substantial range of pedagogically-desirable problems with moderately branching solution spaces – which are within the realm of example-tracing tutor development. The same phenomenon seems to be true as well in some non-CTAT ITSs such as ASSISTMents (Heffernan & Heffernan, 2014) and Wayang Outpost, (Arroyo et al., 2014), which have limited capacity to support multiple solution paths within any given problem.

Some projects with CTAT illustrate the utility of having multiple tutor options within a single tool suite. Two major ITS projects transitioned from rule-based cognitive tutor technology to example-tracing tutor technology, both supported by CTAT, namely, Mathtutor (Aleven et al., 2009) and the Genetics Tutor (Corbett et al., 2010). In both instances, the main reason for switching was that example-tracing tutors can easily be made accessible over the Internet. In both instances, a large number of tutor units could be re-implemented as example-tracing tutors while duplicating the tutor behavior, although a small number of tutor units could not (i.e., required a rule-based model), due to having a large solution space. These two conversion projects are the main reason for the claim above that there is a large class of pedagogically-desirable problems that does not have a large solution space. For example, the Mathtutor problems were created based on a principled pedagogy (Koedinger, 2002), not because they were amenable to

development as example-tracing tutors. In a third project, the converse happened: We originally built the Lynnette equation-solving tutor as an example-tracing tutor, but later re-implemented it as a rule-based Cognitive Tutor, so as to have greater flexibility in recognizing student strategies (Long & Aleven, 2014). These three projects illustrates that one advantage of offering multiple tool options within a given tool suite, is that it facilitates the transition from one tutor type to the other.

A key idea behind both CTAT and GIFT is to support multiple ITS technologies with associated authoring tools within an integrated development infrastructure. It is interesting to consider how much further that idea might be pushed within CTAT. It may well be very useful to integrate more tool/tutor options, both at the high end and the low end of complexity of tutoring behavior. At the low end, an author may want to create, for example, multiple choice questions, perhaps with a small amount of adaptivity depending on student answers. CTAT does not support this simple authoring task in a simple way. If a tool could be integrated with CTAT that makes the authoring of simple activities more cost-effective, this tool could serve as a gentle introduction and a gentle slope towards authoring more complex activities. Likewise, towards the high end of complexity, one could foresee integrating conversational agents (Adamson, Dyke, Jang, & Rosé, 2014; Kumar & Rose, 2011; Rus, D'Mello, Hu, & Graesser, 2013; VanLehn et al., 2007) and other types of ITSs, such as constraint-based tutors (Mitrovic & Ohlsson, 1999). These higher-end options may contribute to making more effective tutors with greater impact on student learning, within the same tool suite. Greater integration therefore is a worthy goal; it raises interesting interoperability questions, discussed below.

Recommendations for GIFT: Continue to work on creating a generalized architecture and on accommodating a wide spectrum of tool options. A key advantage of doing so is that GIFT have more options to match the tool with the particular educational objectives that they are authoring for, which may lead to greater cost-effectiveness or greater effectiveness (e.g., stronger learning outcomes by students), or both.

Recordable and Editable Behavior Graphs as a Central Representational Tool

In CTAT, behavior graphs are a central representational tool. Behavior graphs and the associated Behavior Recorder tool serve many useful functions within CTAT. In this section, we focus on the tutor-general authoring functions of behavior graphs, as opposed to their role of capturing domain knowledge in example-tracing tutors, which is specific to one tutor type. These tutor-general functions include aiding in cognitive task analysis, planning of the development of a tutor's domain model, recording of solution paths for use during testing, navigation within the solution space of a tutor problem (somewhat akin to book marking), and semi-automated testing. In CTAT, these functions have proven to be useful during the development process of both example-tracing tutors and Cognitive Tutors, evidence that they are general across tutor types. It seems likely that these Behavior Recorder functions could be equally useful in the development of other tutor types and in other ITS authoring tools.

A behavior graph captures problem-specific problem-solving behavior targeted in the instruction – that is, problem-solving behavior that the tutor will help students learn. Behavior graphs capture step-by-step solutions to problems, where appropriate with multiple paths within any given problem⁴. They may also capture common errors that students make, marked as such by the author. A given behavior graph is

⁴ Behavior graphs do not capture pedagogical knowledge, they capture ways in which tutor problems can be solved (i.e., problem-solving knowledge). A recent chapter by Pavlik, Brawner, Olney, and Mitrovic (2013) likens behavior graphs to “programmed instruction.” This analogy is inaccurate in that the graphs in programmed instruction capture pedagogical decisions but behavior graphs capture problem-solving knowledge (see also Aleven et al., under review).

specific to a given problem and a given tutor interface. CTAT's Behavior Recorder tool makes it easy to create and edit behavior graphs. To record a graph, all an author needs to do is demonstrate the behaviors in the interface. Behavior graphs are well-aligned with a variety of theoretical frameworks on ITSs, including the notion that ITSs capture complex, multi-step problems solving with multiple strategies within a given problem, Anderson et al.'s (1995) notions of making thinking visible and reducing cognitive load by supporting step-by-step reasoning with feedback, and with VanLehn's (2006) notion of step-based tutoring.

With respect to the tutor-general functions of behavior graphs, first, behavior graphs are a tool for cognitive task analysis in support of tutor design and curriculum design. A behavior graph can help a tutor author identify strategies, common errors, and knowledge components, perhaps aided by student data⁵. That is, after creating one or more behavior graphs, an author may consider what knowledge is involved in each step in the given graph and which steps may involve the same knowledge components. This in turns may help an author think about how to structure the tutor's domain knowledge model (e.g., a rule-based model in the case of a Cognitive Tutor) and possibly even in planning how to implement it, for example by considering which sequence of problems should drive development or which paths in a graph should be implemented first. Knowledge component analysis aided by behavior graphs may also help in making decisions about curriculum and structuring problem sets, for example, decisions about the order in which to introduce new knowledge components. A graph may even prompt re-thinking and re-designing of a tutor interface. For example, when certain steps in a graph turn out to involve multiple knowledge components, an author may decide to break down the step into smaller steps in the tutor interface. These functions are not specific to any given type of tutors, because the only assumption that is made is that mapping out the solution space of a problem is useful. Hand-drawn behavior graphs were used in the process of developing rule-based Cognitive Tutors prior to CTAT. Behavior graphs have also long been used as tools by cognitive scientists (e.g., (Newell & Simon, 1972)).

A second key function of behavior graphs and the Behavior Recorder is to aid testing, both the frequent within-problem testing that happens during development as well as regression testing when a domain model has been modified or re-structured in some way. Importantly, a behavior graph makes it easy for an author to navigate the solution space of a given problem during the development of the tutor's domain model. By clicking on a state in the behavior graph, an author can advance the tutor to the given state, which saves time during testing and debugging, compared to having to enter the steps manually in the interface. For example, in Cognitive Tutor development, when testing the rules for step 17 in a problem, there is no need, after each edit, to enter steps 1 through 16 by hand in the interface. The savings add up especially if one considers that often multiple edit-test-debug cycles are needed.

In addition, after more extensive edits to a tutor's domain model (e.g., a rule-based cognitive model), the graph for a given problem can be used as a semi-automated test case for regression testing, as it captures problem-solving steps that the tutor should recognize as correct. In CTAT, graphs can be used for this purpose by issuing the CTAT command "Test Cognitive Model on All Steps." CTAT reports on whether the tutor is capable of recognizing as correct, all solution paths captured in the behavior graph and recognizing as incorrect, the various errors captured (and marked as such) in the behavior graph. A useful idea that we never came around to implementing is to support batch regression testing in CTAT, by having the tool test a tutor's domain model on a large collection of behavior graphs all at once.

⁵ Although automated or semi-automated approaches to constructing or refining cognitive models from data are becoming more powerful and prevalent (e.g., Aleven & Koedinger, 2013), we nonetheless continue to recommend collecting some data early on during the process of tutor development, before log data can be collected for these (semi-)automated approaches. These data could include for example think-aloud protocols of novices and intermediates in the domain solving tutor problems on paper (Baker, Corbett, & Koedinger, 2007; Lovett 1998).

Finally, as perhaps an unexpected twist on tutor-general use of behavior graphs, example-tracing tutors, which use behavior graphs as their main representation of domain knowledge, could be used as early (throw-away) prototypes of other types of (more sophisticated) tutors, such as rule-based Cognitive Tutors built with CTAT. In fact, in the early days of CTAT, when we started to create the example-tracing technology, we originally conceived of example-tracing tutors in this manner.

In sum, behavior graphs serve many useful tutor-general functions in CTAT and could conceivably do so in other authoring tools as well, if/when the Behavior Recorder were integrated with these tools. We offer some thoughts on how this might be done below. More generally, behavior graphs can be viewed as unifying different tutoring technologies, not just example-tracing tutors and Cognitive Tutors but also, for example, constraint-based tutors (Mitrovic & Ohlsson, 1999) and ASSISTMents (Heffernan & Heffernan, 2014), since solving problems of moderate to somewhat high complexity is common to all.

Recommendations for GIFT: Integrate the Behavior Recorder so behavior graphs can aid in the development of GIFT tutors. This integration will make the tutor-general Behavior Recorder authoring functionality (support for cognitive task analysis, solution space navigation, testing, and early prototyping) available for GIFT authoring, potentially making it more efficient.

Support for Interoperability through the Tool/Tutor Communication Interface

A key requirement for creating a generalized architecture for ITS research and development, such as CTAT and GIFT, is interoperability and re-use of ITS components. By component interoperability we mean that it is relatively easy to use different instantiations of the same functional module within the overall architecture. Therefore, a key issue is: how can we define useful functional modules and interfaces between these modules that allow for general plug-and-play compatibility? Such compatibility has many advantages. For example, it would make it much easier to assemble an ITS out of components as building blocks and would facilitate research into how best to implement or combine typical ITS modules. In this section we focus on one particular form of interoperability that has been central in Cognitive Tutors and has been adopted in CTAT right from the start, adherence to Ritter and Koedinger's (1996) tool/tutor interface. It is not a new idea but one whose proven usefulness may not be widely known.

The main idea is to concentrate tool/interface functionality on the one hand and tutor functionality on the other in separate modules, and define a messaging protocol (or API) between these two modules (Ritter & Koedinger, 1996). By the tool/interface Ritter and Koedinger mean what in CTAT is called the "tutor interface" and what also has been called the "environment module" in older ITS literature, that is, the interface or environment in which the student solves problems with guidance from the tutor. The tool/interface is conceptualized not merely as an interface (i.e., a GUI in which to enter problem-solving steps), but more broadly as a tool that is useful in the given task domain, capable of calculations and inferences that are useful for practitioners, such as a spreadsheet or simulator. The pedagogical goal is often not that students learn to do themselves what the tool does for them, but rather that they learn to use the tool as tool or learn from applying the tool (e.g., concepts or processes in a simulator). The tutor component is the rest of the tutoring system, with its outer loop and inner loop functionality, in VanLehn's (2006) terms. The tutor's responsibility is to guide students with respect to their current problem-solving goals, using the available tools, in a way that helps learning. In practice, it is not always easy to draw a clear line between tool and tutor functionality; nonetheless, the separation of tool and tutor is very useful.

The main driver behind Ritter and Koedinger's (1996) separation was to provide tutoring within a number of existing problem-solving environments, while re-using the existing tutoring technology, the Cognitive Tutor, without modification. For example, they developed prototypes that provide tutoring within Excel

and Geometer's Sketchpad, hooked up a simulator and argument diagramming system (Koedinger, Suthers, & Forbus, 1999), and in a later project provided tutoring within Excel (Mathan & Koedinger, 2005). They were able to hook up the tutor backend without modification (though they needed to develop a new cognitive model for the given task domain) and with relatively minimal changes to the problem-solving environments. This same idea is also being pursued in other ITS authoring tool efforts, such as xPST (Blessing, Devasani, & Gilbert, 2011).

CTAT implements and rigorously adheres to its own version of the tool/tutor interface, documented at <http://ctat.pact.cs.cmu.edu/index.php?id=tool/tutor>. The tool/tutor interface has been highly useful over the lifespan of the project. First and foremost, CTAT has long offered multiple options for the tutor module and the tool module (i.e., the tutor back end and front end, respectively). CTAT supports example-tracing tutors and Cognitive Tutors as tutor back end ("tutor engines" in CTAT parlance) and Java, Flash, and HTML5/Javascript as interface technologies. Due to the tool/tutor interface, these options are interoperable: each tutor option can be combined with each tool option. Therefore, an author can select the interface option that best supports the intended student interactions, without being constrained in the choice of tutor engine. Conversely, an author can select the tutor engine option that best matches the domain, without being forced into a specific interface option. Also, having multiple interface options helps deliver tutors on multiple platforms and has made it possible, over the years, to support the use of a range of different off-the-shelf interface builders (DreamWeaver, Netbeans, IntelliJ, Eclipse, and Flash). As a second main advantage, CTAT tutors have been built for a number of external problem-solving environments, such as a thermodynamics simulator called CyclePad (Aleven et al., 2006), a chemistry simulator called the Vlab (Borek et al., 2009), and most recently, Google sheets. A third advantage is that adherence to the tool/tutor communication protocol has the useful side effect that it implements a logging capability, as it is the message traffic between tool and tutor that is being logged. Finally, and perhaps most importantly, the tool/tutor separation has helped us keep up with developments in web technologies. Over the years, we have moved from Java (still supported), to Actionscript 2, to a new look-and-feel in Actionscript 2, to Actionscript 3, and to HTML5/Javascript. Due to the rigorous tool/tutor separation in the CTAT architecture, these conversions (though extraordinarily labor-intensive) were at least somewhat manageable, as the tutor back-end was not affected.

CTAT uses Ritter and Koedinger's tool/tutor messaging protocol with a few small extensions. For example, we added untutored actions –student actions in the tutor interface that the tutor module needs to know about but does not need to respond to right away, in contrast to the more typical tutored student actions. For example, in the Fractions Tutor (Rau et al., 2013), built with CTAT, untutored actions are used in an interactive interface component for representing fractions as circles. A student may partition a circle into a number of equal pieces equal to the denominator. Oftentimes, this action can be untutored – no need for the tutor to provide feedback until the student submits the fraction. However, if the student asks for a hint before submitting the fraction, the hints can be more apropos if they can take into account into how many pieces the circle has been divided. We are also considering broadening the tool/tutor interface such that the state of the tool or a tool component can be communicated to the tutor. This facility may support tutoring in dynamic environments such as games and simulations.

A limitation of the tool/tutor approach to ITS component interoperability may be that it can be fruitful to distinguish more components within an ITS architecture than just a tool and a tutor component. For example, it may be helpful to break down the tutor component into various subcomponents, such as a learner model, pedagogical module, and so forth. Other possibilities may include supporting Ritter's notion of multiple tutor agents (Ritter, 1997), separating the inner loop and outer loop (as is done in CTAT, see below), or standardizing the API between the student model and other tutor components. A further limitation may be that the tool/tutor interface does not capture a number of other desirable aspects of interoperability, such as interoperability with tutor-general authoring tools such as the Behavior Recorder (see above) or with LMSs and platforms for e-learning or MOOCs (see below).

Recommendations for GIFT: Support a version of Ritter and Koedinger's (1996) tool/tutor interface within GIFT. Implementing the tool/tutor interface could bring the same advantages to GIFT that it has had for CTAT, for example, the mixing and matching of a tool/interface module and a tutor engine, assuming multiple options would be offered for each. Supporting the tool/tutor interface may also facilitate hooking up external components, such as simulations or problem-solving environments. Further, supporting the tool/tutor interface could make it possible to hook external tutor engines into GIFT, if they were made to adhere to the tool/tutor protocol. Generally, these options would enhance the utility of GIFT as a generalized ITS architecture and would be a significant step forward for ITS research and development.

Interoperability with External Tools and Services

In this section, we consider a different form of interoperability, namely, interoperability with external tools and services that are useful or necessary when ITSs are used in real educational settings. Specifically, we look at interoperability with ITS logging services (namely, DataShop, Koedinger et al., 2010) and interoperability with LMSs, MOOC platforms, and other courseware. These aspects of interoperability have to some degree been realized in CTAT. The integration with DataShop has proven its value many times.

First, a key form of interoperability in CTAT and many other ITSs is interoperability with tutor log services and related analysis tools for tutor log data. Tutor log data is an invaluable resource for research and development of tutors. For example, it can be used to gain detailed insight into a tutor's effectiveness, to analyze and refine a tutor's knowledge component model (e.g., Aleven & Koedinger, 2013; Koedinger, Stamper, McLaughlin, & Nixon, 2013), generate hints (Stamper, Eagle, Barnes, & Croy, 2013), or support re-design of a tutor unit that results in more effective or efficient student learning (Cen, Koedinger, & Junker, 2007; Koedinger et al., 2013). This form of data-driven refinement of ITS is well on its way to becoming standard practice in ITS research and development. All CTAT tutors log in DataShop format, without special measures by the author. DataShop provides a web-based data repository plus analysis tools (learning curves, error reports, model fitting to evaluate knowledge component models, etc.). It has also become an important source of data sets for secondary analysis, for example by researchers in the EDM community. The log data is essentially a dump of the tool-tutor message stream (see the previous section; see <https://pslcdatashop.web.cmu.edu/dtd>).

Second, it is important for ITSs to be interoperable with standard e-learning platforms, MOOC systems, and LMSs, for example, to support automated tutoring at scale and in online courses. An easy and general path towards such integration may be to make tutors adhere to existing e-learning interoperability standards, such as SCORM (Sharable Content Object Reference Model, <http://www.adlnet.gov/scorm/>) and LTI (Learning Tools Interoperability, <http://www.imsglobal.org/lti/>). First, we have extended CTAT so that tutors built in CTAT can be saved in SCORM 1.2 compliant form. The SCORM 1.2 format is sufficient for embedding CTAT tutors in for example Moodle, a widely used (open-source) e-learning platform (Rice 2011). Our students have implemented a range of prototypes of Moodle courses with embedded CTAT tutors. Second, CTAT tutors are now LTI 1.1-compliant. We have used this form of integration to field a simple CTAT tutor in an EdX MOOC "Data Analytics for Learning," by Ryan Baker, Carolyn Rose, and George Siemens (fall 2014). This pilot study demonstrates the feasibility of the technology integration (Aleven et al., 2015). In this form of integration, in contrast to the SCORM integration, CTAT tutors are served from the TutorShop, meaning its functionality (e.g., teacher reports and individualized problem selection based on a learner model). The downside is that we are not taking advantage of the MOOC's servers' ability to support very large numbers of learners, unless the TutorShop is installed on these servers.

Additional steps are needed for tutor use in existing MOOCs and e-learning platforms to be fully robust, scalable, and effective. For example, ITSs produce a wealth of analytics – how can we make more learning analytics available in the LMS and its dashboard? How can we embed adaptive problem selection in LMS? These key functions are not available in standard LMSs. Different approaches are needed in the LTI versus the SCORM integration. One approach would be for existing LMSs or MOOC platforms to better accommodate tutors, perhaps through plugins. A related idea is to keep the TutorShop in the loop, as it is in our initial CTAT/EdX integration, and to make it possible for the MOOC or external LMS to better take advantage of TutorShop functionality.

Recommendations for GIFT: (1) Support DataShop logging in GIFT, perhaps as one of multiple possible logging options. Having GIFT tutors log in this format will make it easy to make GIFT datasets available in DataShop to the EDM community and will make it possible to analyze GIFT data sets using the DataShop tools (e.g., the tools for KC model analysis and refinement). (2) Make it possible to deploy tutors in existing LMSs, perhaps through SCORM or LTI integration. This may well be an important step in order for GIFT to be adopted in higher education. In general, the more delivery options GIFT can support, the more widespread the tool may become.

Other Potentially Useful Forms of ITS Component Interoperability

We briefly mention two additional issues regarding interoperability that may be useful in ITS architectures, and consider their potential costs and benefits. The first interoperability issue is inner/outer loop separation. To recall, under VanLehn's (2006) definitions, in ITSs the inner loop is responsible for within-problem guidance, the outer loop is responsible for problem selection and other between-problem pedagogical decisions. Currently, the CTAT/TutorShop architecture separates the inner loop and the outer loop; all communication between them goes through the student model. A key question is whether and why this separation is useful.

Before we answer this question, let us first briefly elaborate on how the inner loop, outer loop, and student model are implemented in CTAT/TutorShop. The inner loop (also referred to as tutor engine or tutoring service, with example tracing and model tracing as options) provides within-problem guidance, with step-level correctness feedback, hints, and error messages. The inner loop is also responsible for updating the student model, in line with VanLehn's (2006) notion. The student model captures a student's mastery of targeted knowledge components (see e.g., Aleven & Koedinger, 2013). The inner loop computes these mastery estimates using the Bayesian knowledge-tracing algorithm (Corbett & Anderson, 1995). In addition, the inner loop generates a fair amount of student history information, essentially raw data about past performance. Both the student model and the student history are stored in the TutorShop database. The student history is used primarily for report generation in the TutorShop. The outer loop offers various task selection options, namely, adaptive problem selection or Cognitive Mastery (Corbett et al., 2000), fixed problem sequence, or random. The adaptive option makes use of the mastery estimates in the student model to select problems that require unmastered knowledge components. As mentioned, the inner loop and outer loop communicate solely through the student model. CTAT also offers some ways in which authors can use the student model information in the inner loop to individualize instruction, although this is a relatively new feature that has not been used in classrooms.

The main advantage of the inner/outer loop separation is that the different inner and outer loop options can be used in mix-and-match manner, as the student model (the communication between the loops) does not change. For the inner loop, an author can choose between example tracing and model tracing as options. For the outer loop, the TutorShop offers a choice (for any given problem set) between adaptive problem selection, fixed problem sequence, or random. The student model is the same, regardless of these choices. Also, somewhat philosophically, defining the student model as the information (and exactly the

information) that is shared between the inner and outer loop seems right to us – if nothing else, it offers useful clarity and modularity. We do not see any downside to the inner/outer loop separation based on our specific experiences with the CTAT/TutorShop architecture.

A limitation of the CTAT/TutorShop architecture is that while an author can select different outer loop options, there are no facilities for authoring new outer loop options. For example, an author may want to implement a problem selection method based on factors other than knowledge component mastery, for example, metacognition (Aleven, Roll, McLaren, & Koedinger, 2010), affect (D'Mello & Graesser, 2014), effort (Arroyo et al., 2014), or interest (Walkington 2013). As another example, an author may want to apply game design principles, such as varying the challenge level based on the need to decompress every once in a while. Also, an author may want her tutor to jump back, when appropriate, to pre-requisite units. And so forth. Currently, new outer loop options can be created only through code modification of the TutorShop, but there is no API documentation or easy access to relevant parts of the code to facilitate this. Outer loop authoring is likely to lead to the requirement that the author has more control over the content of the student model and the methods for updating (whether in the inner loop or outer loop). Thus, in a generalized tutoring architecture, it seems important to support authoring options for outer loop task selection, student model, and methods for updating it.

Recommendations for GIFT: Describe how the inner and outer loop are related; what information do they share and how is this information passed back and forth? Describe inner and outer loop options that are supported as-is by the tool suite; describe the process and tools for authoring outer loops, student model, and student model updates. Describing the inner loop and outer loop options will make it easier for potential authors to understand GIFT's capabilities. It may also help ITS researchers and developers understand to what degree the separation of inner loop and outer loop, with the student model as communication, generalizes.

Who is the Author?

In this final section, we offer some speculative thoughts as to the degree to which different categories of potential ITS authors might have different needs, calling for different tool features. Our thoughts on this issue are no doubt colored by our specific experience: Although over the years we have interacted with many different CTAT users, CTAT has been widely used primarily educational researchers.

We see four broad classes of users for ITS authoring tools. The first category of authors are researchers of various kinds, including students, faculty, and professional researchers in areas such as education, ITS/AIEd, other kinds of educational technology, machine learning and educational data mining, and so forth. A second category of authors are teachers and instructors, in primary, secondary, and post-secondary education. These users want to create tutors for use in their own courses, perhaps helped by tech-savvy research assistants. The third category are professional e-learning developers, of which by now a very wide variety exists, in industry, government, the military, higher education, and even in primary and secondary education. A fourth category of authors may be volunteers who contribute their time and acquired ITS authoring ingenuity to websites where ITSs are freely available (e.g., Assistments (Razzaq et al., 2009), Mathtutor (Aleven et al., 2009)) and perhaps other institutions using ITSs. As mentioned, CTAT has been used mainly by the first category of users. The ASSISTMents authoring tools have been used mainly by the second (teachers) and to a lesser degree by the fourth (volunteers), the Carnegie Learning tools by the third (professionals) (Blessing, Gilbert, Ourada, & Ritter, 2009). To simplify the discussion, it may make sense to distinguish between only two classes of users, occasional authors (categories 1, 2, and 4) versus professional authors (category 3). Let us consider the assumed goals and needs of these two groups of ITS authors.

Our professional author might have a master's degree from an educational technology program and may have some knowledge of the research literature around ITSs as well as the use of data analytics in education. She may have some programming experience though not a strong CS background. Professional authors may often be driven by the goal to make their company's e-learning product better. Although ITSs have a proven track record in improving student learning (Ma, Adesope, Nesbit, & Liu, 2014; Pane, Griffin, McCaffrey, & Karam, 2013; Steenbergen-Hu & Cooper, 2013, 2014; VanLehn 2011), research-based evidence may not (yet?) be a strong factor in the market place, nor is being able to say that a product has intelligent tutoring technology embedded in it. The main draw for professional authors may therefore be the support for personalization and adaptivity that ITSs offer. Some professional authors may be interested in trying to document learning gains attributable to the use of ITS technology in their products, at least if the tool makes that very easy to do.

We might surmise further that professional authors often cover whole courses or whole curricula. When they consider adding ITS technology to an existing course, they may have data up front that could help them in identifying where the course is not effective and where an ITS might help strengthen it. They likely have time to invest in learning the ins and outs of the tool. Although for all categories of users, it is important that the initial cut at the tutors they build are very effective, professional authors may have greater latitude than other authors to iterate on the tutors that they author with the tool, informed by analytics obtained from the tutors. For professional authors, the ITS authoring tool may be one of a range of tools that they use in their daily practice. They have high expectations with respect to robustness and functionality offered, but are not scared away by having to download and install software. More than other categories of authors, they have knowledge of instructional design and research-supported learning principles that they can bring to bear as they design tutors. Importantly, professional authors are typically bought in to a specific e-learning platform, whether proprietary or publicly available. Thus, the tutors they author must be compatible with and easily deployed within this platform. They also need to have a look and feel that is compatible with that of the company's platform.

Occasional authors are a very diverse group. They often have the goal of making a difference in the education of their own students, of students in the organization in which they are volunteering, or of students in general (e.g., by doing research on tutors). Occasional authors are driven by various specific needs: they may have observed a particular learning hurdle within their own course that their students have a hard time clearing. They may want to be able to spend less time grading homework for their students and more time providing timely feedback or discussing specific challenges, informed by a clear picture of what students are already good at and where the remaining learning hurdles are. They may want to make sure their students do well on standardized tests. ITS analytics and reports can provide that information (e.g., (Arroyo et al., 2014; Kelly et al., 2013; Segedy, Sulcer, & Biswas, 2010)). They may have been tasked to address a particular learning issue on the website to which they are contributing as volunteer. They may see a good opportunity to address an educational research question in the context of an existing ITS or by building a new one, typically for challenging subject matter. For them, ITS development may be a point solution to a specific educational problem, often a hard educational problem. They may have found the tool online or heard about the tool and want to try it out.

Occasional authors want to get started with the tool really quickly, without having to invest much time in installation and configuration. They want to have their tutors online, accessible to students, really quickly. They are willing to watch some YouTube videos showing how best to use the tool. Some occasional authors may even be willing to attend a week-long summer school but many others prefer to learn by themselves, in their own time and on their own schedule. Occasional authors do not have an e-learning platform in which to deploy their tutors. Therefore, the ITS authoring environment must provide seamless integration between authoring and deployment.

Should ITS authoring tool developers try to cater to these categories of authors within a single ITS authoring tool? In our opinion this is an important unresolved issue in the field of AIEd. It can be difficult, within a single tool, to make authoring simple artifacts really simple while also offering the possibility to author very complex artifacts. Generalized ITS authoring architecture, such as GIFT or CTAT, may be a good way of approaching this challenge, as a range of tools could be plugged in.

Recommendations for GIFT: Support a range of authors. Keep on generalizing!

Conclusions

We presented eight themes that emerge from our 12+ years of experience building CTAT, using CTAT, and assisting others as they use CTAT. CTAT has been used by over 600 authors to build a wide and diverse range of tutors. The themes are meant to be relevant not just to CTAT and GIFT; they touch on a range of issues that are relevant to other ITS authoring tools and ITS architectures as well.

First, in our experience, the use-driven development of authoring tools is key to ensuring that the tools address users' main goals and needs. This strategy entails learning from users' experiences, being driven by their needs when prioritizing development, providing services to help users get started with the tools, and supporting them through education, consulting, an online discussion forum, and tutor hosting services. Second, it is often useful to describe and compare ITS authoring tools in terms of the tutoring behaviors that they support, following a recommendation by VanLehn (2006). For example, clarity regarding which tutoring behaviors are supported may facilitate discussions about authoring efficiency and interoperability. Third, supporting both programmer and non-programmer ITS authoring within a single ITS authoring tool suite can be advantageous. It enables the ITS author to use a simpler tool to build simpler tutors and more complex tools to build more complex tutors, while staying within the same tool suite. Fourth, solution space graphs (or behavior graphs, as they are called in CTAT) can be used to support a variety of authoring functions that may be useful across a range of ITS authoring tools. Fifth, tool/tutor separation as defined by Ritter and Koedinger (1996) supports a form of ITS component interoperability that has proven to be useful in multiple projects. It involves separating the tutor front end ("tool") and back end ("tutor") with standardized communication between them. In CTAT, this separation has multiple advantages. It makes it possible for an author to mix and match front-end and back-end technology options and to provide tutoring within existing interfaces or simulators, after doing the programming necessary to make them adhere to the tool/tutor messaging protocol. Further, this separation has greatly facilitated keeping the tools up-to-date with changing web technologies. Sixth, it is useful if an ITS authoring tool can be interoperable with existing components such as tutor log analysis services (e.g., DataShop) and standard MOOC or e-learning platforms (e.g., EdX or Moodle). Instrumenting ITS authoring tools so they support DataShop logging, a general format for ITS log data makes it possible to use the DataShop analysis tools for knowledge modeling and iterative tutor improvement, and make tutor data sets available to the EDM community for secondary analysis. Further, making ITS authoring tools adhere to e-learning standards such as SCORM and LTI facilitates the use of tutors in many contexts and may help spread ITS technology. Seventh, it may be useful to explore advantages of inner loop and outer loop, with the learner model as communication between them. Eighth, different categories of ITS authoring tool users have different needs; specific ITS authoring tools may need to target particular categories, although generalized architectures with pluggable tools can cater to a very broad audience.

We hope our thoughts can inform useful discussion within the field regarding ITS authoring tools and generalized ITS architectures. Interoperability and generalized architectures may be key in making ITS technology spread.

Acknowledgments

CTAT has been and continues to be a team effort. Ken Koedinger and Bruce McLaren have been influential in shaping CTAT in addition to the co-authors. The research was funded by the National Science Foundation, the Grable Foundation, the Institute of Education Sciences, and the Office of Naval Research. We gratefully acknowledge their contributions.

References

- Adamson, D., Dyke, G., Jang, H., & Rosé, C. P. (2014). Towards an agile approach to adapting dynamic collaboration support to student needs. *International Journal of Artificial Intelligence in Education*, 24(1), 92-124. doi:10.1007/s40593-013-0012-6
- Aleven, V. (2010). Rule-Based cognitive modeling for intelligent tutoring systems. In R. Nkambou, J. Bourdeau, & R. Mizoguchi (Eds.), *Studies in Computational Intelligence: Vol. 308. Advances in intelligent tutoring systems* (pp. 33-62). Berlin, Heidelberg: Springer. doi:10.1007/978-3-642-14363-2_3
- Aleven, V., & Koedinger, K. R. (2013). Knowledge component approaches to learner modeling. In R. Sottilare, A. Graesser, X. Hu, & H. Holden (Eds.), *Design recommendations for adaptive intelligent tutoring systems* (Vol. I, Learner Modeling, pp. 165-182). Orlando, FL: US Army Research Laboratory.
- Aleven, V., McLaren, B. M., & Sewall, J. (2009). Scaling up programming by demonstration for intelligent tutoring systems development: An open-access web site for middle school mathematics learning. *IEEE Transactions on Learning Technologies*, 2(2), 64-78.
- Aleven, V., McLaren, B. M., Sewall, J., & Koedinger, K. R. (2006). The cognitive tutor authoring tools (CTAT): Preliminary evaluation of efficiency gains. In M. Ikeda, K. D. Ashley, & T. W. Chan (Eds.), *Proceedings of the 8th International Conference on Intelligent Tutoring Systems (ITS 2006)* (pp. 61-70). Berlin, Heidelberg: Springer-Verlag. doi:10.1007/11774303_7
- Aleven, V., McLaren, B. M., Sewall, J., & Koedinger, K. R. (2009). A new paradigm for intelligent tutoring systems: Example-Tracing tutors. *International Journal of Artificial Intelligence in Education*, 19(2), 105-154.
- Aleven, V., McLaren, B. M., Sewall, J., Popescu, O., Demi, S., & Koedinger, K. R. (under review). Towards tutoring at scale: Reflections on “A new paradigm for intelligent tutoring systems: Example-Tracing tutors”. *International Journal of Artificial Intelligence in Education*.
- Aleven, V., Roll, I., McLaren, B. M., & Koedinger, K. R. (2010). Automated, unobtrusive, action-by-action assessment of self-regulation during learning with an intelligent tutoring system. *Educational Psychologist*, 45(4), 224-233.
- Aleven, V., Sewall, J., McLaren, B. M., & Koedinger, K. R. (2006). Rapid authoring of intelligent tutors for real-world and experimental use. In Kinshuk, R. Koper, P. Kommers, P. Kirschner, D. G. Sampson, & W. Didderen (Eds.), *Proceedings of the 6th IEEE International Conference on Advanced Learning Technologies (ICALT 2006)* (pp. 847-851). Los Alamitos, CA: IEEE Computer Society.
- Aleven, V., Sewall, J., Popescu, O., Xhakaj, F., Chand, D., Baker, R. S., . . . Gasevic, D. (2015). The beginning of a beautiful friendship? Intelligent tutoring systems and MOOCs: To appear in the *Proceedings of the 17th International Conference on AI in education, AIED 2015*.
- Anderson, J. R., Corbett, A. T., Koedinger, K. R., & Pelletier, R. (1995). Cognitive tutors: Lessons learned. *The Journal of the Learning Sciences*, 4(2), 167-207.
- Arroyo, I., Woolf, B. P., Burelson, W., Muldner, K., Rai, D., & Tai, M. (2014). A multimedia adaptive tutoring system for mathematics that addresses cognition, metacognition and affect. *International Journal of Artificial Intelligence in Education*, 24(4), 387-426. doi:10.1007/s40593-014-0023-y
- Baker, R. S. J. d., Corbett, A. T., & Koedinger, K. R. (2007). The difficulty factors approach to the design of lessons in intelligent tutor curricula. *International Journal of Artificial Intelligence and Education*, 17(4), 341-369.
- Blessing, S., Devasani, S., & Gilbert, S. (2011). Evaluation of webxpst: A browser-based authoring tool for problem-specific tutors. In G. Biswas, S. Bull, J. Kay, & A. Mitrovic (Eds.), *Proceedings of the 15th International Conference on Artificial Intelligence in Education (AIED 2011)* (pp. 423-425). Berlin: Springer.
- Blessing, S. B., Gilbert, S. B., Ourada, S., & Ritter, S. (2009). Authoring model-tracing cognitive tutors. *International Journal of Artificial Intelligence in Education*, 19(2), 189-210.

- Borek, A., McLaren, B. M., Karabinos, M., & Yaron, D. (2009). How much assistance is helpful to students in discovery learning? In U. Cress, V. Dimitrova, & M. Specht (Eds.), *Proceedings 4th European Conference on Technology-Enhanced Learning, EC-TEL 2009* (pp. 391-404). Berlin, Heidelberg: Springer doi:10.1007/978-3-642-04636-0_38
- Cen, H., Koedinger, K. R., & Junker, B. (2007). Is over practice necessary?-improving learning efficiency with the cognitive tutor through educational data mining. In R. Luckin, K. R. Koedinger, & J. Greer (Eds.), *Proceedings of the 13th International Conference on Artificial Intelligence in Education* (pp. 511-518). Amsterdam: IOS Press.
- Chase, C., Marks, J., Bennett, D., & Aleven, V. (under review). The design of the an exploratory learning environment to support Invention. Submitted to the International Workshop on Intelligent Support in Exploratory and Open-ended Learning Environments, to be held in conjunction with the 17th International Conference on Artificial Intelligence in Education (AIED 2015).
- Corbett, A., Kauffman, L., MacLaren, B., Wagner, A., & Jones, E. (2010). A cognitive tutor for genetics problem solving: Learning gains and student modeling. *Journal of Educational Computing Research*, 42(2), 219-239.
- Corbett, A., McLaughlin, M., & Scarpinatto, K. C. (2000). Modeling student knowledge: Cognitive tutors in high school and college. *User Modeling and User-Adapted Interaction*, 10, 81-108.
- Corbett, A. T., & Anderson, J. R. (1995). Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Modeling and User-Adapted Interaction*, 4(4), 253-278.
- D'Mello, S. K., & Graesser, A. C. (2014). Feeling, thinking, and computing with affect-aware learning. In R. A. Calvo, S. K. D'Mello, J. Gratch, & A. Kappas (Eds.), *The oxford handbook of affective computing* (pp. 419-434). Oxford University Press. doi: 10.1093/oxfordhb/9780199942237.013.032.
- Forlizzi, J., McLaren, B. M., Ganoe, C., McLaren, P. B., Kihumba, G., & Lister, K. (2014). Decimal point: Designing and developing an educational game to teach decimals to middle school students. In C. Busch (Ed.), *Proceedings of the 8th European Conference on Games Based Learning: ECGBL 2014* (pp. 128-135). Reading, UK: Academic Conferences and Publishing International.
- Friedman-Hill, E. (2003). *JESS in action*. Greenwich, CT: Manning. doi:friedman-hill
- Heffernan, N. T., & Heffernan, C. L. (2014). The ASSISTments ecosystem: Building a platform that brings scientists and teachers together for minimally invasive research on human learning and teaching. *International Journal of Artificial Intelligence in Education*, 24(4), 470-497. doi:10.1007/s40593-014-0024-x
- Kelly, K., Heffernan, N., Heffernan, C., Goldman, S., Pellegrino, J., & Goldstein, D. S. (2013). Estimating the effect of web-based homework. In H. C. Lane, K. Yacef, J. Mostow, & P. Pavlik (Eds.), *Proceedings of the 16th International Conference on Artificial Intelligence in Education AIED 2013* (pp. 824-827). Berlin, Heidelberg: Springer.
- Kodaganallur, V., Weitz, R., & Rosenthal, D. (2005). A comparison of model-tracing and constraint-based intelligent tutoring paradigms. *International Journal of Artificial Intelligence in Education*, 15(1), 117-144.
- Koedinger, K. R. (2002). Toward evidence for instructional design principles: Examples from Cognitive Tutor Math 6. In D. Mewborn, P. Sztajn, D. Y. White, H. G. Wiegel, R. L. Bryant, & K. Nooney (Eds.), *Proceedings of the 24th Annual Meeting of the North American Chapter of the International Group for the Psychology of Mathematics Education* (pp. 21-49). Columbus, OH: ERIC Clearhinghouse for Science, Mathematics, and Environmental Education.
- Koedinger, K. R., Aleven, V., Heffernan, N., McLaren, B., & Hockenberry, M. (2004). Opening the door to non-programmers: Authoring intelligent tutor behavior by demonstration. In J. C. Lester, R. M. Vicario, & F. Paraguaçu (Eds.), *Proceedings of seventh International Conference on Intelligent Tutoring Systems, ITS 2004* (pp. 162-174). Berlin: Springer.
- Koedinger, K. R., Aleven, V. A., & Heffernan, N. (2003). Toward a rapid development environment for cognitive tutors. In U. Hoppe, F. Verdejo, & J. Kay (Eds.), *Proceedings of the 11th International Conference on Artificial Intelligence in Education (AIED 2003)* (pp. 455-457). Amsterdam: IOS Press.
- Koedinger, K. R., Baker, R. S. J. d., Cunningham, K., Skogsholm, A., Leber, B., & Stamper, J. (2010). A data repository for the EDM community: The PSLC datashop. In S. Ventura, C. Romero, M. Pechenizkiy, & R. S. J. d. Baker (Eds.), *Handbook of educational data mining* (pp. 43-55). Boca Raton, FL: CRC Press.
- Koedinger, K. R., Stamper, J. C., McLaughlin, E. A., & Nixon, T. (2013). Using data-driven discovery of better student models to improve student learning. In H. C. Lane, K. Yacef, J. Mostow, & P. Pavlik (Eds.), *Proceedings of the 16th International Conference on Artificial Intelligence in Education AIED 2013* (pp. 421-430). Berlin, Heidelberg: Springer. doi:10.1007/978-3-642-39112-5_43

- Koedinger, K. R., Suthers, D. D., & Forbus, K. D. (1999). Component-based construction of a science learning space. *International Journal of Artificial Intelligence in Education*, 10(3), 292-313.
- Kumar, R., & Rose, C. P. (2011). Architecture for building conversational agents that support collaborative learning. *Learning Technologies, IEEE Transactions on*, 4(1), 21-34. doi:10.1109/TLT.2010.41
- Long, Y., & Aleven, V. (2013). Supporting students' self-regulated learning with an open learner model in a linear equation tutor. In H. C. Lane, K. Yacef, J. Mostow, & P. Pavlik (Eds.), *Proceedings of the 16th International Conference on Artificial Intelligence in Education (AIED 2013)* (pp. 249-258). Berlin: Springer.
- Long, Y., & Aleven, V. (2014). Gamification of joint student/system control over problem selection in a linear equation tutor. In S. Trausan-Matu, K. E. Boyer, M. Crosby, & K. Panourgia (Eds.), *Proceedings of the 12th International Conference on Intelligent Tutoring Systems, ITS 2014* (pp. 378-387). New York: Springer. doi:10.1007/978-3-319-07221-0_47
- Lovett, M. C. (1998). Cognitive task analysis in the service of intelligent tutoring system design: A case study in statistics. In B. P. Goettle, H. M. Halff, C. L. Redfield, & V. J. Shute (Eds.), *Proceedings of the 4th International Conference on Intelligent tutoring systems, ITS 1998* (pp. 234-243). Berlin: Springer Verlag.
- Ma, W., Adesope, O. O., Nesbit, J. C., & Liu, Q. (2014). Intelligent tutoring systems and learning outcomes: A meta-analysis. *Journal of Educational Psychology*, 106(4), 901. doi:10.1037/a0037123
- MacLellan, C., Koedinger, K. R., & Matsuda, N. (2014). Authoring tutors with simstudent: An evaluation of efficiency and model quality. In S. Trausan-Matu, K. E. Boyer, M. Crosby, & K. Panourgia (Eds.), *Proceedings of the 12th International Conference on Intelligent Tutoring Systems, ITS 2014* (pp. 551-560). Berlin: Springer. doi:10.1007/978-3-319-07221-0_66
- Mathan, S. A., & Koedinger, K. R. (2005). Fostering the intelligent novice: Learning from errors with metacognitive tutoring. *Educational Psychologist*, 40(4), 257-265.
- Matsuda, N., Cohen, W. W., & Koedinger, K. R. (2005). Applying programming by demonstration in an intelligent authoring tool for cognitive tutors. In D. Oblinger, T. Lau, Y. Gil, & M. Bauer (Eds.), AAAI workshop on human comprehensible machine learning (technical report WS-05-04) (pp. 1-8). Menlo Park, CA: AAAI Association.
- Matsuda, N., Cohen, W. W., & Koedinger, K. R. (2015). Teaching the teacher: Tutoring simstudent leads to more effective cognitive tutor authoring. *International Journal of Artificial Intelligence in Education*, 25(1), 1-34. doi:10.1007/s40593-014-0020-1
- McLaren, B. M., Adams, D., Durkin, K., Goguadze, G., Mayer, R. E., Rittle-Johnson, B., . . . Velsen, M. V. (2012). To err is human, to explain and correct is divine: A study of interactive erroneous examples with middle school math students. In A. Ravenscroft, S. Lindstaedt, C. Delgado Kloos, & D. Hernández-Leo (Eds.), *21st century learning for 21st century skills: 7th European Conference of Technology Enhanced learning, EC-TEL 2012* (pp. 222-235). Berlin, Heidelberg: Springer. doi:10.1007/978-3-642-33263-0_18
- Mitrovic, A., & Ohlsson, S. (1999). Evaluation of a constraint-based tutor for a database language. *International Journal of Artificial Intelligence in Education*, 10(3-4), 238-256.
- Mitrovic, A., Koedinger, K. R., & Martin, B. (2003). A comparative analysis of cognitive tutoring and constraint-based modeling. In P. Brusilovsky, Corbett, & de Rosis (Eds.), *Proceedings of the 9th International Conference on User Modeling (UM 2003)* (pp. 313-322). Springer Berlin Heidelberg. doi:10.1007/3-540-44963-9_42
- Mitrovic, A., Suraweera, P., Martin, B., Zakharov, K., Milik, N., & Holland, J. (2006). Authoring constraint-based tutors in ASPIRE. In M. Ikeda, K. D. Ashley, & T. W. Chan (Eds.), Lecture Notes in Computer Science: *Proceedings of the 8th International Conference on Intelligent Tutoring Systems, ITS 2006* (Vol. 4053, pp. 41-50). Berlin: Springer. Retrieved from Google Scholar.
- Newell, A., & Simon, H. A. (1972). *Human problem solving*. Englewood Cliffs, NJ: Prentice-Hall.
- Olsen, J. K., Belenky, D. M., Aleven, V., Rummel, N., Sewall, J., & Ringenberg, M. (2014). Authoring tools for collaborative intelligent tutoring system environments. In S. Trausan-Matu, K. E. Boyer, M. Crosby, & K. Panourgia (Eds.), *Proceedings of the 12th International Conference on Intelligent Tutoring Systems, ITS 2014* (pp. 523-528). Berlin: Springer. doi:10.1007/978-3-319-07221-0_66
- Pane, J. F., Griffin, B. A., McCaffrey, D. F., & Karam, R. (2013). Effectiveness of cognitive tutor algebra I at scale. *Educational Evaluation and Policy Analysis*, 0162373713507480. doi:10.3102/0162373713507480
- Pavlik, P. I., Brawner, K., Olney, A., & Mitrovic, A. (2013). A review of student models used in intelligent tutoring systems. In R. Sottilare, A. Graesser, X. Hu, & H. Holden (Eds.), *Design recommendations for adaptive intelligent tutoring systems* (Vol. I, Learner Modeling, pp. 39-68). Orlando, FL: US Army Research Laboratory.

- Rau, M., Aleven, V., & Rummel, N. (2013). Interleaved practice in multi-dimensional learning tasks: Which dimension should we interleave? *Learning and Instruction*, 23, 98-114. doi:learninstruc.2012.07.003
- Rau, M. A., Aleven, V., & Rummel, N. (2015). Successful learning with multiple graphical representations and self-explanation prompts. *Journal of Educational Psychology*, 107(1), 30-46. doi:10.1037/a0037211
- Rau, M. A., Aleven, V., Rummel, N., & Pardos, Z. (2014). How should intelligent tutoring systems sequence multiple graphical representations of fractions? A multi-methods study. *International Journal of Artificial Intelligence in Education*, 24(2), 125-161.
- Razzaq, L., Patvarczki, J., Almeida, S. F., Vartak, M., Feng, M., Heffernan, N. T., & Koedinger, K. R. (2009). The assistance builder: Supporting the life cycle of tutoring system content creation. *IEEE Transactions on Learning Technologies*, 2(2), 157-166. Retrieved from <http://doi.ieeecomputersociety.org/10.1109/TLT.2009.23>
- Rice, W. (2011). Moodle 2.0 e-learning course development: A complete guide to successful learning using Moodle. Birmingham, UK: Packt Publishing Ltd.
- Ritter, S. (1997). Communication, cooperation, and competition among multiple tutor agents. In B. du Boulay & R. Mizoguchi (Eds.), *Proceedings of the Artificial intelligence in education, AI-ED 97* (pp. 31-38). Amsterdam: IOS Press.
- Ritter, S., & Koedinger, K. R. (1996). An architecture for plug-in tutor agents. *International Journal of Artificial Intelligence in Education*, 7(3-4), 315-347.
- Roll, I., Aleven, V., & Koedinger, K. R. (2010). The invention lab: Using a hybrid of model tracing and constraint-based modeling to offer intelligent support in inquiry environments. In V. Aleven, J. Kay, & J. Mostow (Eds.), *Lecture Notes in Computer Science: Proceedings of the 10th International Conference on Intelligent Tutoring Systems, ITS 2010* (Vol. 1, pp. 115-124). Berlin: Springer.
- Rus, V., D'Mello, S., Hu, X., & Graesser, A. (2013). Recent advances in conversational intelligent tutoring systems. *AI Magazine*, 34(3), 42-54.
- Segedy, J., Sulcer, B., & Biswas, G. (2010). Are files ready for the classroom? Bringing teachers into the feedback loop. In V. Aleven, J. Kay, & J. Mostow (Eds.), *Proceedings, 10th International Conference on Intelligent Tutoring Systems, ITS 2010* (Part II, pp. 405-407). Berlin, Heidelberg: Springer. doi:10.1007/978-3-642-13437-1_85
- Sottilare, R.A., Brawner, K.W., Goldberg, B.S. & Holden, H.K. (2012). The Generalized Intelligent Framework for Tutoring (GIFT). Orlando, FL: U.S. Army Research Laboratory – Human Research & Engineering Directorate (ARL-HRED).
- Stamper, J., Eagle, M., Barnes, T., & Croy, M. (2013). Experimental evaluation of automatic hint generation for a logic tutor. *International Journal of Artificial Intelligence in Education*, 22(1-2), 3-17. doi:10.3233/JAI-130029
- Steenbergen-Hu, S., & Cooper, H. (2013). A meta-analysis of the effectiveness of intelligent tutoring systems on K–12 students' mathematical learning. *Journal of Educational Psychology*, 105(4), 970-987. doi:10.1037/a0032447
- Steenbergen-Hu, S., & Cooper, H. (2014). A meta-analysis of the effectiveness of intelligent tutoring systems on college students' academic learning. *Journal of Educational Psychology*, 106(2), 331-347. doi:10.1037/a0034752
- VanLehn, K. (2006). The behavior of tutoring systems. *International Journal of Artificial Intelligence in Education*, 16(3), 227-265.
- VanLehn, K. (2011). The relative effectiveness of human tutoring, intelligent tutoring systems, and other tutoring systems. *Educational Psychologist*, 46(4), 197-221.
- VanLehn, K., Graesser, A. C., Jackson, G. T., Jordan, P., Olney, A., & Rosé, C. P. (2007). When are tutorial dialogues more effective than reading? *Cognitive Science*, 31(1), 3-62. doi:10.1080/03640210709336984
- Waalkens, M., Aleven, V., & Taatgen, N. (2013). Does supporting multiple student strategies lead to greater learning and motivation? Investigating a source of complexity in the architecture of intelligent tutoring systems. *Computers & Education*, 60(1), 159 - 171. doi:10.1016/j.compedu.2012.07.016
- Walkington, C. A. (2013). Using adaptive learning technologies to personalize instruction to student interests: The impact of relevant contexts on performance and learning outcomes. *Journal of Educational Psychology*, 105(4), 932.

CHAPTER 23 Usability Considerations and Different User Roles in the Generalized Intelligent Framework for Tutoring

Anne M. Sinatra¹, Heather K. Holden², Scott J. Ososky¹ & Karissa Berkey³

¹US Army Research Laboratory; ²Mount Washington College, ³Stetson University

The Challenges of Being “Generalized”

One of the most unique and challenging aspects of the design of the Generalized Intelligent Framework for Tutoring (GIFT) can be found in its name: “Generalized.” The main goal of GIFT is to provide a framework that allows for the creation of domain-independent intelligent tutoring systems (ITSs). Additionally, GIFT includes functionality that can result in using it as an experimental testbed (for experiments in many domains) or to specifically analyze the impact of changing out different intelligent tutoring components (Sottilare, Graesser, Hu & Holden, 2013). While having these goals leads to a system with a great amount of flexibility, it also leads to very important design challenges⁶.

User Roles in Gift

The traditional components and modules of ITSs are present within the design of GIFT; however, special care was taken to make sure that they remain domain-independent. The domain module of GIFT is the only one that is specific to the content that an individual is intending to teach (Sottilare, Brawner, Goldberg & Holden, 2012). The individual who is authoring adaptive feedback can write specific rules to link assessments of learner state to the domain. By following this design goal, GIFT can successfully be used to create domain-independent ITSs, which allow users to bring their own content and plug it into the system. However, while this design meets the requirement of providing a generalized system for creating ITSs, additional design decisions should be made to guide the different functionalities and user roles that are created by GIFT’s flexibility.

There are three main categories of individuals who interact with GIFT:

- Students
- Authors
- Researchers/analysts

In most cases, *students* will only encounter an exported tutor that has successfully been designed with GIFT. *Authors* will work with the full version of GIFT and its authoring tools. They can design both adaptive and non-adaptive course content using GIFT, and then ultimately export their tutors for student use. *Researchers* will also work with the full version of GIFT; however, their goals may be different than authors’ goals. There are a wide range of disciplines that may engage with research using GIFT. Researchers may use GIFT to run psychology experiments (Sinatra, 2014), or they may be testing specific configurations of an intelligent tutor to discover what the best learning outcome will be. Researchers in varying disciplines may have different expectations and design requirements for studies that they wish to use GIFT to run. While GIFT does have a number of different user interfaces, it currently does not have ones that are dedicated to specific types of users. Part of the challenge of this generalized system is

⁶ As a note to orient the reader, at the time of the writing of this chapter the latest version of GIFT was GIFT 2014-2.

deciding how and when to make divisions. Namely, what types of interfaces should individuals interact with? What types of authoring tools should they have access to? How should we sub-divide groups of authors? Should we break users down into beginner and advanced users?

Authoring Tools in Gift

An area of GIFT that has made great strides toward becoming more user-friendly is its authoring tools. In early versions of GIFT the authoring tools were primarily a set of extensible markup language (XML) editors that allow for creation of elements of ITSs such as course files, sensor files, and adaptive feedback files (domain knowledge files or DKFs). While these tools have carried over to later versions of GIFT, newer versions are also being developed. The GIFT Authoring Tool (GAT) is ultimately being developed to be the one-stop course development tool that provides the individual access to all of the tools that they will require while creating their intelligent tutor. The early iterations of the GAT include a new user interface with drop-down menus, and explanations/prompts that explain the different elements that individuals can add to their courses. Additionally, a new DKF authoring tool debuted with GIFT 2014-2. This tool makes the complex process of creating adaptive feedback more straightforward for both new and returning GIFT users. See Figure 1 for the original Course Authoring Tool (CAT) and Figure 2 for the same course loaded in the GAT.

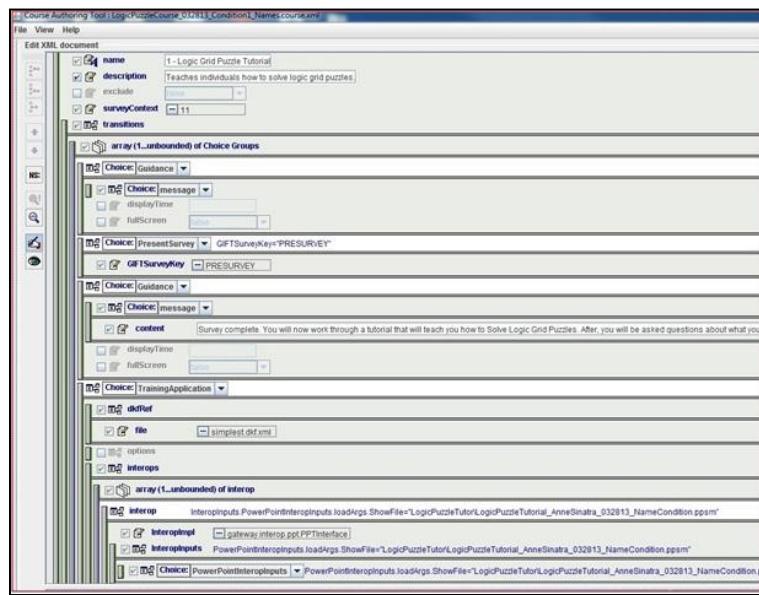


Figure 1. A course loaded in the XML editor style course authoring tool

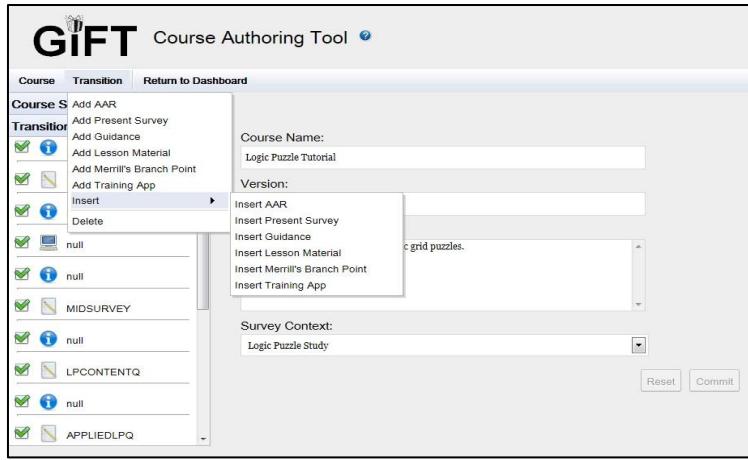


Figure 2. The same course loaded in the new GAT

For future development of GIFT, it is important to start establishing the user groups that are likely to interact with it, and how their login experiences should be different. For instance, for a student, it is important that once a tutor is installed, they know where to click to get it to successfully run. Additionally, it is important for an author to login to the system and be presented with an authoring tools menu, as opposed to the student login as was present in early versions of GIFT. One place to start in regard to determining what type and level of user groups to design for is to examine other ITS authoring tools that exist. Many of these tools have chosen to go with what-you-see-is-what-you-get (WYSIWYG) style authoring tools and have offered users templates for commonly used features (Brawner & Sinatra, 2014). These features allow for both advanced and beginning users to interact successfully with their systems. Additionally, it would be advantageous for GIFT's designers to examine established usability principles and to try to make future changes to their interfaces consistent with usability guidelines. By lining up the design of authoring tools and user interfaces with established usability principles the system should become easier to use. Further, this ease of use will lead to increased adoption of the system by new users.

Understanding Usability Principles and Designing for the User Experience

The previous section illustrated how designing a system with only functionality in mind is just one of many critical elements in supporting users of that system. For instance, developing a complex software application such as GIFT to provide an efficient, easy to use, and pleasurable experience is a daunting task. Moreover, creating a positive experience for multiple user groups (i.e., students, authors, researchers) adds additional complexity. To that end, this section takes a closer look at usability principles and heuristics. This section also examines the role of utility and usability in support of the overall user experience.

Consider Jordan's (2000) three-level hierarchy of consumer needs for a product: *functionality*, *usability*, and *pleasure*. The base level, *functionality*, declared that "a product will be useless if it does not contain appropriate functionality; a product cannot be usable if it does not contain the functions necessary to perform the tasks for which it is needed" (p. 5). The next level, *usability*, stated that "once people had become used to having appropriate functionality, they then wanted products that were easy to use" (p. 6). *Pleasure* is the highest level in Jordan's hierarchy, which declared that "having become used to usable products, it seems that inevitable that more people will soon want something more...products that bring not only functional benefits, but also emotional ones" (p. 6).

As described above, GIFT provides the functionality required to create an adaptive course. Therefore, GIFT has *utility* for that purpose. Sharp and colleagues (2007) described utility as “the extent to which the product provides the right kind of functionality so that users can do what they need or want to do” (p.22). That definition closely aligns with Jordan’s (2000) first level of functionality in the hierarchy of consumer need. Utility is an important, complementary attribute to usability. Thus, software that is usable, and allows the user to achieve their desired goals might be described as *useful* (Nielsen, 2012).

Current development efforts in GIFT include improving the usability of the system for our target user groups. By comparison, *usability* is “the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use” (ISO 9241-11; Jordan 1998). Similarly, Nielsen (2012) operationalized usability as *methods* for “improving ease-of-use during the design process.” Nielsen (2012) further characterized usability by the following five components:

Learnability: How easy is it for users to accomplish basic tasks the first time they encounter the design?

Efficiency: Once users have learned the design, how quickly can they perform tasks?

Memorability: When users return to the design after a period of not using it, how easily can they reestablish proficiency?

Errors: How many errors do users make, how severe are these errors, and how easily can they recover from the errors?

Satisfaction: How pleasant is it to use the design?

These attributes can be measured through objective and subjective data in order to discover problems improve the overall usefulness of a product or system. With respect to GIFT, improving upon the usability of the current system by reducing the time, skill, and effort required to create and manage adaptive course content should contribute to a positive user experience with the system.

User experience, then, relates to how a user feels about the overall interaction with the target system. User experience is described as “a person’s perceptions and responses that result from the use or anticipated use of a product, system, or service” (ISO 9241-210). User experience links the two highest levels in Jordan’s (2000) hierarchy (usability and pleasure) and indicates users’ responses to interaction with a particular system. These responses are complicated in nature as they are a combination of (a) users’ individual perceptions of the system in terms of their attitudes, motivations, and individual needs; (b) characteristics of the system, such as purpose, functionality, and usability; and (c) contextual dependencies in terms of task and environment situations (Norman, 2002). User experience goals are more subjective in nature and can include elements evaluating the degree to which a user perceives the technology as satisfying, enjoyable, engaging, motivating, aesthetically pleasing, cognitively stimulating, supportive of creativity, etc. (Sharp et al., 2007). Negative perceptions of frustration, annoyance, and boredom exhibited by a technology are also metrics that can be used to capture user experience.

Design Principles

Creating a positive user experience for the different types of GIFT users outlined in this chapter requires, in part, careful consideration of design for usability, aesthetics, and symbolism. Here, principles are highlighted that help guide and inform those design goals. Don Norman, an academic in the fields of cognitive science and design and usability engineering, identified the first set of design principles for

interaction design. Don Norman's "Principles of Design" include visibility; feedback; constraints; mapping consistency; and affordance (Norman, 2002).

Visibility

The visibility principle is based on the notion that usability and learnability are enhanced when the user can readily see available options and commands. Most options and commands, especially key functions, should not be hidden, but should be visible and placed in a logical order. Sometimes complex systems, such as GIFT, may have too many functions to visually represent all of them at once. One suggestion for complex systems is to consider using drop-down/pull-down menus for functions that are not always needed. The function will be out of sight, but will easily be available as needed. When considering the different user roles of GIFT, all visible options and commands should only pertain to key functions for the specific role.

Feedback

Feedback pertains to the system providing adequate confirmation of actions being performed by the user. Feedback can be provided with any combination of messages, sounds, highlighting, and/or animation. It is important to provide feedback immediately to the user about the successful or unsuccessful results of their actions. Norman (2002) suggests that there are two types of feedback as system can perform: activational feedback (i.e., evidence that a control was activated successfully, such as a button is pressed or a menu option is selected) and behavioral feedback (i.e., evidence that the activation or adjustment of the control has now had some effect in the system). The GIFT Monitor Module is one interface in which activational feedback is provided. The interface was designed to provide diagnostic information related to the operational status of GIFT components. In that interface, active (i.e., running) modules are indicated with a green icon, while red icons indicate modules that are offline (modules can easily be restarted from the same interface). Similarly, with respect to behavioral feedback, when XML data are validated with the GIFT authoring tools, the system lets the user know when the validation is complete or if a file needs to be inspected for errors. This feedback could be improved, for instance, by indicating the specific line(s) within an XML file in which an error was detected. Finally, from the student perspective, GIFT provides continuous feedback on their actions and sequences the learning material accordingly.

Consistency

Sharp, Rogers, and Preece (2007) described consistency as "designing interfaces to have similar operations and use similar elements for achieving similar tasks" (p. 32). Consistency is critical for learnability because it helps users recognize and apply existing patterns when new situations arise. Inconsistency can have a negative impact on user perceptions, especially when things do not work the way the user thinks they should. Attention to consistency can inspire user's confidence in the system by supporting the impression that the system design is logical and rational. Many of GIFT's tools are consistent within release versions. However, the interface of GIFT slightly changes between releases (resulting from continuous development and improvement), which may lead to inconsistency between releases, and ultimately, user confusion. For instance, GIFT 2014-2 moved the authoring tools to a control panel that is independent from the primary startup program; further, the reasoning behind this change has not been presented to the user. As such, it is important to be cautious of interface *redesign* and provide users with proper information on future changes and the reasons behind those changes.

Constraints

Constraints prevent invalid data from being entered or invalid actions from being performed. Having proper constraints prevents system errors and system failures. There is a desired balance between providing users with flexibility and implementing constraints to ensure system reliability. GIFT designers have done a sound job of providing constraints; however, as GIFT continues to expand, keeping control of these constraints can be difficult. That difficulty in balancing flexibility and constraints will increase as GIFT begins to incorporate new functionality and interfaces.

Affordances

Affordances refer to inherent visual traits of an object that suggest how to interact with it. For example, chairs afford sitting, however the surface of a school desk does not. One can consider everything that users interact with in the system (i.e., buttons, scrollbars, the mouse and keyboard, etc.) as an affordance. The GIFT program has many affordances, such as tooltips, throughout the application. Additional tooltips and on-demand prompts are recommended to support usability. For example, after GIFT is installed, a new user might ask, “What is next? How do I start GIFT?” The new user might not know, for example, to *launchActiveMQ.bat* and *launchMonitor.bat* to start GIFT. Placing shortcut icons for different tools on the desktop is one recommendation for improving access to authoring, student, and researcher interfaces.

Other Rules for Designing User Interfaces

Norman’s design principles helped pave the way for designing systems from a usability and user-centered perspective. These design principles complement each other and serve as the basis for human-centered design. Other relevant sets of guidelines to consider in usability / user experience design include the following:

- **Jakob Nielsen’s “10 Usability Heuristics for User Interface Design”** include visibility of the system status; match between the system and the real world; user control and freedom; consistency and standards; error prevention; recognition rather than recall; flexibility and efficiency of use; aesthetic and minimalist design; help for users recognize, diagnose, and recover from errors; and help and documentation (Nielsen, 2005).
- **Ben Shneiderman’s “Eight Golden Rules of Interface Design”** include strive for consistency; enable frequent users to use shortcuts; offer informative feedback; design dialogue to yield closure; offer simple error handling; permit easy reversal of actions; support internal locus of control; and reduce short-term memory load (Shneiderman, Plaisant, Cohen & Jacobs, 2009).

The common principles within each of these sets of guidelines are: *visibility*, *consistency*, and *feedback*. The prevalence of these principles further emphasizes their importance for system design, development, and implementation. As GIFT’s user interface continues to evolve, each of these principles will be of critical importance in order to positively influence each of the user experiences that GIFT provides.

Connections/Recommendations for GIFT

Since GIFT has become well established and gone through many iterations of design, it is an ideal time to consider a heuristic/expert usability evaluation of GIFT. Those evaluations, also known as usability audits, are conducted by usability experts. Those evaluations are used to identify issues with the user interface and generate recommendations based on industry standards and best practices. The findings of

heuristic evaluations may also be used to develop testing with actual users from populations of interest, to examine their expectations of how to navigate and interact with the system.

In order to conduct these usability analyses, it is important not to lose sight of GIFT's different user groups. The skill levels and user requirements will vary between groups. For example, instructional designers and subject matter experts may have different expectancies regarding authoring tools and navigation of the system. Additionally, the functions that are used by an author who is instructing a class may vary greatly from the tools that are used by a researcher. One of the most important questions that it is necessary for GIFT to answer is what level of support they want to provide for each type of user. For instance, design for usability may result in an open-ended experience for an advanced user, while a novice user may benefit from a wizard-like experience.

One “non-invasive” way to start designing for users with less experience is through supplementary materials that can assist the beginning user with the system. One of these supplementary materials would be to include a simple “how-to” guide to assist users upon their initial download of the system. While GIFT currently has *readme* files, they should be examined/rephrased to streamline and clarify the processes that the user needs to take to initially begin working with the system. Creating an introductory set of brief directions on how to open the system, how to access specific interfaces, and complete basic tasks would help beginner users learn the system without resorting to trial-and-error methods. While the user is interacting with the system, it would be beneficial if there was a detailed help function that users could click on that would bring up directions for the tool they are interacting with, complete with a searchable version of the documents. With respect to authoring tools, new users may benefit from a few simple example courses, which can be quickly modified to generate unique course content. This approach would be easier than creating an entirely new course. Additionally, templates can be generated that require less work from the user, but are aimed at specific tasks that an instructor or researcher may want to engage in with the system.

As GIFT moves toward designing distinct user experiences for its user groups, we offer Figure 3 as a suggestion of functions that will be most relevant to students, authors, and researchers.

	Student	Author	Researcher
Student/Participant Interface	Login X		
GIFT Authoring Tool		X	X
Survey Authoring System		X	X
Event Reporting Tool			X

Figure 3. Tools and interfaces that are most relevant to different user groups.

Students will primarily interact with the login interface, whereas authors and researchers will use more advanced features. Of these advanced features, the GAT and Survey Authoring System are very relevant. However, these systems may be used with very different purposes by authors and researchers, respectively. Further, while the author may extract some survey data using the Event Reporting Tool (ERT), a researcher is much more likely to interact with it for an extended period of time and use many of its functions. As GIFT's design and functionality continue to move forward, it is important to think in

terms of the types of users that will interact with it and their desired goals with the system. Keeping these groups and their expectations in mind will lead to improved interface design, positive user experiences, and the continued growth of the GIFT community.

References

- Brawner, K.W. & Sinatra, A.M. (2014). Intelligent Tutoring System Authoring Tools: Harvesting the Current Crop and Planting Seeds for the Future. In Workshop Proceedings of the 12th International Conference on Intelligent Tutoring Systems, Honolulu, HI, June 2014.
- Jordan, P. (1998). Human Factors for Pleasure in Product Use. *Applied Ergonomics*, 29(1), 25-33.
- Jordan, P. (2000). *Designing Pleasurable Products: An Introduction to the New Human Factors*. London: Taylor and Francis.
- Nielsen, J. (2005). Ten Usability Heuristics for User Interface Design. Retrieved April 2014, from www.nngroup.com/articles/ten-usability-heuristics
- Nielsen, J. (2012). Usability 101: Introduction to Usability. Retrieved September 30, 2014, from <http://www.nngroup.com/articles/usability-101-introduction-to-usability>
- Norman, D. (2002). *The Design of Everyday Things*. New York, NY: Basic Books.
- Sharp, H., Rogers, Y. & Preece, J. (2007). *Interaction Design: Beyond Human-Computer Interaction*: 2nd Edition John Wiley & Sons, Inc.
- Shneiderman, B., Plaisant, C., Cohen, M. & Jacobs, S. (2009). *Designing the User Interface: Strategies for Effective Human-Computer Interaction* (5th ed.). USA: Prentice Hall.
- Sinatra, A.M. (June 2014). The Research Psychologist's Guide to GIFT. Proceedings of *GIFT Symposium 2* in Pittsburgh, PA, June 2014.
- Sottilare, R.A., Brawner, K.W., Goldberg, B.S. & Holden, H.K. (2012). The Generalized Intelligent Framework for Tutoring (GIFT). Orlando, FL: U.S Army Research Laboratory Human Research & Engineering Directorate (ARL-HRED).
- Sottilare, R., Graesser, A., Hu, X., and Holden, H. (Eds.). (2013). Preface in *Design Recommendations for Intelligent Tutoring Systems: Volume 1 - Learner Modeling* (pp. i - xiii). Orlando, FL: U.S. Army Research Laboratory.

Chapter 24 Invisible Intelligent Authoring Tools

Stephen B. Gilbert¹, Stephen B. Blessing²

¹Iowa State University; ²University of Tampa

Motivation

Imagine that you, an expert in technology and in the learning sciences, have decided to help your colleagues pass on their expertise to others by helping them build intelligent tutors systems (ITSs). Your expert colleagues can be in only one place at a time, and an ITS would multiply the impact of their expertise better than an online video, since an ITS can personalize the instruction. ITSs have demonstrated significant learning gains in a variety of disciplines, after all (Anderson, 1989; Koedinger, 1997; Lesgold, Lajoie, Bunzo & Eggan, 1992; Ritter, Kulikowich, Lei, McGuire & Morgan, 2007; VanLehn, et al., 2005), so this approach makes sense.

As you reflect on who these “expert colleagues” really are, you decide to focus on science, technology, engineering, and mathematics (STEM) topics, since the US has a dire need for more STEM expertise (Institute of Medicine, National Academy of Sciences & National Academy of Engineering, 2007), and since a wide variety of people have expertise that they would like to share with others, you’d like to focus on four kinds of experts: university faculty, K12 teachers, professional instructional designers and trainers, and high school students. Some of these experts will be reflective practitioners (Schön, 1983), who will bring a rich conceptual repertoire to the design of the tutor, while others will lack a conceptual model of the domain, much less the tutoring process. For this reason it is critical that the authoring tools be intelligent, i.e., able to adapt to the author.

You include the students as experts because you know that students can learn so effectively through the process of teaching and peer-mentoring (Biswas, Leelawong, Schwartz, Vye & The Teachable Agents Group at Vanderbilt, 2005; Crouch & Mazur, 2001), as well as from design-based activities (Kolodner, et al., 2003; Resnick, et al., 2009; Vattam & Kolodner, 2011). In fact, you realize, the process of formalizing knowledge into a particular representation can change the representation, so it would be worth studying all of your experts along the way, both to see if you can learn more about the basics of their conceptual change and to make sure that the tools you provide don’t force undesired conceptual change on them. You know from Don Norman (1988) that a gap between your experts’ expectations of your tools and their actual experiences with them will make the tools feel unnatural and frustrate your colleagues. Also, only the K12 teachers have actually received significant instruction on how to teach, so the authoring tools would have to incorporate good pedagogy implicitly, and the tutors that result from these tools would need to be evaluated for learning gains.

You then search the Internet for existing authoring tools for creating ITSs, and the results are disappointing. You find a summary of previous ITS authoring tools (Murray, Blessing & Ainsworth, 2003), but it offers more of a history than a guide to available tools. One chapter (Murray, 2003b) does offer lessons learned and guidance for the design of the ideal authoring tool; you’ll keep that in mind. There are more recent search hits for ITS authoring tools, but most are academic papers, not software that’s usable right now. Plus, some are all constrained to a specific domain, e.g., authoring algebra tutors. In terms of actual existing tutor authoring tools that you can sit down and use, six systems float to the top: Cognitive Tutor Authoring Tools (CTAT) (Aleven, Sewall, McLaren & Koedinger, 2006), the Extensible Problem-Solving Tutor (xPST) (Gilbert, Blessing & Kodavali, 2009), Authoring Software Platform for Intelligent Resources in Education (ASPIRE) (Mitrovic, et al., 2009), SimStudent (Matsuda, Cohen,

Sewall, Lacerda & Koedinger, 2007), ASSISTments (Razzaq, et al., 2009), and the Generalized Intelligent Framework for Tutoring (GIFT) (Sottilare, 2012).

As you examine these systems more carefully, however, you realize that none of them meet your desires entirely. The tools take different approaches to enabling the author to input and structure her knowledge. Some have a graphical user interface (GUI)-based system and some have what looks more like programming source code. Some might be easy for your experts to use to create simple problems, but require more computational thinking for more complex ones or for creating tutors that apply more generally. You find a comparison of the usability of CTAT and xPST (Devasani, Gilbert & Blessing, 2012), pointing out pros and cons to GUI vs. text-based approaches to tutor authoring, depending on the domain. That analysis suggests that the ideal ITS authoring tool, much like Adobe Dreamweaver or Microsoft Visual Studio, with both code views and views of the interface, should have multiple representations of content with which to interact. But the comparison article omits discussion of experts' individual differences. Surely two of your colleagues, even if they were experts in the same domain, might have different preferences of the kind of software they would like to use to represent their knowledge.

Delving more deeply into these authoring tools makes you realize the wide variety of interfaces for which ITSs have been built: equations and graphs, geometry and other diagrams, traditional desktop software applications, written text, virtual reality simulations for maintenance and repair, and Socratic dialogue, to name a few. You now realize that it would be best if there were a set of ITS authoring tools that could create tutors for all of these different interfaces, while catering to the individual differences of your colleagues and the needs of the particular knowledge domain. While that sounds like a significant challenge, you remember Don Norman's vision for the invisible computer (Norman, 1998), and his prescient early call for replacing a PC with many "information appliances" that would work for us while not burdening our minds or daily work. In effect, today's Internet cloud and mobile apps have begun to do that. App users seem happy to alternate the information representations with which they engage, when the representation feels natural for the chosen task. Therefore, you reason, the perfect ITS authoring tools would offer an invisible authoring system, one that allows your expert colleagues to, in effect, teach a computer what they know without getting in the way, just as naturally as a musician plays a composition into the iPad GarageBand app. Too bad those tools don't exist yet.

Introduction

A chapter in this volume by Blessing et al. offers a detailed comparison of the ITS authoring tools mentioned above. This chapter, in contrast, uses a user-experience lens to characterize the challenge of designing the ideally appropriate authoring tools to address the above scenario. Creating authoring tools that could meet this challenge would serve two purposes: (1) make it easier and faster to create ITSs much like the ones that exist today, and (2) create a framework that will lead to fundamentally better tutors in terms of pedagogy.

Definitions

To maintain clarity, we offer the following description of an intelligent tutor, its components, and the terms we are using. Shute (1994) notes that all ITSs contain the following: an expert model that contains the expert's knowledge; a student model that records what the student has learned (skills); and a pedagogical model that enables the tutor to react appropriately to the student's behavior. Other researchers note that the fourth critical component is the interface or problem-solving environment (Corbett, Koedinger & Anderson, 1997), which may be an off-the-shelf, third-party system or a

customized interface just for a particular tutor. To use standardized terminology proposed by Van Lehn (2006), the “task domain” is the discipline and content being taught, a “task” is a challenge assigned to a student or students, and tasks can be broken down into “steps.” Finally, ITSs vary in the generality of their expert models. Some are example-tracing tutors, focusing on providing feedback around one specific example task. When we say “tutor,” we intend the more general tutor, which contains knowledge that can be applied across many tasks of the same kind.

Extending the authoring task analysis described by Ritter, Blessing & Wheeler (2003), Table 1 contains the authoring tasks typically required to create a tutor. These steps are not strictly ordered; they typically are completed iteratively.

Table 2: Tasks of Authoring a Tutor

Characterize the Learning Environment	What are the possible actions or states of the learner’s environment that need to be noted by the tutor? What objects will the learner manipulate, and what actions are permitted?
Organize the Curriculum	What topics and skills needed to be learned? Are there subskills?
Characterize the Learning Activities	What are the steps the learner needs to take, generally speaking? How do those steps demonstrate the skills that need to be learned, i.e., what is the mapping between steps taken and skills?
Describe Good Tutoring	What does a right answer look like? What are frequent wrong answers? How do you evaluate a learner’s answer? What hints should be given if help is requested? What feedback should be given when the learner makes incorrect choices, and under what circumstances should it be given?

An Author’s User Interfaces

The ideal user interfaces (UIs) that the expert will use to accomplish the above authoring tasks will likely vary between tasks and may vary by individual author, if the author’s preferences for knowledge representation are known. Figure 8 illustrates example UIs that might be used.

The Wizard Dialog asks the expert a series of questions to refine the structure of the tutor, e.g., “Do your tasks have one right answer?” or “In your task domain, do students practice a task many times, 5–15 times, or fewer than 5 times?” The Wizard Dialog is essentially an expert system to narrow down to the most appropriate tutor structure.

The Decision Tree facilitates the creation of branching IF...THEN predicates. The idea of predicates and a predicate hierarchy is based on the original ACT-R inspired tutors, built using production rules (Anderson, Boyle, Corbett & Lewis, 1990), as well as on Carnegie Learning, Inc.’s SDK authoring tool (Blessing, Gilbert & Ritter, 2006; Blessing, Gilbert, Ourada & Ritter, 2009b).

The Click & Annotate UI works similarly to Camtasia or other software for making instructional screen-capture videos, in which the expert highlights elements of the interface and adds specific instructions. This UI would be particularly helpful for creating tutors on diagrams (e.g., free-body diagrams, blueprints, or geometry proofs) and with tutors on software applications.

The natural language scripting UI allows the expert to define a set of nouns (objects), adjectives (properties), and verbs (relationships), and then use natural language to describe conditions and interactions. In a game-based tutor, for example, an expert who is interested in guiding the student to stay close to the walls when exploring unknown corridors might write:

“Stay-close-to-wall” means the Player’s location **is near** a wall of the corridor. “**Near**” means closer than 10% of the width of the corridor.

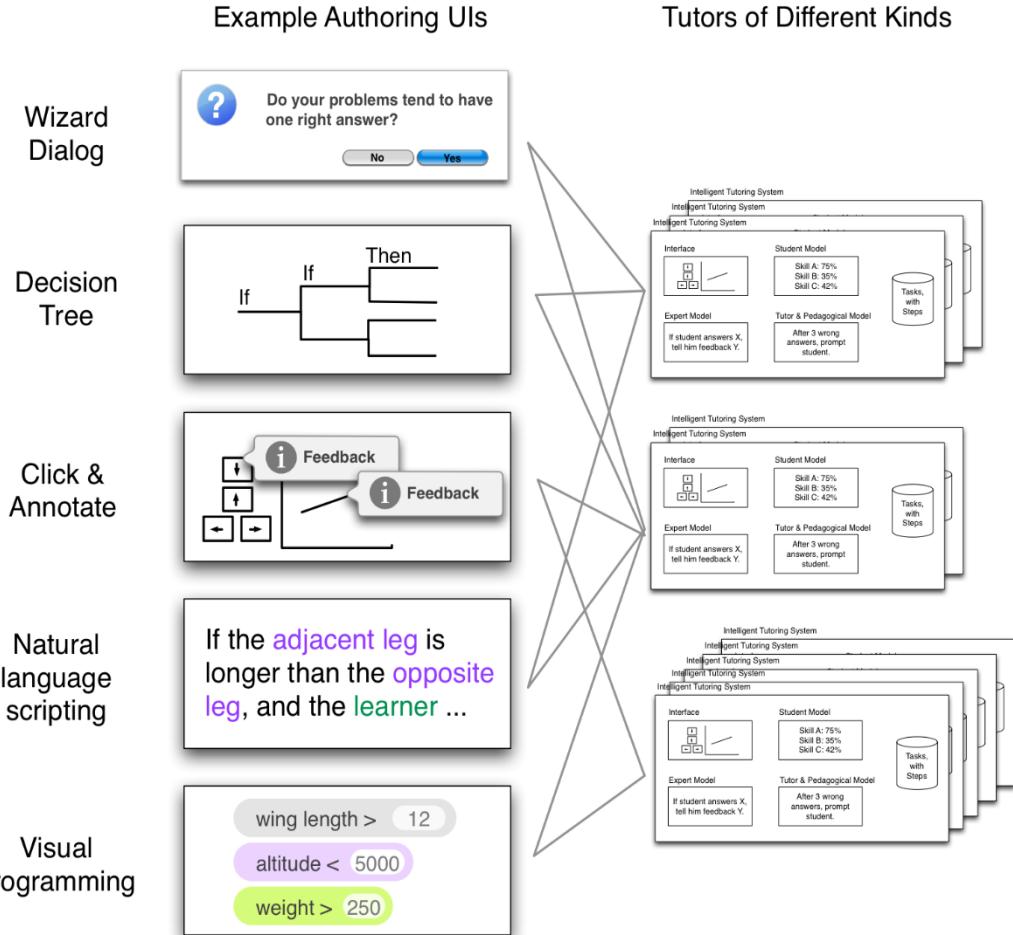


Figure 8: Different user interfaces are appropriate for different authoring tasks.

This approach uses principles of Applescript (2007) and Tutorscript (Blessing, et al., 2006), a language used with Cognitive Tutors at Carnegie Learning, Inc. A similar idea is proposed in the form of Natural-K (Jung & VanLehn, 2010).

The visual programming UI is similar to those used by Alice (Pausch, et al., 1995), Scratch (Resnick, et al., 2009), and Greenfoot (Henriksen & Kölking, 2004). The expert is given a set of primitives and operators and assembles them as blocks. This UI will be particularly useful for defining characteristics of the interface state, e.g., which components of a simulation are powered on, or the positions of entities within a serious game. One approach to tutoring based on game state is described in our previous work (Devasani, Gilbert, Shetty, Ramaswamy & Blessing, 2011; Gilbert, Devasani, Kodavali & Blessing, 2011).

This list of UIs is not exhaustive. Other UIs not depicted, for example, include a state transition graph, like the CTAT behavior graph (Aleven, et al., 2006), as well as a UI for constructing natural language parses and phrase classifiers, such as the Concept Grid (Blessing, Devasani & Gilbert, 2012; Devasani, Aist, Blessing & Gilbert, 2011). And of course plain source code or extensible markup language (XML)

is a possibility. Other new forms of UIs will become appropriate as new kinds for tutors arise, e.g., interfaces to allow conditions based on a student's affect or motivation.

A tutor would likely be created using a combination of these UIs. A *tutor template* would be a particular configuration of UIs designed to help create a tutor of a specific kind. As mentioned above, Norman's information appliances are the inspiration, so that the ideal ITS authoring system becomes an invisible collection of tools well designed for their purposes. Just as mobile phone users are familiar with switching between an email app, a calendar app, and a contacts app, which all share data, tutor templates will provide an optimal configuration of UIs for a given task domain, and that the backend will allow appropriate data sharing across them. In addition, a Template Recommender could be created, an expert system that will recommend templates to tutor authors based on a Wizard-style interview about the discipline and learning goals.

GIFTscript

While natural language scripting is mentioned above, it is worth considering it in more detail because of the power of allowing an expert to use her own language to create a tutor. Statements like the above "stay-close-to-wall" include the definition of new concept (stay-close-to-wall) and of a new condition ("near"). It assumes that the objects "corridor" and "wall" have been defined elsewhere and that a property "width" is associated with "corridor" (perhaps inherited from a parent object such as "object"). The approach suggests an object hierarchy with properties assigned to the objects that could be considered adjectives, e.g., a generic "building" object might have properties of "height," "location," and "dimensions" that could be inherited by child objects "house," "wall," "store," etc. Those child objects could in turn have specialized properties. This approach is taken by Carnegie Learning's tutor authoring tools (Blessing, et al., 2006), but user interfaces don't take advantage of natural language to author these hierarchies on the fly. And, working with inheritance hierarchies requires a degree of computational thinking (Wing, 2008) that our target users probably do not all have.

We suggest that if users could think of their knowledge domain in terms of scenarios, tasks, and concepts (or perhaps skills), an interactive language-based authoring tool might be able to be created, which creates the aforementioned object hierarchy more naturally. Since there is interest in developing the perfect authoring tools for GIFT, we call this language GIFTscript.

GIFTscript would be integrated with the other UIs described above by using text editing boxes that check GIFTscript syntax. Special visual indicators on the boxes would indicate to the user that (1) GIFTscript is available, (2) that syntax is violated, and (3) that valid GIFTscript is present. Also, these text boxes will support auto-completion and color coding, much like Visual Studio. A mockup of a sample interface is provided in Figure 2 as illustration. In this example, using some of the primitives already extant in GIFT, you could imagine script such as

`"Avoid Location {x}" means player location is far from {x} location.`

The interface would recognize that the user had defined a new condition called Avoid Location. It would also recognize noun phrases (objects) such as "player location" and "location," and if it didn't recognize those noun phrases, it would ask the user to define them. In this example, it recognizes a new adjectival phrase, "far from," and asks for a definition. The user might use some known objects to define it:

`"far from" means distance > than 20 ft.`

Then, using the new Avoid Location condition, the author could write rules such as

If the Player does not avoid location enemy bunker then remind, "Stay a safe distance from the enemy."



Figure 9: Mockup of interactive dialogue box for editing GIFTscript.

The critical feature of a UI featuring GIFTscript is offering a usable visualization of the objects and structures resulting from this approach.

Authoring Tool as Research Tool

Just as many creators of learning technologies are studying the misconceptions of their learners and getting usability feedback via educational data mining techniques and the use of embedded or stealth assessment (Shute & Ventura, 2013; Shute, Ventura, Bauer & Zapata-Rivera, 2009), it will be valuable if ITS authoring tools are themselves instrumented with click-stream logging similar embedded assessments. Using these methods, authors' own conceptual change can be monitored, especially if combined with a pre- and post-assessment of the author's understanding of the domain. Even without such assessments, however, an unsupervised learning classification method could be used to cluster different authors as to their approaches to knowledge representation, and perhaps, the interface elements could be personalized to each author's style. Also, it has been found useful in previous studies of authoring tools, e.g., Blessing, Gilbert, Ourada, and Ritter (2009a) to monitor the time spent by the author in different components of the authoring system. These usage profiles can be used to broadly characterize authors as experts or novices and perhaps give intelligent tutoring-style feedback to the authors themselves on the task of authoring.

Recommendations and Future Research

To create a mapping between types of tutors and appropriate UIs for authoring, it will be important to create a taxonomy of tutor types. The larger categories of tutor interfaces might be, for example, diagrams, equations, text, procedures, and cases. Each larger category might have subcategories, e.g., text-based tutors categorized by focus on short-answer responses, longer text passages, English language learning, reading, or Socratic dialogue. While other researchers have provided overviews of the gamut of intelligent tutors (Murray, 2003a; Sottilare, 2012; VanLehn , 2006), they have not focused on categorizing

tutors by the kind of authoring approach required, or by the knowledge representation required of an expert author. A taxonomy effort such as this might lead to a table as shown in Figure 3, which could then guide the creation of the ideal invisible authoring tools, for GIFT or other tutoring systems.

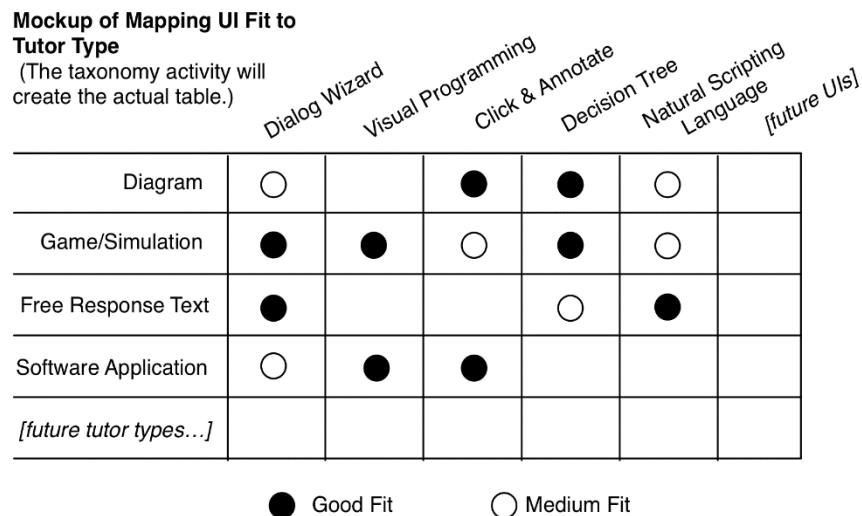


Figure 10: A mockup of the type of matrix that might emerge from mapping a taxonomy of tutor types to user interfaces for authoring them.

Research questions remain that are worth exploring. Are there noteworthy individual differences in UI preferences among experts in the same field? How does authoring with a given UI affect an expert's own understanding of his expertise? What effect does the choice of a tutor's authoring UI have on student learning? Does having natural UIs for authoring improve the quality of the resulting tutor?

References

- Apple (2007). Introduction to AppleScript Overview Retrieved July 6, 2012, from <http://developer.apple.com/applescript/>
- Aleven, V., Sewall, J., McLaren, B. M. & Koedinger, K. R. (2006). *Rapid Authoring of Intelligent Tutors for Real-World and Experimental Use*. Paper presented at the Sixth International Conference on Advanced Learning Technologies.
- Anderson, J. R., Boyle, C. F., Corbett, A. T. & Lewis, M. W. (1990). Cognitive modeling and intelligent tutoring. *Artificial intelligence*, 42(1), 7-49.
- Anderson, J. R., Conrad, F. G. & Corbett, A. T. (1989). Skill acquisition and the LISP tutor. *Cognitive Science*, 13, 467–505.
- Biswas, G., Leelawong, K., Schwartz, D., Vye, N. & The Teachable Agents Group at Vanderbilt. (2005). Learning by Teaching: A new agent paradigm for educational software. *Applied Artificial Intelligence*, 19(3-4), 363-392. doi: 10.1080/08839510590910200
- Blessing, S., Gilbert, S. B. & Ritter, S. (2006). *Developing an authoring system for cognitive models within commercial-quality ITSs*. Paper presented at the Nineteenth International FLAIRS Conference.
- Blessing, S. B., Devasani, S. & Gilbert, S. B. (2012). *Evaluating ConceptGrid: An Authoring System for Natural Language Responses*. Paper presented at the Twenty-Fifth International FLAIRS Conference, Marco Island, FL, USA.
- Blessing, S. B., Gilbert, S. B., Ourada, S. & Ritter, S. (2009a). Authoring model-tracing cognitive tutors. *International Journal for Artificial Intelligence in Education*, 19(2).

- Blessing, S. B., Gilbert, S. B., Ourada, S. & Ritter, S. (2009b). Authoring model-tracing cognitive tutors. *International Journal for Artificial Intelligence in Education*.
- Corbett, A. T., Koedinger, K. R. & Anderson, J. R. (1997). Chapter 37 - Intelligent Tutoring Systems. In G. H. Marting, K. L. Thomas & V. P. Prasad (Eds.), *Handbook of Human-Computer Interaction (Second Edition)* (pp. 849-874). Amsterdam: North-Holland.
- Crouch, C. H. & Mazur, E. (2001). Peer Instruction: Ten years of experience and results. *American Journal of Physics*, 69(9), 970-977.
- Devasani, S., Aist, G., Blessing, S. & Gilbert, S. B. (2011). *Lattice-Based Approach to Building Templates for Natural Language Understanding in Intelligent Tutoring Systems*. Paper presented at the Proceedings of the Fifteenth Conference on Artificial Intelligence in Education, Auckland.
- Devasani, S., Gilbert, S. B. & Blessing, S. B. (2012). *Evaluation of Two Intelligent Tutoring System Authoring Tool Paradigms: Graphical User Interface-Based and Text-Based*. Paper presented at the Twenty-First Conference on Behavior Representation in Modeling and Simulation (BRIMS), Amelia Island, FL, USA.
- Devasani, S., Gilbert, S. B., Shetty, S., Ramaswamy, N. & Blessing, S. (2011). *Authoring Intelligent Tutoring Systems for 3D Game Environments*. Paper presented at the Proceedings of the Authoring Simulation and Game-based Intelligent Tutoring Workshop at the Fifteenth Conference on Artificial Intelligence in Education, Auckland.
- Gilbert, S. B., Blessing, S. B. & Kodavali, S. (2009). *The Extensible Problem-Specific Tutor (xPST): Evaluation of an API for Tutoring on Existing Interfaces*. Paper presented at the 14th International Conference on Artificial Intelligence in Education.
- Gilbert, S. B., Devasani, S., Kodavali, S. & Blessing, S. (2011). *Easy Authoring of Intelligent Tutoring Systems for Synthetic Environments*. Paper presented at the Twentieth Conference on Behavior Representation in Modeling and Simulation (BRIMS), Sundance, UT, USA.
- Henriksen, P. & Kölling, M. (2004). *Greenfoot: Combining object visualisation with interaction*. Paper presented at the Companion to the 19th annual ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications.
- Institute of Medicine, National Academy of Sciences & National Academy of Engineering. (2007). *Rising above the gathering storm: Energizing and employing America for a brighter economic future*. Washington, D.C.: National Academies Press.
- Jung, S.-Y. & VanLehn, K. (2010). *Developing an intelligent tutoring system using natural language for knowledge representation*. Paper presented at the Intelligent Tutoring Systems Conference (ITS 2010).
- Koedinger, K. R., Anderson, J.R., Hadley, W.H. & Mark, M.A. (1997). Intelligent tutoring goes to school in the big city. *International Journal for Artificial Intelligence in Education*, 8, 30-43.
- Kolodner, J. L., Camp, P. J., Crismond, D., Fasse, B., Gray, J., Holbrook, J., et al. (2003). Problem-based learning meets case-based reasoning in the middle-school science classroom: Putting learning by design (tm) into practice. *The journal of the learning sciences*, 12(4), 495-547.
- Lesgold, A., Lajoie, S., Bunzo, M. & Eggan, G. (1992). SHERLOCK: A coached practice environment for an electronics troubleshooting job. In J. Larkin & R. Chabay (Eds.), *Computer-assisted instruction and intelligent tutoring systems: Shared goals and complementary approaches*. Hillsdale, NJ: Lawrence Erlbaum Associates.
- Matsuda, N., Cohen, W. W., Sewall, J., Lacerda, G. & Koedinger, K. R. (2007). Predicting students' performance with simstudent: Learning cognitive skills from observation. *FRONTIERS IN ARTIFICIAL INTELLIGENCE AND APPLICATIONS*, 158, 467.
- Mitrovic, A., Martin, B., Suraweera, P., Zakharov, K., Milik, N., Holland, J., et al. (2009). ASPIRE: an authoring system and deployment environment for constraint-based tutors. *International Journal of Artificial Intelligence in Education*, 19(2), 155-188.
- Murray, T. (2003a). An Overview of Intelligent Tutoring System Authoring Tools: Updated analysis of the state of the art *Authoring tools for advanced technology learning environments* (pp. 491-544): Springer.
- Murray, T. (2003b). Principles for pedagogy-oriented knowledge based tutor authoring systems: Lessons learned and a design meta-model *Authoring Tools for Advanced Technology Learning Environments* (pp. 439-466): Springer.
- Murray, T., Blessing, S. & Ainsworth, S. (Eds.). (2003). *Authoring Tools for Advanced Technology Learning Environments: Toward Cost-effective Adaptive, Interactive, and Intelligent Educational Software*. Norwell, MA: Kluwer Academic Publishers.
- Norman, D. (1988). *The design of everyday things*. New York: Basic Books.

- Norman, D. (1998). *The Invisible Computer: Why Good Products Can Fail, the Personal Computer Is So Complex, and Information Appliances Are the Solution*. Cambridge, MA: MIT Press.
- Pausch, R., Burnette, T., Capeheart, A., Conway, M., Cosgrove, D., DeLine, R., et al. (1995). Alice: Rapid prototyping system for virtual reality. *IEEE Computer Graphics and Applications*, 15(3), 8-11.
- Razzaq, L., Patvarczki, J., Almeida, S. F., Vartak, M., Feng, M., Heffernan, N. T., et al. (2009). The Assitment Builder: Supporting the life cycle of tutoring system content creation. *Learning Technologies, IEEE Transactions on*, 2(2), 157-166.
- Resnick, M., Maloney, J., Monroy-Hernandez, A., Rusk, N., Eastmond, E., Brennan, K., et al. (2009). Scratch: programming for all. *Commun. ACM*, 52(11), 60-67. doi: <http://doi.acm.org/10.1145/1592761.1592779>
- Ritter, S., Blessing, S. B. & Wheeler, L. (2003). Authoring tools for component-based learning environments. In T. Murray, S. Blessing & S. Ainsworth (Eds.), *Authoring Tools for Advanced Technology Learning Environments* (pp. 467-489). Norwell, MA: Kluwer Academic Publishers.
- Ritter, S., Kulikowich, J., Lei, P., McGuire, C. L. & Morgan, P. (2007). What evidence matters? A randomized field trial of Cognitive Tutor Algebra I. In T. Hirashima, U. Hoppe & S. S. Young (Eds.), *Supporting Learning Flow through Integrative Technologies* (Vol. 162, pp. 13-20). Amsterdam: IOS Press.
- Schön, D. A. (1983). *The reflective practitioner: How professionals think in action*. New York: Basic books.
- Shute, V. & Ventura, M. (2013). *Stealth assessment: Measuring and supporting learning in video games*: MIT Press.
- Shute, V. J. & Psotka, J. (1994). Intelligent tutoring systems: Past, present, future. *Technical Report AL/HR-TP-1994-0005, USAF, Armstrong Laboratory*.
- Shute, V. J., Ventura, M., Bauer, M. & Zapata-Rivera, D. (2009). Melding the power of serious games and embedded assessment to monitor and foster learning. *Serious games: Mechanisms and effects*, 295-321.
- Sottilare, R. (2012). *Considerations in the Development of an Ontology for a Generalized Intelligent Framework for Tutoring* Paper presented at the International Defense and Homeland Security Simulation Workshop 2012, Vienna, Austria.
- VanLehn, K. (2006). The behavior of tutoring systems. *International journal of artificial intelligence in education*, 16(3), 227-265.
- VanLehn, K., Lynch, C., Schulze, K., Shapiro, J. A., Shelby, R., Taylor, L., et al. (2005). *The Andes Physics Tutoring System: Five Years of Evaluations*. Paper presented at the Proceedings of the 2005 conference on Artificial Intelligence in Education: Supporting Learning through Intelligent and Socially Informed Technology.
- Vattam, S. S. & Kolodner, J. L. (2011). On foundations of technological support for addressing challenges facing design-based science learning. *Technology Enhanced Learning and Cognition*, 27, 233.
- Wing, J. M. (2008). Computational Thinking and Thinking about Computing. *Philosophical Transactions of the Royal Society*, 366, 3717-3725.

Chapter 25 Lowering the Technical Skill Requirements for Building Intelligent Tutors: A Review of Authoring Tools

H. Chad Lane¹, Mark G. Core², Benjamin S. Goldberg³

¹University of Illinois, Urbana-Champaign, ²University of Southern California, ³US Army Research Laboratory

Introduction

Educational technologies have come to play an important role in advancing the science of learning. By consistently applying a set of pedagogical policies (and not getting tired while doing so), educational technologies can be used to address precise questions about how people learn and how to best help them. The resulting findings often answer important questions about human learning, which, in turn, can positively influence the design of future educational technologies or even possibly educational practices. A second way learning science researchers seek to have impact is by getting the technology in the hands of as many learners as possible. Unfortunately, with more users come more requirements, and therefore, additional questions educational software designers need to address. For example, can a system be tailored to the specific needs of a class, teacher, or individual learner? Can it be used in a new task domain? Is it possible to reorganize or create new course content? Can the pedagogical approach and/or content embedded in the system be adjusted or even replaced? Sadly, but understandably, software that is created for lab studies or specific end-user needs do not often address these questions. If the aim is to “go big,” then it is no longer feasible to create one system suited for all needs tools for configuring and creating content are a requirement.

In this chapter, we focus on intelligent tutoring systems (ITSs), an instance of educational technology that is often criticized for not reaching its full potential (Nye, 2013). Researchers have debated why, given such strong empirical evidence in their favor (Anderson, Corbett, Koedinger & Pelletier, 1995; D'Mello & Graesser, 2012; VanLehn et al., 2005; Woolf, 2009), intelligent tutors are not in every classroom, on every device, providing educators with fine-grained assessment information about their students. Although many factors contribute to a lack of adoption (Nye, 2014), one widely agreed upon reason behind slow adoption and poor scalability of ITSs is that the engineering demands are simply too great. This is no surprise given that the effectiveness of ITSs is often attributable to the use of rich knowledge representations and cognitively plausible models of domain knowledge (Mark & Greer, 1995; Valerie J. Shute & Psotka, 1996; VanLehn, 2006; Woolf, 2009), which are inherently burdensome to build. To put it another way: the features that tend to make ITSs effective are also the hardest to build. The heavy reliance on cognitive scientists and artificial intelligence (AI) software engineers seems to be a bottleneck.

This issue has led to decades of research geared toward reducing both the skills and time to build intelligent tutors. The resulting ITS *authoring tools* serve different educational goals, but generally seek to enable creating, editing, revising, and configuring the content and interfaces of ITSs (Murray, Blessing & Ainsworth, 2003). A significant challenge lies in the accurate capture of the domain and pedagogical expertise required by an ITS, and many authoring tools focus on eliciting this knowledge. Unfortunately, as ITS technology has evolved, the authoring burden has increased rather than decreased, and so the tension between usability of an authoring tool and the sophistication of the target ITS knowledge representation is substantial.

In this chapter, we focus on the problem of *reducing* the technical knowledge required to build ITSs (only one of many possible goals for authoring tools). We review the important historical attempts that most directly sought to open up the creation of ITSs to nonprogrammers (like educators) as well as more recent work that addresses the same goal. We review popular approaches, such as programming by

demonstration, visualization tools, and what you see is what you get (WYSIWYG) authoring, and summarize the limited experimental evidence validating these approaches. The central questions driving this review are (1) In what ways have researchers sought to make authoring more intuitive and accessible to nonprogrammers? (2) For what purposes have these tools been developed? (3) What components of an ITS have they addressed? and (4) How have researchers evaluated these approaches and what have they learned about intuitive authoring? The chapter ends with suggestions for future research, including identifying ways to empirically understand the sophistication vs. ease-of-authoring tradeoff, leveraging more findings from the human-computer interaction (HCI) community, and addressing the glaring gap in authoring research as it relates to learning management systems (LMSs).

The Problem

What makes ITSs so difficult to create? ITSs provide the fine-grained support necessary for deep learning unlike most traditional computer-aided instructional systems (VanLehn, 2011), but this ability requires greater complexity (VanLehn, 2006). We use the Generalized Intelligent Framework for Tutoring (GIFT) architecture, shown in Figure 1, as a general model of the complex system of interconnected components typically present in an ITS, and the end product of ITS authoring. The specific requirements, roles, and complexities of each component are described elsewhere (Sottilare, 2012), but some of the more prominent components in terms of authoring include the following:

- The **learner module** tracks the learner's state over the course of a session. The learner module updates performance measures based upon assessments of learner activities, and may estimate changes in understanding, acquisition of skills and learner emotions.
- The **pedagogical module** makes the instructional decisions that drive the tutor's behavior. These decisions can vary in scope from topic and problem selection to deciding whether to give feedback and the content and style of this feedback.
- The **domain module** handles domain-specific responsibilities such as assessing learner problem solving and providing help. This module may use general information about the target task and any associated simulation as well as rely upon problem-specific data.
- The **tutor-user interface** provides the communication channel(s) between learner and tutor, such as speech, text, visualizations, etc. It defines the scope of potential tutoring interactions.

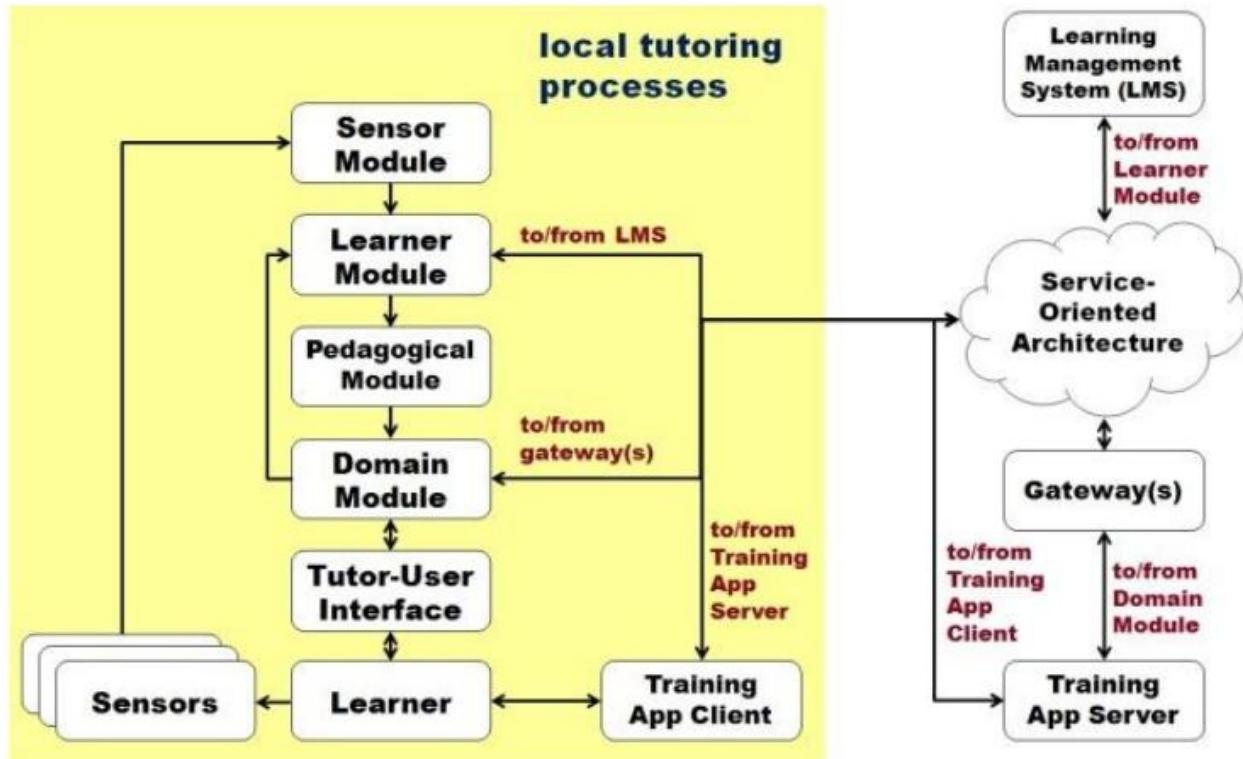


Figure 1. Overview of the GIFT architecture (Sottilare, 2012).

It is typical for authoring tools to focus on specific components that are most important or relevant for the target ITS. This approach also makes the process more viable as authors focus their attention on a few components while the rest may remain unchanged from problem to problem, or domain to domain. By starting with the GIFT framework, which seeks to maximize generality, rather than a specific system, the full range of possible targets for an ITS authoring tool is more apparent.

ITS Authoring Tools: Goals And Tradeoffs

In two broad and thorough reviews of the field (Murray, 1999, 2003), two (of the many) take-away messages are that authoring tools (1) have been developed with a wide variety of goals in mind and for many different categories of users, and (2) present a huge space of tradeoffs both in terms of their own implementation and those that must be addressed by authors using the system. Authoring success stories, such as Cognitive Tutor Authoring Tools (CTAT) (Aleven, McLaren, Sewall & Koedinger, 2006) and REDEEM (Ainsworth et al., 2003), always impose a reasonable level of constraints on their authors and make assumptions so that tasks can stay manageable.

Murray (2003) identifies five typical goals for authoring tools, roughly in order of importance or predominance (p. 509):

- (1) Decrease the effort required to build an ITS (e.g., time, cost).
- (2) Decrease the “skill threshold” for building ITSs.

- (3) Support authors in the articulation of domain or pedagogical knowledge.
- (4) Enable rapid prototyping of ITSs.
- (5) Rapidly modify and/or prototype with the aim of evaluating different approaches.

Systems in category (1) can include those built for cognitive scientists and programmers. For example, CTAT (Aleven, et al., 2006) includes two primary methods for tutor creation, the first of which involves creating and debugging production rule-based cognitive models directly. CTAT includes a variety of tools for organizing, testing, watching, and editing cognitive models, all designed for users with high levels of technical skill. The second type of authoring uses *example-tracing*, which is described below and more directly addresses Murray's second goal.

Category (1) stands in direct contrast to (2), which involves lowering the bar on what authors need to know or be able to do. Typically, authoring tools that seek to do this have the aim of allowing teachers, subject-matter experts, and other educators to create ITSs to address their own needs. Category (2) is the focus of this chapter — how have researchers attempted to “simplify,” or at least remove some of the more technically onerous aspects of building ITSs? Goals in categories (3) through (5) are in many ways orthogonal to (1) and (2). Articulating domain and pedagogical knowledge (3) is a requirement for ITSs and can be accomplished with no specialized tools, tools for technically skilled authors, or tools for nonprogrammers. Similarly, the rapid creation of ITSs and variations on existing ITSs for the purposes of testing or running experiments can also be accomplished regardless of the tools used.

Of course, whatever purposes an authoring tool is intended to serve will directly impact the design. In turn, the design of any content-creation tool (e.g., word processors, presentation software) involves tradeoffs. In the case of ITS authoring tools, a variety of tradeoffs are apparent. One tension is that complexity in one area of authoring can introduce difficulties in other areas. For example, a tool could allow authors to create a custom graphical user interface (GUI) for their ITS. However, the underlying ITS has no idea what the controls (e.g., text boxes and menus) of the GUI mean. Thus, either the author has to develop a knowledge representation and link it to the GUI, or develop a model in terms of GUI actions (e.g., example-tracing in CTAT). A second tension that arises is between the complexity of an authoring tool and ease of use. For example, the full power of cognitive modeling is available in CTAT, but requires programming skills and the resulting cognitive models can be onerous. In general, the greater the expressive power provided by an authoring system, for any module, the more complicated other components become to build. Therefore, we find a distinct tradeoff between this expressive power and the ease at which authoring can be accomplished.

In the remainder of this chapter, we focus our attention on techniques researchers have used to reduce the skills needed in order to build ITSs (category (2) from the list above). A tradeoff made in most of these examples is increased authorability but reduced sophistication of the underlying tutors that are built. In other words, the assumptions made and steps taken to “simplify” the authoring process have generally led to simpler models. This is not necessarily true in all of the cases below, however. SimStudent (Matsuda et al., 2013) and Authoring Software Platform for Intelligent Resources in Education (ASPIRE) (Mitrovic et al., 2009), for example, use machine learning techniques to infer more than what the authoring activities provide on the surface. Another common tradeoff is simplifying the process by limiting what the author can create or change. For example, REDEEM (Ainsworth, et al., 2003) uses a “courseware catalogue” as a starting point, and SitPed (Lane, Core, et al., in-press) starts with pre-constructed scenarios.

Approaches to Building Intuitive Authoring Tools

The AI-heavy components of an ITS (domain module, pedagogical module, learner module) generally require detailed information from authors. This type of ITS authoring is similar to the task of *knowledge engineering*, which was widely recognized as onerous and a possible impediment to the growth of expert systems. The general challenge of capturing or eliciting knowledge from subject matter experts was recognized early, leading to a great deal of research focused on the knowledge elicitation problem (Hoffman, Shadbolt, Burton & Klein, 1995). ITSs share this problem, but with the additional burden of needing to address issues related to pedagogy – that is, how to present information, assess learners' knowledge, deliver feedback, and so on. Thankfully, ITSs often do *not* require such heavy-duty models as fully elaborated expert systems, and solutions that go beyond basic computer-aided instruction but not as far as a fully-fledged ITS still have a great deal of value. For example, the ASSISTments approach provides teachers with authoring tools to create, share, and deploy step-by-step example problems that approximate ITS behaviors by providing step-level support for a learner without the need for traditional student modeling, expert modeling, and so on (Heffernan & Heffernan, 2014).⁷

Although many authoring tools indirectly lower the skill threshold needed to build a tutor, either through hiding implementation details or automating steps, the systems included in this review are systems that prioritize usability. A requirement for inclusion is that a system be explicitly built for and tested with nonprogrammers who are either instructors or subject-matter experts in an educational role. A second requirement is that the authoring tool supports the population of ITS-like components (see Figure 1) via interaction with the author. The end result of the authoring process is a learning system that performs at least some of the behaviors from the standard definition of a tutoring system (VanLehn, 2006).

A final dimension we highlight is Murray's (2003) distinction between *performance* and *presentation* roles that an ITS can play. An ITS that focuses on performance typically assesses the learner during problem solving (or other form of practicing a skill) and provides feedback and scaffolding. This is perhaps the most common role of an ITS given that they often help learners during homework. An authoring tool that addresses the presentation of content is geared toward a more direct instructional role, such as that played by educational videos in massive open online courses (MOOCs) or flipped classrooms. The presentation of content can be made highly adaptive based on learner behaviors and embedded assessments, and so there are many opportunities to go beyond what simple videos or reading can achieve. Historically, many ITSs sought to play both roles and use shared models between the two to increase the level of personalization (Brusilovsky, 2012). The distinction between authoring for performance or presentation (or both) is used in the discussion that follows.

Authoring for Content Delivery

REDEEM (Ainsworth, et al., 2003) represents one of earliest and most successful attempts to put authoring tools in the hands of teachers. Using intuitive interfaces, REDEEM walks authors through a workflow that operates primarily on existing content so that a generated ITS can later present it adaptively to learners. In addition, authors can increase interactivity in the resulting lessons by creating questions and identifying “reflection points” that allow the system to know when students should spend more time processing the material. REDEEM most directly supports non-technical authors in the following ways:

1. REDEEM provides a well-defined *workflow* with integrated stages that are all clearly defined for authors so that they understand the overall process and end result.

⁷ Although ASSISTments does include elaborate records of student performance used in service of helping instructors assess their students' learning.

2. It adopts a slider-based approach to allow authors to specify parameters such as suitability of resources for learners, and amount of student choice.
3. Instructional strategies are expressed in ways that are familiar to authors (who are instructors).

The REDEEM workflow takes authors through three distinct phases: (1) describe the course material by organizing and marking content with structural annotations, (2) define the kinds of learners who will use the resulting ITS, and (3) describe how the system should teach those learners with the content articulated during step 1. REDEEM assumes the availability of a “courseware catalogue,” which consists primarily of reading content. For this content, authors are asked to identify sections, label the content in sections along various dimensions (e.g., how difficult or familiar it will be to students), and finally describe relations between sections (e.g., section B is an application of the concept described in section A). These relationships help the system build a semantic representation of the course content, which is then used by the resulting ITS to adapt instruction. In addition, authors also create interactive content such as multiple choice and fill-in-the-blank questions. The second step for REDEEM authors is to create a base of learner types (based on the author’s discretion), while the third is the “glue” that brings these steps together.

To complete their ITS, authors must define a body of tutoring *strategies* that ultimately tell a REDEEM ITS how to use the annotated content. A slider metaphor is used to configure the details of the tutoring strategies. For example, if a teacher wants to allow the ITS to engage in practice with the learner, they can define this strategy by using a series of sliders to set the parameters appropriately (Figure 2). The strategy includes information about when to use it, how to deploy it, and how to provide help. Although there is a cap of 20,000 different strategies, authors tend to create about 7 per tutor (Ainsworth, et al., 2003, p. 213).

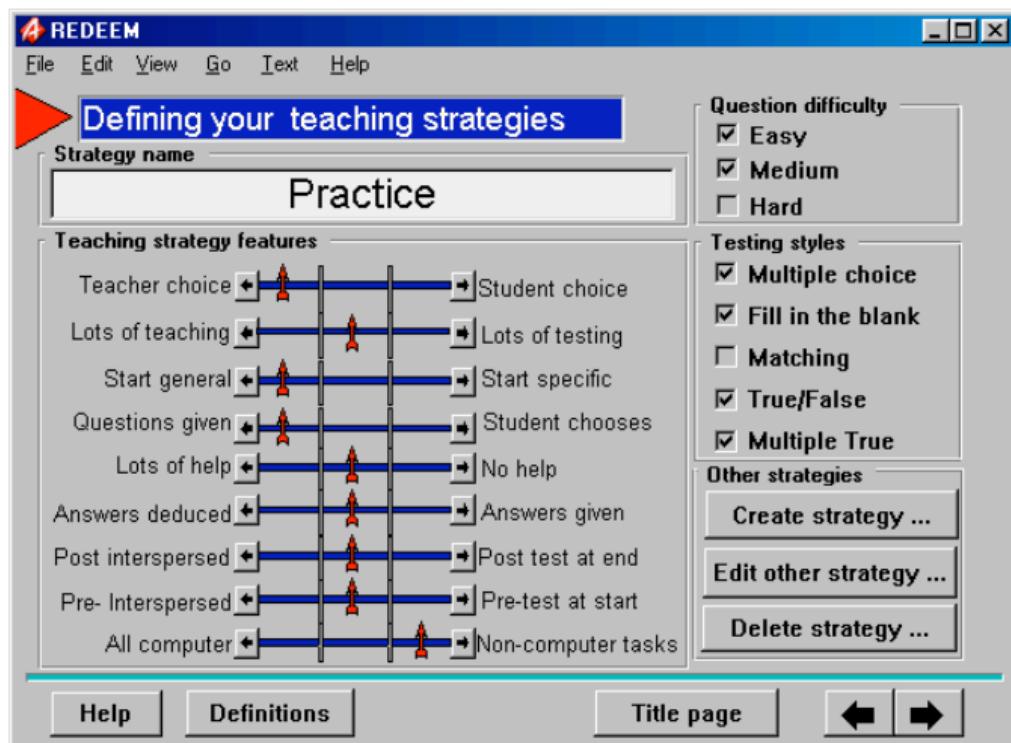


Figure 2. Creating a teaching strategy in REDEEM (Ainsworth, et al., 2003).

REDEEM has undergone multiple evaluations showing that teachers can use the system to create tutors, and that they find it easy to use. In addition, tutors created with REDEEM have been compared against computer-aided instruction (CAI) counterparts. These studies have demonstrated a significant difference in learning outcomes with effect sizes as large as 0.76 for REDEEM vs. 0.36 for CAI systems with the same content (Ainsworth & Fleming, 2006).

Generalizing from Examples

One important skill instructors and subject matter experts share is that they can solve problems in the domain in which they teach or work. This observation was made in ITS authoring as well as in the areas of expert systems and programming (Cypher & Halbert, 1993). Authoring by demonstration seeks to leverage this observation by allowing instructors and experts to simply *show* the authoring system how students should solve problems instead of forcing instructors and experts to explicitly encode domain knowledge. Given a sufficient number of examples, the authoring tool will develop a generalized model of the skill that can be used in an ITS.

Building on the work in expert systems and programming-by-demonstration, Blessing (2003) was one of the pioneers in applying this idea to building an ITS. His Demonstr8 authoring system could generate an ACT-style tutor for arithmetic based on authors solving problems. ACT Tutors have expert models consisting of production rules capable of solving problems, and learner actions are continually compared against this model through a process called model tracing. Demonstr8 creates such an expert model by attempting to determine the rationale behind steps in solved examples and create rules that apply across examples. However, an author must first create an underlying knowledge representation such that Demonstr8 can generalize from the author's behavior. For example, authors need to define the concept of a column of numbers as well as basic arithmetic operations such as subtracting two digits. It is also the case that the underlying model cannot be hidden from the author who may need to adjust production rules as well as specify details such as goals and subgoals.

It is worth noting a similar approach adopted in DISCIPLE (Tecuci & Keeling, 1998). Here, the domain is history and the task is to determine whether a source is relevant to a specific research assignment, and why or why not. Only the GUI and the specialized problem solver are specific to the target domain/task. To define the target task, an educator with help from a knowledge engineer first builds an ontology. The next step is developing a set of problem-solving rules; in this case, the rules determine whether a source is relevant based on properties of the source and the research assignment. The general approach is for the educator to teach the authoring system through demonstration (i.e., providing a correct answer), limited explanation (e.g., pointing out a relevant feature), and feedback. In the case of feedback, the system generates new examples and the educator helps debug the rules when incorrect results appear. The educator must then extend the set of natural language generation templates allowing the system to generate tutoring guidance from the underlying knowledge representation. The resulting history ITS was highly rated in surveys from an experiment with students and teachers, and the domain module was judged highly accurate by an external expert.

ASPIRE (Mitrovic, et al., 2009) is an authoring tool developed at the University of Canterbury for the construction of tutors that use constraint-based modeling as the primary method for knowledge representation. Constraints are fundamentally different from production rules in that they encode properties that must hold in solutions rather than generate problem solving steps. Constraints, when violated by a student's solution, capture teachable moments in which constraint-based tutors can help. Authors are required to create constraints, feedback messages, problems, and if necessary, a user interface. Although different from production rules, constraints are still a form of knowledge representation and building a constraint base for an ITS requires a certain level of technical skill.

ASPIRE's predecessor, Web-Enabled Tutor Authoring System (WETAS) (Mitrovic, Martin & Suraweera, 2007), supported users with some technical expertise while ASPIRE falls into the category of systems built for nonprogrammers seeking to automate some of the more complex tasks of building ITSs with machine learning.

To build a constraint-based tutor with ASPIRE, authors must perform three steps, none of which require programming skills: (1) design a (text-based) interface, (2) build a domain ontology (Figure 3), (3) create problems and solutions in the interface. ASPIRE can use the ontology to automatically generate syntactic constraints corresponding to domain requirements. For example, an instructor might require students to specify a lowest common denominator (e.g., the student must type a number into the relevant location in the GUI). Semantic constraints model the correctness of the answer. Authors are required to specify alternate solutions, and ASPIRE automatically generates semantic constraints accommodating these variations (Mitrovic, et al., 2009). Perhaps the most burdensome step in creating an ASPIRE tutor lies in the creation of a domain ontology, which has the potential to become overly complex. However, as can be seen in Figure 3, the process is simplified in that only basic hierarchical relationships are needed (such as specialization).

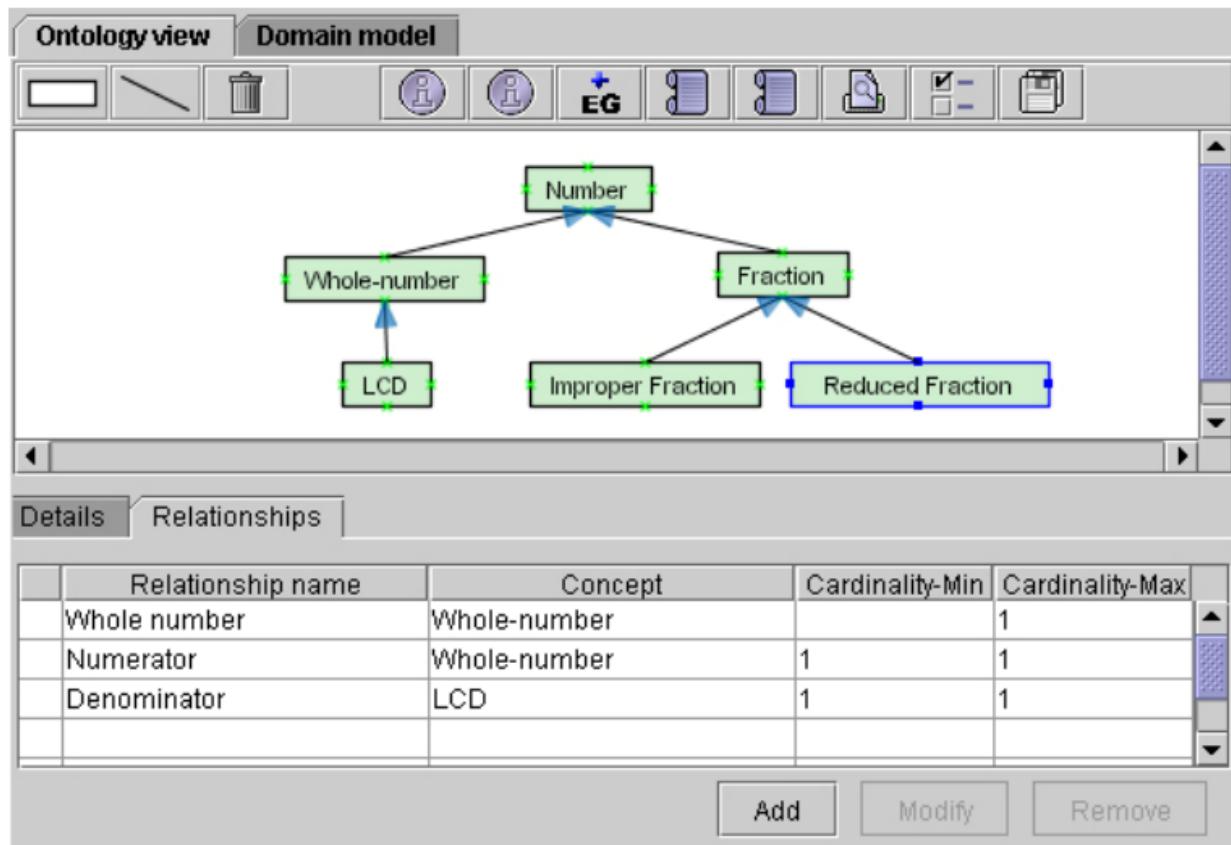


Figure 3. The ASPIRE ontology editor, used in the automated generation of constraints
(Mitrovic, et al., 2009)

When compared to a hand-authored domain model, ASPIRE was found to generate all of the same syntactic constraints and covered 85% of the semantic. A larger pilot evaluation of a tutoring system generated by ASPIRE with a subject-matter expert author (and nonprogrammer), showed that it produced significant learning gains and that learners followed expected learning curves (Mitrovic et al., 2008).

Authoring by Constructing Elaborated Examples

One of the most prominent efforts to provide authoring tools for non-experts is CTAT at Carnegie Mellon University. CTAT provides tools for building two kinds of tutoring systems: *example-tracing* tutors, which involve problem-specific authoring but no programming (Aleven, McLaren, Sewall & Koedinger, 2009), and *cognitive tutors*, which require AI programming and the development of a problem-independent cognitive model of the target skill(s) (Anderson, et al., 1995). The work on Cognitive Tutors shows there is room for improvement in authoring tools for ITS programmers. For Cognitive Tutors, CTAT has been shown to reduce development time by a factor of 1.4 to 2 (Aleven, et al., 2006).

In keeping with the goals of our chapter, we focus on non-programmers and example-tracing tutors. For example-tracing tutors, early evaluations of efficiency gains using CTAT are also impressive: a reduction in development costs by a factor of 4 to 8 over traditional estimates on ITS development (Aleven, et al., 2009). Example-tracing tutors are created by demonstration. The appeal, therefore, is that an author can create solution models by simply solving problems in ways that learners might. This is accomplished in CTAT by defining a specific problem, solving it, and then expanding the resulting solution in ways so that other common problem solving actions can be recognized, such as alternate solutions and common misconceptions. These different problem-solving steps form a *behavior graph* such as the one on the right side of Figure 4. Learners take actions by manipulating a GUI (left side of Figure 4) and the tutoring system matches these actions to nodes in the behavior graph. The graph branches as authors specify a variety of correct and incorrect approaches to solving the problem.

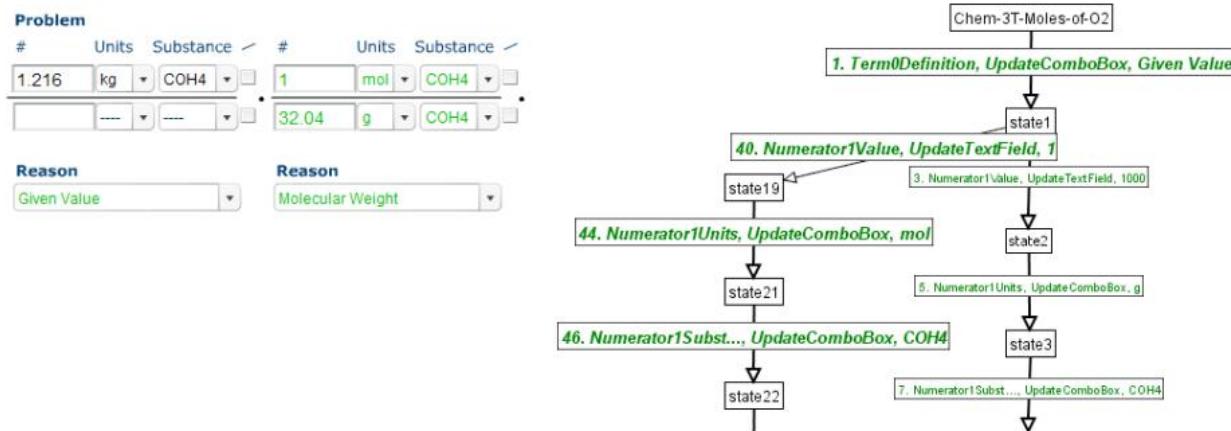


Figure 4. An authored example in CTAT for stoichiometry (Aleven, et al., 2009).

Example-tracing does not require a machine-readable ontology defining the domain and there is no machine learning that must be debugged by the author. However, this knowledge-light approach means that authors must annotate steps in the behavior graph with hint and feedback messages as well as links to the learner module (e.g., taking this step provides evidence that the learner understands a certain domain concept). This annotated behavior graph contains all the information needed for the ITS to assess learner actions and provide step-level feedback.

The Situated Pedagogical Authoring (SitPed) project at the University of Southern California (Lane, Core, et al., in-press) focuses on problem-solving through conversation (e.g., a therapist talking to a client) using simulated role players. Using SitPed, authors create an ITS by doing the following:

- Specifying paths through the problem space by simultaneously *solving* problems (either correctly or intentionally incorrectly) and indicating the relevant skills and misconceptions.
- Pausing during problem solving to create hints and feedback messages associated with the current situation.

Like the case of example-tracing tutors, authors work in the same learning environment that learners use. Thus, SitPed falls roughly into the category of WYSIWYG authoring tools (Murray, 2003) because authors are constantly reminded of what the resulting learning experience will be like. In the case of SitPed, demonstration is not simply a technique to hide technical details. Simulated conversations and simulations in general allow learners to explore a wide space of possibilities, and it can be difficult for authors to visualize the learner's perspective unless they are also working through examples in the same simulation.

One of the difficulties in building an ITS for a simulated role play is that simulated role players can be implemented in a variety of ways. The initial version of SitPed targets branching conversations. At each step in the conversation, learners are selecting utterances from a menu and the virtual role player consults a tree to lookup its response and the next set of menu items. This tree simply contains the lines of the conversation as well as the associated animations corresponding to performance of the role player lines. In branching conversations, it is necessary for the author to play through all branches of the tree and link each possible learner choice to the skills and misconceptions of the domain. This process is illustrated in Figure 5. Although the goal is to recreate the learner experience as much as possible, authors need to be able to see relevant context (e.g., the dialogue history in the middle) and make annotations corresponding to the skills and common mistakes of the domain which we refer to as tasks (via the menu labeled "Task List"). This exhaustive exploration of the possibilities is necessary because of the difficulty of automatically understanding the dialogue well enough to identify skills such as active listening. For simulated role players with models of dialogue and emotion, more scope for generalization may be possible based on expert conversations.

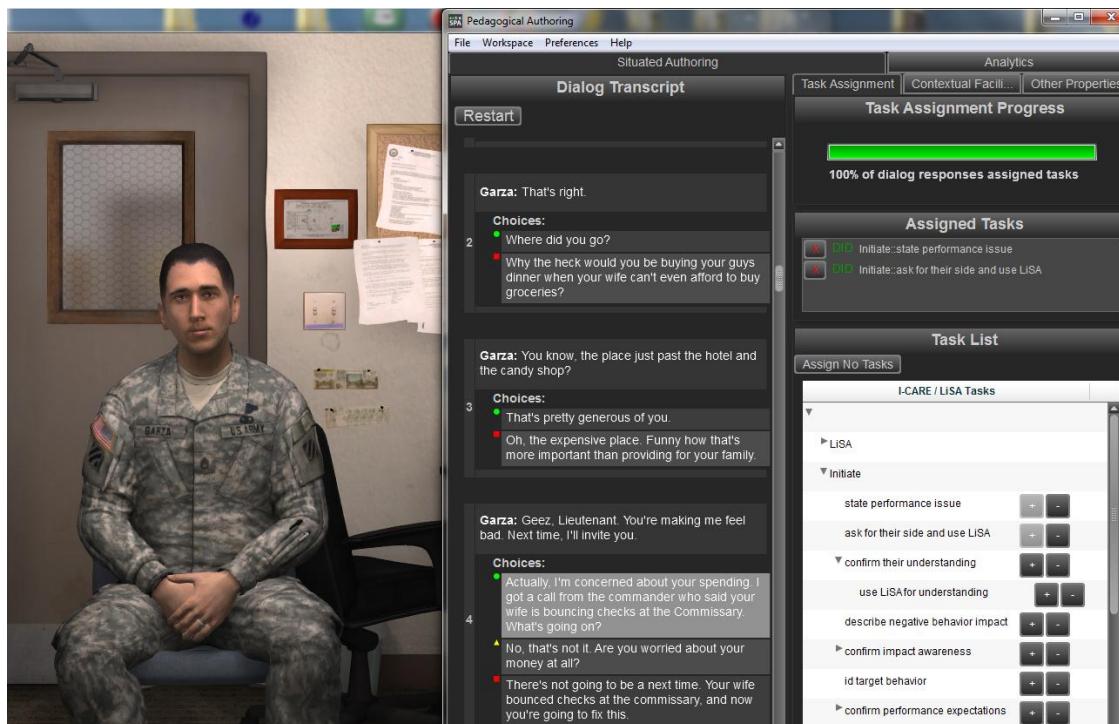


Figure 5. The SitPed Authoring domain tagging screen (Lane, Core, et al., in-press).

SitPed also provides a tool to create simple, hierachal task models (similar to those created in GIFT), which form the basis for the linking screen shown in Figure 5. The full workflow requires an author to (1) create a task list; (2) load a scenario file (these are authored in a commercial product called Chat Mapper, an editor designed specifically for tree-based branching stories; (3) run through the scenario as many times as needed to annotate the possible student choices; (4) create hints and feedback content during those runs;

(5) test the final product out in a “student” view that recreates the actual learning experience (with no additional tools visible). In step 5, authors also have the option to activate the assessment engine and see how the authored model classifies each action (i.e., positive, negative, or mixed) with respect to the task list.

A study of SitPed was conducted in 2014 that consisted of two phases. In the first phase, a set of 11 domain experts were split across three authoring conditions: full SitPed (as described), SitPed Lite (hypertext-only with no graphics or sound), and a specialized spreadsheet. Authors were given scenario data and asked to annotate it both with appropriate tasks (as well as known or likely misconceptions) and tutor messages. In the second phase, the data sets generated from each condition were used to create three separate tutoring systems (randomly using one of the data sets from each corresponding group). Initial results from phase 1 suggest that authors in the SitPed conditions generally wrote *longer* feedback messages and created *more* links to the task list in their authored models, but covered far less of the scenario space. Although the results of phase 2 are still being analyzed at the time of this writing, initial results suggest that learners in all three conditions demonstrated learning gains with trends in favor of SitPed (Lane, Core, et al., in-press).

Our final example of an authoring tool that leverages examples (or equivalently, demonstrations) is actually a more recent addition to CTAT. *SimStudent* (Matsuda, Cohen & Koedinger, 2014; Matsuda, et al., 2013) extends work started with Demonstr8 to pursue the holy grail of authoring: deriving cognitive models (or more generally, expert models with rich representations) based on demonstration of the skill. SimStudent uses inductive logic programming to infer production rules interactively with an author. That is, as an author interacts with SimStudent, the author is tutoring the system – an author can simply show SimStudent what to do at any point or it can let SimStudent perform problem solving steps. In the latter case, the author gives feedback to confirm or correct the emerging model. SimStudent creators have demonstrated that the interactive approach produces higher levels of accuracy in the resulting models (Matsuda, et al., 2014). Similarly, the use of induction allowed the resulting model to go beyond the examples used to train it (MacLellan, Koedinger & Matsuda, 2014). By adding the element of teaching a model (versus simply demonstrating and letting the system observe), there seems to be a payoff in terms of how close the resulting models can get to a hand-authored cognitive model.

Authoring by Demonstration in Simulation-Based Learning Environments

Other key ITS authoring efforts have leveraged ideas from programming by demonstration and authoring with examples. In the area of simulation-based ITS authoring tools, RIDES was a pioneering effort that, among a wide variety of other capabilities, included demonstration as a method for extracting tutoring content (Munro, 2003; Munro et al., 1997; Munro, Pizzini, Johnson, Walker & Surmon, 2006). RIDES provided authors the ability to build their own interfaces and show how to use them in a specialized *demonstration* mode. In addition, RIDES had a rich language for modeling physical systems, identifying expert pathways through the system, and authoring pedagogical feedback. The system incorporates simulation along with other instructional materials to support procedural learning. RIDES has been used

to develop tutors for a variety of domains, including diagnosis of faulty electronic systems and shipboard radar tracking.

A more recent instance of authoring by demonstration, being used in a simulated environment, is the Extensible Problem Specific Tutor (xPST) (S. Gilbert, Devasani, Kodavali & Blessing, 2011). By seeking to capture programming by demonstration in simulated, 3D environments, xPST is unique in its use of freely explorable 3D environments combined with authoring. The system allows authors to link tutoring behaviors to events in a 3D environment to enable the detection of key events (such as recognizing that a step in a procedure has been taken) and then tutoring at those times by delivering hints and feedback. While xPST does not involve authoring directly inside the 3D environment, a web interface is available that has been tested with nonprogrammers who have been able to successfully create tutors for specific skills (S. B. Gilbert, Blessing & Kodavali, 2009).

Themes

In this review of authoring systems that have specifically built to be accessible by nonprogrammers, a few key themes emerge that both highlight the limitations imposed by addressing this audience, and the key areas for future research that we address in our conclusions. Although the systems discussed in this chapter have seen some limited successes, they remain largely in the research and prototype categories with much to be desired in terms of ease of use and accessibility. This seems to be one stable, and undesirable, status of the field (Devasani, 2011; Murray, et al., 2003), although authoring ITS-like interactions have seen dramatic adoption rates (Heffernan & Heffernan, 2014). In nearly every system reviewed here, authors are not spared the inherent complexity of ITS creation (with SimStudent and ASPIRE standing out as possible exceptions). Authors must organize and annotate the vast space of possible learner actions given a specific problem (e.g., example-tracing tutors and SitPed), or alternatively create the complex pedagogical policies needed for ITS decision-making (e.g., REDEEM and xPST). It could be argued that this inherent complexity is unavoidable. In the rest of this section, we summarize these issues by highlighting three themes that emerge from the review.

Theme One: Leveraging of Intuition and Existing Skillsets

All of the authoring tools in this review seek, to varying degrees, to leverage intuitive and simplified elicitation methods such as providing examples or using basic ontology creation tools. The two primary categories of approaches reviewed are (1) intuitive interfaces for capturing domain and pedagogical knowledge and (2) leveraging of an author's existing skillsets and knowledge, such as solving problems in the targeted task domain. Developing GUIs (both for the learner by an author and for the author by a software engineer) are general challenges for authoring systems (Murray, 2003); however, the specific challenge of developing an interface for nonprogrammers is at its core a HCI problem. Interfaces matter, and they matter to a very large degree when system designers ask non-technical audiences to create highly technical content. The second approach, and one that shows up in several different forms in the reviewed systems, is to ask authors to do what they already know how to do: solve problems. Whether it is creating examples, demonstrating a task, or interactively walking through a problem space emulating a learner, this path to building tutors is showing significant promise. Evaluations cited in this review suggest that nonprogrammers can do this, and that they can produce models with reasonably good accuracy.

Theme Two: Tradeoff Between ITS Sophistication and Methods of Elicitation

A closely related theme is the fundamental tradeoff that occurs when a system limits the expressive power of the author (which is a subset of the full space of tradeoffs outlined by Murray (2003, p. 518)). Of

course, limiting expressive power is necessary to simplifying knowledge elicitation (barring a profound discovery in the knowledge elicitation field). For example, REDEEM provides a wide array of easy-to-understand sliders and a logical workflow while allowing authors to create “simple” tutors. Similarly, SitPed supports authors in navigating a large problem space by working in the learner’s environment, but does not produce an expert representation with the richness of most traditional ITSs. But emerging research that leverages machine learning techniques is beginning to close this gap. Both SimStudent and ASPIRE go one step further by attempting to build cognitive models and constraint bases from simplified interactions with an author. These are important advances in the field that directly address known shortcomings. The primary limitation is that these approaches are currently limited to fairly simple task domains that involve symbol manipulation (although important ones, including algebra and arithmetic). This is a typical progression in AI and so we suspect future work will push this research into new task domains and uncharted spaces.

Theme Three: Focused Outputs from Authoring and Carefully Chosen Pedagogical Goals

Authoring tools naturally tend to narrow the problem in appropriate ways to help authors complete the task and produce a usable system. In other words, authoring tools for nonprogrammers may never allow a single author to create an end-to-end tutor for any task domain imaginable, but they can certainly operate in defined spaces and allow authors to tailor existing systems for their particular needs. For example, xPST provides tools for problem-specific tutoring (i.e., a new model for each new problem) and example-tracing is similar in that respect. In these cases, generalized ITSs are not the goal—the goal is to offer step-level help and assess learner actions in specific situations. The norm for nonprogrammer authoring is to populate only the ITS modules (see Figure 1) that matter most given specific pedagogical aims.

Conclusions

The overarching contribution of work on authoring tools for nonprogrammers thus far is that it provides proof-of-concepts that simplify authoring environments, and can be used to produce usable tutors. There continues to be limited studies on the *products* of authoring tools in terms of teaching and learning efficacy since most studies look at either time to create content, or the completeness of a produced model. Simple success at building a tutor is also an important, but limiting measure.

But, it is true that more studies are being conducted with authoring tools. REDEEM has been the focus of a long list of evaluations, even comparing performance against CAI counterparts (and outperforming them). Along those lines, SitPed’s two-phase study model seeks to link authoring affordances to differences in learning. These are promising trends that will lead to better authoring tools that do not simply make authoring easier, but also nudge authors to make decisions that are in line with known principles of learning and effective pedagogy. This large and growing body of work supports the notion that serious attention is being paid to nonprogrammers and scalability issues and suggests that the field as a whole recognizes the problem with having effective ITSs in the world, but only seeing limited adoption.

There are still significant gaps in the research as a whole, however. Very few, if any of the systems reviewed here have treated the act of authoring as a cognitive skill. Although Murray (Murray, 2003) suggests that authoring tools could do a better job *teaching* authors how to author, the skill itself is largely treated as a black box. Highly relevant work on automated cognitive task analysis tools, such as DNA (Valerie J Shute & Torreano, 2003), has had an influence on ITS authoring tools, but there is still much room for improvement in modeling and supporting processes, decisions, and the creativity inherent in good authoring (which is a form of teaching, as evidenced by SimStudent).

With respect to the implications for the future of the GIFT architecture, it seems clear that a truly general system for non-technical audiences is not likely to bear fruit. Rather, given the nature of the systems built thus far and reviewed here, a more productive vision for GIFT would likely lie in creating specialized versions of GIFT for broad domain categories. For example, the history of cognitive tutors (Anderson, et al., 1995) and also with CTAT, many (but not all) of the systems focus on symbol manipulation tasks, such as geometry proofs, equation solving, stoichiometry, and so on. The commonality between a class of domains can be a powerful force in the world of authoring since it can support reuse of formalisms, pedagogy, and even interfaces. While GIFT is currently built around standards for formalizing assessment and pedagogy in any domain, a goal should be to provide tools that differentiate authoring across cognitive, affective, and psychomotor domains. The real challenge of authoring however is removing the required programming component to establish real-time assessment functions. As of now, all authoring is completed in GIFT's domain knowledge file (DKF) where a hierarchical representation of a domain is linked to specific condition classes authored for determining performance on an "at-", "above-", and "below-expectation" rating. Having a situated interface that enables an author to establish performance criteria while building out scenarios and problem sets is recommended.

A second, but related possible implication for GIFT from this work is the idea of building tools for building authoring tools (in the spirit of "compiler compilers" popular in the early years of computer science). In other words, designing a pipeline from engineers to end-users who face specific authoring tasks could further spread the work of building ITSs and expedite the production of ITSs that meet real needs. The problem would still exist for extracting rich knowledge from a non-programmer, but the engineer in the loop design could select the best methods given the task domain and the pedagogical goals, then produce tools that meet the need (e.g., authoring by example for practice support, or adaptive presentation of content using multiple-choice quizzes). The benefit is that GIFT already has many of the tools in place for programmers, so building customization tools and easily adjusted interfaces could imply an authoring tool generator is not far off. The open source nature of GIFT supports this approach, but buy-in is required from the ITS community as a whole so that the authoring tools, processes, and methods evolve as more systems are established using the GIFT architectural dependencies.

A final recommendation for GIFT that emerges from this review highlights the need for a longer-term vision for the future of authoring tools. GIFT is equipped with a number of important capabilities that address the needs of educators and researchers. This combination could play a major role in bringing together the creation of ITSs with authoring tools and evolving them to both improve (in terms of their ability to promote learning) and be shared within a community. In the spirit of the ASSISTments project that has brought together unprecedented numbers of teachers to create, share, and evaluate content (Heffernan & Heffernan, 2014), GIFT could provide the underlying tools that allow authors to create ITSs, collect important data such as pre- and post-test scores, physiological data, and usability data, to act as the hub for a living cycle of growth and improvement of user-generated ITSs. The vision of an instructor creating content then reviewing the aggregated results of days-worth of usage to then make adjustments and improvements to the content, is a powerful one for GIFT developers to consider. It is understood that future GIFT developments are focused on pushing authoring to a collaborative, web-based interface that uses customized tools to remove low-level programming for establishing the modules to deliver an ITS for a domain. The initial focus is on authoring environments to remove the burden of building out XML files that GIFT runs on. To make GIFT more extensible, a focus needs to address learning and training effectiveness tools to assess the effect of authored pedagogical approaches and their influence on performance outcomes. A sought after goal would be to establish probabilistic modeling approaches that can use reinforcement learning techniques to automatically adjust pedagogical policies based on outcomes across a large dataset of learners interacting with the system. This in turn introduces further complications when it comes to authoring tools, in that new techniques would be required to build out these policies when an individual is void of this knowledge and understanding.

In sum, research on authoring tools for nonprogrammers is making a great deal of progress and producing useful results. There is much room for continued improvements and, sadly, ITS authoring tools still fall way behind those of commercially viable CAI authoring tools (Murray, 2003). Researchers should continue to engage in more studies on authoring and focus on accuracy of produced models, usability of tools, and increase efficacy studies that link authoring affordances to learning outcomes. In addition, an issue not addressed in this review but of high importance the need for greater emphasis on multi-party authoring. It is unlikely that one person will be qualified to do end-to-end authoring, and so explicit support for collaboration and workspaces is definitely needed. Authoring researchers should seek to provide explicit support for good pedagogy and the targeting the right level of complexity for desired learning outcomes. Together, and with general frameworks like GIFT to underlie them, it is likely that authoring tools for ITSs will continue to close the gap between research and real-world needs. As newer technologies, such as tablets and immersive games, work their way into educational technologies more deeply, authoring tools will evolve to meet those needs and research should be conducted to inform these inevitable trends.

References

- Ainsworth, S. & Fleming, P. (2006). Evaluating authoring tools for teachers as instructional designers. *Computers in human behavior*, 22(1), 131-148.
- Ainsworth, S., Major, N., Grimshaw, S., Hays, M., Underwood, J. & Williams, B. (2003). REDEEM: Simple Intelligent Tutoring Systems from Usable Tools. In T. Murray, S. Ainsworth & S. Blessing (Eds.), *Authoring Tools for Advanced Technology Learning Environments* (pp. 205-232).
- Aleven, V., McLaren, B., Sewall, J. & Koedinger, K. (2006). The Cognitive Tutor Authoring Tools (CTAT): Preliminary Evaluation of Efficiency Gains. In M. Ikeda, K. Ashley & T.-W. Chan (Eds.), *Intelligent Tutoring Systems* (Vol. 4053, pp. 61-70): Springer Berlin / Heidelberg.
- Aleven, V., McLaren, B. M., Sewall, J. & Koedinger, K. R. (2009). A New Paradigm for Intelligent Tutoring Systems: Example-Tracing Tutors. *International Journal of Artificial Intelligence in Education*, 19(2), 105-154.
- Anderson, J. R., Corbett, A. T., Koedinger, K. R. & Pelletier, R. (1995). Cognitive tutors: Lessons learned. *Journal of the Learning Sciences*, 4(2), 167-207.
- Brusilovsky, P. (2012). Adaptive Hypermedia for Education and Training. *Adaptive Technologies for Training and Education*, 46.
- Cypher, A. & Halbert, D. C. (1993). *Watch what I do: programming by demonstration*: MIT press.
- D'Mello, S. K. & Graesser, A. C. (2012). AutoTutor and affective AutoTutor: Learning by talking with cognitively and emotionally intelligent computers that talk back. *ACM Transactions on Interactive Intelligent Systems*, 2 (4)(23), 1 - 38.
- Devasani, S. (2011). *Intelligent tutoring system authoring tools for non-programmers*. Master's Thesis, Iowa State University.
- Gilbert, S., Devasani, S., Kodavali, S. & Blessing, S. (2011). *Easy authoring of intelligent tutoring systems for synthetic environments*. Paper presented at the Proceedings of the Twentieth Conference on Behavior Representation in Modeling and Simulation.
- Gilbert, S. B., Blessing, S. & Kodavali, S. K. (2009). *The Extensible Problem-Specific Tutor (xPST): Evaluation of an API for Tutoring on Existing Interfaces*. Paper presented at the International Conference on Artificial Intelligence in Education.
- 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.
- Hoffman, R. R., Shadbolt, N. R., Burton, A. M. & Klein, G. (1995). Eliciting knowledge from experts: A methodological analysis. *Organizational behavior and human decision processes*, 62(2), 129-158.
- MacLellan, C. J., Koedinger, K. R. & Matsuda, N. (2014). *Authoring Tutors with SimStudent: An Evaluation of Efficiency and Model Quality*. Paper presented at the Intelligent Tutoring Systems.
- Mark, M. A. & Greer, J. E. (1995). The VCR Tutor: Effective Instruction for Device Operation. *The Journal of the Learning Sciences*, 4(2), 209-246.

- Matsuda, N., Cohen, W. & Koedinger, K. (2014). Teaching the Teacher: Tutoring SimStudent Leads to More Effective Cognitive Tutor Authoring. *International Journal of Artificial Intelligence in Education*, 1-34. doi: 10.1007/s40593-014-0020-1
- Matsuda, N., Yarzebinski, E., Keiser, V., Raizada, R., Cohen, W. W., Stylianides, G. J. & Koedinger, K. R. (2013). Cognitive anatomy of tutor learning: Lessons learned with SimStudent. *Journal of Educational Psychology*, 105(4), 1152.
- Mitrovic, A., Martin, B. & Suraweera, P. (2007). Intelligent Tutors for All: The Constraint-Based Approach. *IEEE Intelligent Systems*, 22(4), 38-45.
- Mitrovic, A., Martin, B., Suraweera, P., Zakharov, K., Milik, N., Holland, J. & McGuigan, N. (2009). ASPIRE: An Authoring System and Deployment Environment for Constraint-Based Tutors. *Int. J. Artif. Intell. Ed.*, 19(2), 155-188.
- Mitrovic, A., McGuigan, N., Martin, B., Suraweera, P., Milik, N. & Holland, J. (2008). *Authoring Constraint-based Tutors in ASPIRE: a Case Study of a Capital Investment Tutor*. Paper presented at the World Conference on Educational Multimedia, Hypermedia and Telecommunications.
- Munro, A. (2003). Authoring simulation-centered learning environments with RIDES and VIVIDS. In T. Murray, S. Blessing & S. Ainsworth (Eds.), *Authoring Tools for Advanced Technology Learning Environments* (pp. 61-91). Dordrecht, Netherlands: Kluwer Academic Publishers.
- Munro, A., Johnson, M. C., Pizzini, Q. A., Surmon, D. S., Towne, D. M. & Wogulis, J. L. (1997). Authoring simulation-centered tutors with RIDES. *International Journal of Artificial Intelligence in Education*, 8, 284-316.
- Munro, A., Pizzini, Q. A., Johnson, M. C., Walker, J. & Surmon, D. (2006). Knowledge, models, and tools in support of advanced distance learning final report: The iRides performance simulation / instruction delivery and authoring systems: University of Southern California Behavioral Technology Laboratory.
- Murray, T. (1999). Authoring intelligent tutoring systems: An analysis of the state of the art. *International Journal of Artificial Intelligence in Education (IJAIED)*, 10, 98-129.
- Murray, T. (2003). An Overview of Intelligent Tutoring System Authoring Tools: Updated Analysis of the State of the Art. In T. Murray, S. Blessing & S. Ainsworth (Eds.), *Authoring Tools for Advanced Technology Learning Environments* (pp. 491-544). Springer Netherlands.
- Murray, T., Blessing, S. & Ainsworth, S. (2003). *Authoring Tools for Advanced Technology Learning Environments*. Dordrecht: Kluwer Academic Publishers.
- Nye, B. D. (2013). *ITS and the digital divide: Trends, challenges, and opportunities*. Paper presented at the Artificial Intelligence in Education.
- Nye, B. D. (2014). Barriers to ITS Adoption: A Systematic Mapping Study. In S. Trausan-Matu, K. Boyer, M. Crosby & K. Panourgia (Eds.), *Intelligent Tutoring Systems* (Vol. 8474, pp. 583-590). Springer International Publishing.
- Shute, V. J. & Psotka, J. (1996). Intelligent tutoring systems: Past, present, and future. In D. H. Jonassen (Ed.), *Handbook for research for educational communications and technology* (pp. 570-599). New York, NY: Macmillan.
- Shute, V. J. & Torreano, L. A. (2003). Formative evaluation of an automated knowledge elicitation and organization tool *Authoring Tools for Advanced Technology Learning Environments* (pp. 149-180). Springer.
- Sottilare, R. A. (2012). *A modular framework to support the authoring and assessment of adaptive computer-based tutoring systems*. Paper presented at the Interservice/Industry Training, Simulation & Education Conference (I/ITSEC), Orlando, FL.
- Tecuci, G. & Keeling, H. (1998). Developing Intelligent Educational Agents with the Disciple Learning Agent Shell. In B. Goettl, H. Halff, C. Redfield & V. Shute (Eds.), *Intelligent Tutoring Systems* (Vol. 1452, pp. 454-463). Springer Berlin / Heidelberg.
- VanLehn, K. (2006). The Behavior of Tutoring Systems. *International Journal of Artificial Intelligence in Education*, 16(3), 227-265.
- VanLehn, K. (2011). The Relative Effectiveness of Human Tutoring, Intelligent Tutoring Systems, and Other Tutoring Systems. *Educational Psychologist*, 46(4), 197-221. doi: 10.1080/00461520.2011.611369
- VanLehn, K., Lynch, C., Schulze, K., Shapiro, J. A., Shelby, R., Taylor, L., . . . Wintersgill, M. (2005). The Andes Physics Tutoring System: Lessons Learned. *Int. J. Artif. Intell. Ed.*, 15(3), 147-204.
- Woolf, B. P. (2009). *Building Intelligent Interactive Tutors: Student-centered Strategies for Revolutionizing E-learning*. Amsterdam, Netherlands: Morgan Kaufmann.

CHAPTER 26 Authoring Instructional Management Logic in GIFT Using the Engine for Management of Adaptive Pedagogy (EMAP)

Benjamin Goldberg¹, Michael Hoffman², and Ronald Tarr³

¹US Army Research Laboratory, ²Dignitas Technologies, ³Institute for Simulation and Training

Introduction

The Generalized Intelligent Framework for Tutoring (GIFT) is a modular architecture built upon standardized tools and methods to support personalized instruction across an array of computer-based training applications (Sotilare, Brawner, Goldberg & Holden, 2013). With an emphasis on personalization, GIFT requires a domain-agnostic pedagogical structure to support the authoring and delivery of varying instructional techniques that can be executed within any learning environment (Wang-Costello, Goldberg, Tarr, Cintron & Jiang, 2013). From an implementation standpoint, this pedagogical framework requires authoring functions that enable a system developer to configure strategy types to specific domain calls and a runtime component that supports strategy execution in real-time based on learner model inputs. During an initial market survey to address this requirement, no open-source solution was recognized that provides adaptive course-flow capabilities based on both individual differences across learners and historical and real-time performance metrics. The goal is to establish functions in GIFT that determine what information to present, how best to present it, what assessments to deliver, how best to grade them, and what guidance to provide and how best to moderate it.

To facilitate this need, the Engine for Management of Adaptive Pedagogy (EMAP) was developed. The EMAP is a pedagogical architecture built around GIFT standards to support the authoring and delivery of adaptive course materials that function on both an inner-loop and outer-loop capacity (Goldberg et al., 2012; VanLehn , 2006). It is structured around David Merrill's Component Display Theory (CDT; Merrill, 1994) and is designed to support adaptive instruction based on the tenets of knowledge and skill acquisition. The framework is designed to assist with two facets of lesson creation. First, it is designed to serve as a guiding template for subject matter experts when constructing intelligent and adaptive course materials that adhere to sound instructional design principles. And second, it serves as a framework to support instructional strategy focused research to examine pedagogical practices and the influence of individual differences on learning outcomes. As such, the EMAP required the development of authoring tools to support the types of functions and data calls linked to its implementation. In this chapter, we define a notional authoring workflow using the EMAP. We highlight the varying authoring tools and processes required for implementing individualized instruction along with the instructional system theory informing its design. In addition, we highlight intended use cases of the EMAP across two distinctly different application environments: (1) EMAP for support of experimentation within the learning sciences research community and (2) EMAP for support of course development across instructional designers and educators in a classroom or distributed setting. This is followed by a description of future work involving the EMAP to support automated creation of learning materials through the use of reusable learning objects and large ontological representations of domains and courses.

The Engine For Management Of Adaptive Pedagogy (EMAP)

The EMAP enhances GIFT through a set of authoring tools and runtime components that enable customized personalization across multiple instructional strategy types. The goal is to provide a means for

instructors and course developers to build highly individualized lesson interactions that manage the student experience based on configured learning paths and real-time performance. In addition, the EMAP serves as an automated lesson manager when a learning event is executed in a self-regulated environment. It is currently being built to manage the delivery of information and course materials, perform assessment related practices to determine an individual's knowledge and skill for a set of concepts linked with a domain, manage the delivery of real-time guidance and interventions found to support learning outcomes, and direct remediation paths based on performance.

The pedagogical framework is intended to support the various interactions that occur when an individual is learning a new topic or skill. This includes the design of actionable logic that determines for every domain of instruction (1) what material to present, (2) how best to assess knowledge and skill linked to a task and set of competencies, and (3) what guidance and feedback practices most reliably impact subsequent performance outcomes. The goal is to establish a data-driven pedagogical model that adapts strategy execution based on the type of learner the interventions moderate most appropriately. To make this a manageable problem space, the first steps associated with the EMAP design was identifying a generalized framework to support its function that is grounded in sound instructional system design (ISD) theory. In the following subsection, we highlight the ISD tenets informing the EMAP's implementation and the constraints and limitations associated with the selected approach.

Instructional Design Informing EMAP Development

The EMAP design was the resulting outcome of a collaborative project between the US Army Research Laboratory (ARL) and the Institute for Simulation and Training (IST) at the University of Central Florida. The task was to define a generalized pedagogical framework that enables the execution of instructional strategies and interventions that are personalized to a given learner and are based on prior empirical evidence supporting their utility. Another goal was to establish a simplistic pedagogical framework that accounts for the multiple interactions associated with learning a new domain, on a general level. The simplistic approach is desired for two reasons: (1) to ease the authoring burden on the course developer in an effort to reduce their workload associated with intelligent tutoring system (ITS) creation and (2) to establish a lesson flow that focuses on domain relevant information that can support adaptive remediation loops.

Following an extensive review of literature focused on instructional system design and pedagogical strategy implementations within computer-based learning environments, the team selected David Merrill's component display theory (CDT) to serve as the guiding framework to formalize the EMAP requirements around (Wang-Costello, Goldberg, Tarr, Cintron & Jiang, 2013). The CDT was originally constructed as a way to simplify theories and models of instruction around a set of core interactions a learner engages in when mastering a new domain (Merrill, 1994). For our efforts, the CDT served as a simplistic framework to organize instructional strategy research that would ultimately inform the development of the EMAP's domain independent structure.

The CDT breaks learning down into four fundamental presentation forms that focus on content and presentation modes. These instructional conditions, known as CDT's Presentation Forms, provide the basic building blocks for the instructional strategies present in the EMAP. CDT indicates two paths when it comes to content as depicted in the Primary Presentation Forms (Figure 1): Content can be presented (expository); or the instructor asks the student to remember or use the content (inquisitory). The content can represent a general case (generality) or it can represent a specific case (instance). Therefore, instruction can be divided into four categories: Expository generality present a general case (Rule); Expository instance present a specific case (Example); Inquisitory generality ask the student to remember or apply the general case (Recall); and Inquisitory instance ask the student to remember or

apply the specific case (Practice). These four categories can be used as high-level metadata descriptors to label training content, with each category applying different pedagogical practices inherent to the learning process. Therefore, instructional strategies can be explicitly defined and categorized within each component of the CDT. This association allows an instructional designer to understand what a piece of content is intended to provide in a lesson context (i.e., this video provides an example for enabling objective x), and further instructional strategies can be defined to inform when this piece of material is most suitable for use. With a framework for organizing content and applying metadata descriptors, a model is required to determine selection criteria and to perform conflict resolution.

CDT Model The Primary Presentation Forms			
Content Mode	Generality	Rule	Recall
	Instances	Example	Practice
		Expository	Inquisitory
		Presentation Mode	

Figure 1. The CDT model

Addressing Individual Differences in the EMAP

With the CDT serving as a generalized pedagogical framework for course construction, the next task was building empirically driven condition statements for the delivery of instructional materials that account for individual differences across a set of learners. The initial construction was facilitated through the creation of an algorithm in the form of a decision tree that informs adaptation based on general learner characteristics. Specifically, the decision tree informs the selection of instructional strategies based on known information about the learner (e.g., learner motivation, learning style, previous experience, etc.) and the logic is established using the Pedagogical Configuration Authoring Tool (PCAT), which is described in detail below. The resulting strategies were identified through an extensive literature review of empirically based research in an attempt to produce a list of commonly applied techniques found to reliably impact learning outcomes.

While many strategies were investigated across the learning sciences community, and many themes recognized, the studies often were limited to single domains and learning environments. The summary of this work can be seen across two publications produced during the execution of the described effort (see Goldberg et al., 2012; Wang-Costello et al., 2013). As a result, establishing a truly generalized pedagogical manager requires future research in an attempt to study the effect of strategies on outcomes and how those specific instructional techniques transfer to different learning environments. In this vein, the EMAP was designed to support both the creation of customized learning applications in addition to providing tools to support robust instructional strategy focused research. The aim for such work is to inform the learning community as a whole and to inform future GIFT developments. In order for GIFT and the EMAP to become widely accepted, its application must be easy to learn and apply. In the next section, we highlight the various authoring processes developed to support EMAP functions and how the tools are designed for use when building out adaptive GIFT managed courses.

Gift Authoring Tools to Support EMAP Functions

A number of authoring processes have been established to support the various pedagogical functions made available by the EMAP. In this section, we introduce the environments currently in place to implement specific configurations that the EMAP operates on. These include (1) the PCAT, (2) the Survey Authoring System (SAS), (3) the Metadata Authoring Tool (MAT), and (4) the Course Authoring Tool (CAT). Each tool serves a different purpose in building out the adaptive logic associated with a lesson using Merrill's CDT to structure interaction and assessment types. It is important to note that each tool is currently implemented within an open-source extensible markup language (XML) editor that has pre-established schemas. These tools are not intended to be the final authoring interfaces, as each process is being converted into web-based tools designed around usability principles and heuristics to support ease of interaction. However, with that said it is important to understand the underlying logic informing the EMAP, as it will support a better applications of its method.

Pedagogical Configuration Authoring Tool

The PCAT is an interface component designed for linking learner relevant information with metadata that drives pedagogical decisions in GIFT. One important note to mention is the domain independent nature of what is authored in the PCAT. More often than not, systems tightly couple pedagogy with the domain, providing little reuse for future developmental efforts. The EMAP differs from this description in that the authoring process is based on generalized tools and methods that translate across domain applications, including the pedagogical strategies the system acts upon (Goldberg et al., 2012). As seen in Figure 1, a developer interacting with the PCAT defines learner model attributes they want GIFT to use for moderating adaptive practices. This is accomplished by linking a set of content descriptors with a given attribute across each of the established CDT quadrants represented in GIFT. These descriptors are a collection of metadata tags that are machine actionable and provide generalized information associated with content and guidance that can be used to configure system interactions in real time. The metadata currently in use is based on the learning object metadata (LOM; Mitchell & Farha, 2007) standard put in place by the Institute for Electrical and Electronics Engineers (IEEE). This provides a set of high level categories (e.g., interactivity type, difficulty, skill level, coverage, etc.) and value ranges (i.e., skill level is broken down into novice, journeyman, and expert) that inform characteristics for a type of interaction.

As an example, one can see in Figure 2 that personalization is moderated by an individual's prior knowledge, self-regulatory ability, motivation, and grit. When establishing these variables, a current assumption is that the learner model has a historical representation of these measures or that the system has a means for collecting inputs to inform classification. Continuing with this example, a developer has the ability to define the attributes that dictate varying pedagogical techniques based on the CDT quadrant. For the Rules and Examples quadrants, an author's primary function is to define ideal types of content with relation to the individual differences variable that moderates it selection. For instance, you can see specific metadata tags linked with the learner attribute Motivation and the value Low. For this type of learner, the PCAT will look for Rules associated content that has tags marked as visual, animations, and low interactivity. The tags currently referenced are based on the previously mentioned instructional strategies literature review we conducted. These configurations are informed by prior instructional strategy focused research, but their true utility as a generalized strategy needs further verification and validation.

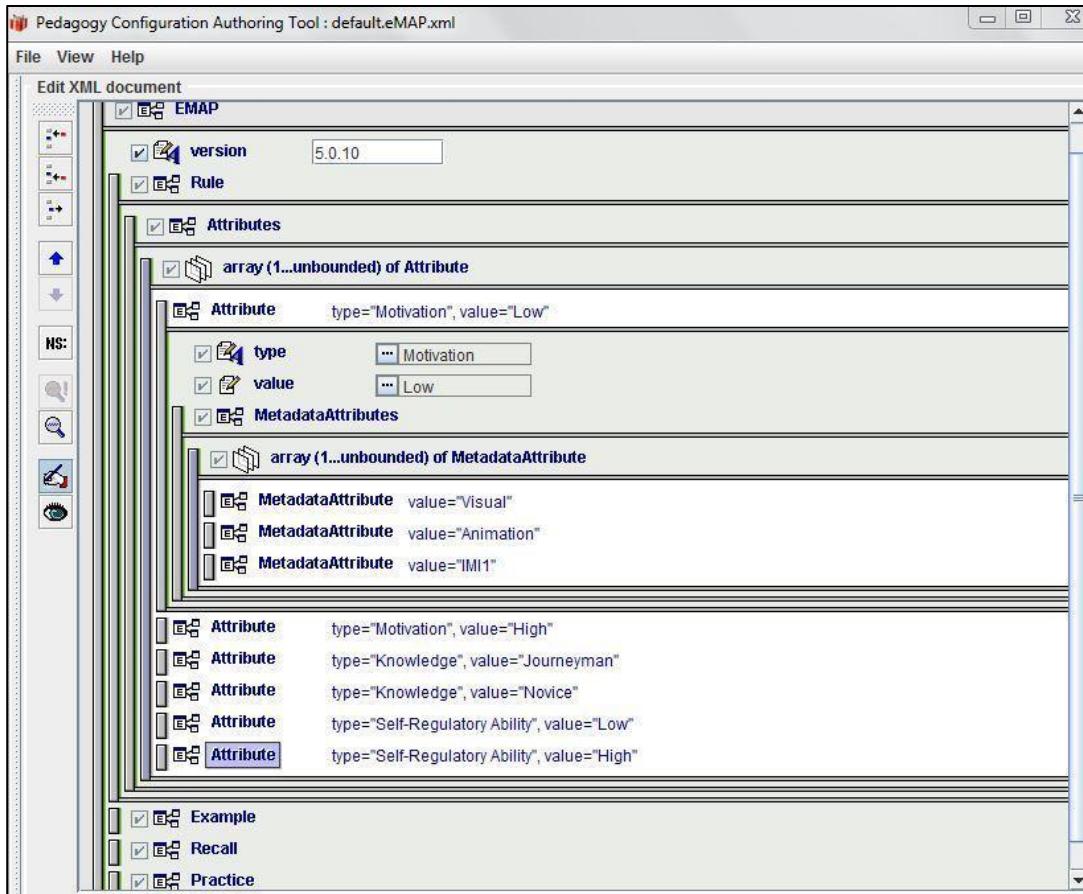


Figure 2. GIFT's PCAT

While Rules and Examples primarily focus on information delivery, the Recall and Practice quadrants are focused on assessment, guidance, and remediation. The Recall quadrant is unique because it focuses on knowledge elicitation through an automatically generated quiz. To enable this capability, there are two additional authoring processes that take place in the SAS. First, one builds a bank of concept questions to generate a recall assessment from. Second, one uses the CAT to configure assessment scoring outcomes that determine if one advances to practice or if a remediation loop is triggered. Each process is covered in more detail below.

In the Practice quadrant, an author establishes adaptive configurations associated with the difficulty and complexity of a scenario, along with timing and specificity configurations linked to concept assessments being managed during run-time. The PCAT is unique because it differentiates the type of pedagogy associated with different types of interactions tied to learning a new topic or domain. This enables personalization on a number of facets linked to instruction, such as modifying delivery of content based on prior knowledge/experiences and learning preferences to focusing guidance and remediation around knowledge gaps and impasses recognized during assessment. While the PCAT establishes configurations of learner information with pedagogical technique, the MAT is set up to build metadata files that associate with a specific learning material or training application.

Metadata Authoring Tool (MAT)

The MAT (Figure 3) is an interface component in GIFT that provides a simple function; it allows one to tag existing training content with concept-dependent metadata that is acted upon by the EMAP. While the PCAT enables an ITS developer to build configurations between learner model data and metadata descriptors linked with pedagogical techniques, the MAT is established to build files that link metadata with actual content that can be delivered for learning purposes. When in an EMAP managed lesson, a learner is directed through defined branching points, called Merrill's Branching, that associate with the four quadrants of the CDT. When a learner starts instruction in the Rules quadrant, the PCAT produced file is referenced for determining the type of content to search for based on a learner's individual profile and the type of metadata their models inform. Next, the EMAP searches through the various files generated in the MAT, for a given lesson, and looks for the closest match with respect to the number of descriptors for a given learning material and the ideal match informed by the PCAT. As such, the MAT is a very important tool as it allows an author and system developer to label their content in a controlled fashion that is then moderated autonomously by the EMAP during run-time.

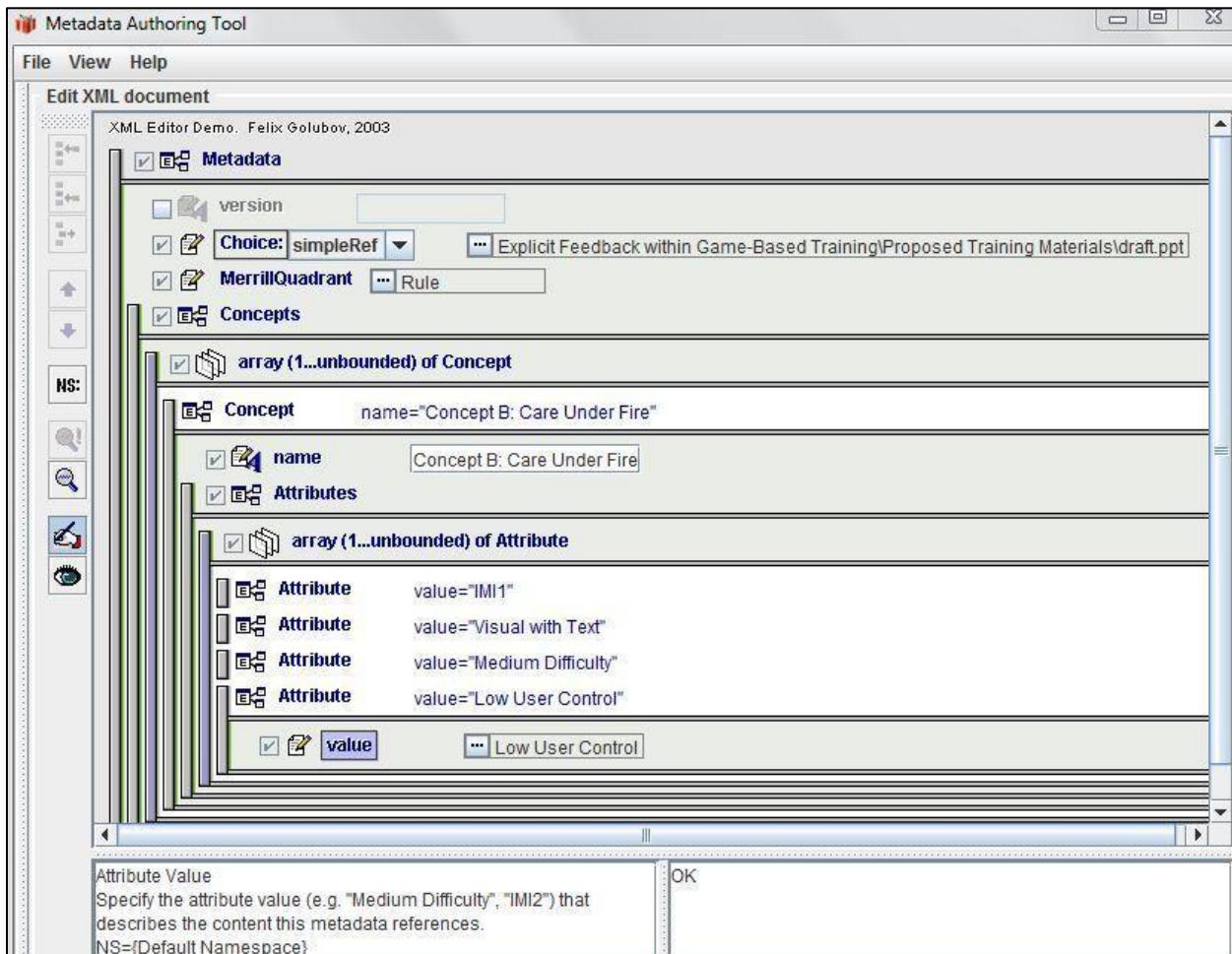


Figure 3. GIFT's MAT

As the MAT is based on EMAP theory and design, it provides multiple input fields that relate to how the pedagogical model operates. These input fields are used to create a GIFT metadata file that when collocated with any number of additional metadata file instances, can be mined during course execution to

determine what material to present to a learner. Figure 3 provides a screenshot of the MAT authoring environment and displays input entries for a specific PowerPoint used in medic training. The initial activity for a developer is to provide a reference to the learning material that is being described with metadata attributes. The MAT allows the author to select whether additional features are needed when the content is delivered such as concept assessments and useful instructional strategies. Next, the author is responsible for entering the quadrant of the CDT this material most appropriately belongs to. This is a drop-down menu that allows one to enter whether the referenced item associates with rules, examples, recall, or practice. This is followed by designating the concepts that are covered for a referenced item. This entry is very important as each Merrill's Branch Point functions on specific concepts defined when building out the GIFT course file (this will be explained in more detail when presenting GIFT's CAT).

Once concepts are identified, it is up to the interfacing developer to begin adding LOM tags that best describe the piece the material in question. In the example above, one can see that the author entered tags that inform a learning object's interactivity, content style, difficulty/skill level, and the amount of user control. An individual author can tag a learning object with as many descriptors as they deem appropriate. With metadata in place to support the delivery of lesson materials and objects across the various EMAP quadrants, an author also has the ability to author concept based assessment prompts for presentation in the Recall quadrant. These assessment questions are developed in the Survey Authoring System.

Survey Authoring System (SAS)

When an individual installs the latest version of GIFT, that user is given access to the SAS. The SAS was developed for two primary purposes: (1) to collect learner relevant information through surveys that are used to update learner model attributes for the purpose of adaptation and (2) to deliver knowledge assessments in the form of questions that can be automatically scored for updating competency states and reporting performance across a set of learning objectives and concepts. In terms of the EMAP, both functions are highly relevant. Firstly, if one defines configurations in the PCAT that act on learner trait information such as motivation or grit, the SAS allows a developer to present validated questionnaires and surveys that can be used to classify that individual learner. These functions are described in more detail in the provided GIFT documentation once downloaded and many examples are made available to work from.

For the purpose of this chapter, we focus on how the SAS is used to support the Recall quadrant in the EMAP by assembling and delivering a set of questions for determining competency and informing remediation paths. The SAS was modified to support the EMAP by providing a “check on learning” function through the assessment of knowledge states across a set of predefined concepts for a lesson. A course developer would start interaction in the SAS by building out a bank of questions that are of relevance to the topic of instruction. As can be seen in Figure 4, GIFT's SAS provides a web interface to author the question, the answer type, and scoring weights that are used in determining knowledge states. A new field added to the question building process is defining a set of associated concepts the specific item informs. For EMAP purposes, this field is very important as it is used in defining the specific question bank that will be used within the Recall quadrant interactions. One can also define the difficulty level of a question in the properties filed, as a course developer can configure the number of questions to be administered for a difficulty level across a specified concept as well.

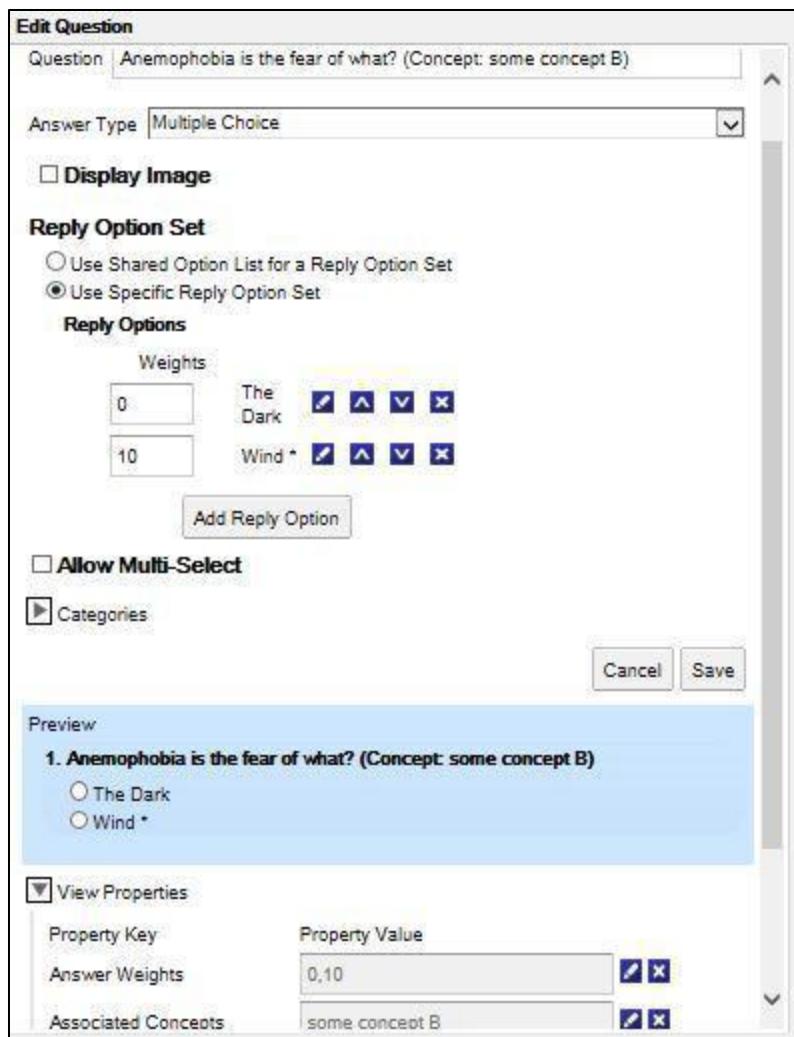


Figure 4. GIFT’s SAS question creation interface

Following question generation, a course developer is then required to build the context for which the generated items will be acted upon. This is achieved by establishing the specified concept question bank the EMAP will operate on within the SAS’s survey context interface window (Figure 5). The survey context is referenced by GIFT’s CAT, which is described next, and is used to identify the questionnaires in the database that are referenced during course construction, as well as establishing question banks that are used for Recall assessment purposes. In the SAS survey context interface, the author has the ability to add concept question banks by selecting a specific concept that was established during question generation. If the author selects a question bank for “Some Concept A,” then all questions that have an “Associated Concept” property field with that particular entry will be included in the designated bank. If one is using the EMAP to deliver a lesson on three concepts, this is where one organizes separate questions banks for each concept, thus providing a granular approach to assessment for the purpose of moderating personalized and performance driven remediation. For each concept question bank, an author has an option of building in multiple questions of varying difficulty so as to avoid assessment on the same question sets (i.e., imagine only 1 question existed for a concept, it would always be presented in a pre-test, 1st recall and 2nd/3rd recall after remediation for the same learner each time the concept was taught). Based on this, it is recommended a user author multiple questions across each concept so as to maintain random assessment selection.

Editing Survey Context - Simple Example EMAP

Context Name:	Simple Example EMAP		
Survey Key	Survey ID	Survey Name	
Grit	63	Simple Learning Ability	<input checked="" type="checkbox"/> <input type="checkbox"/>
Motivation	59	Simple Motivation Survey	<input checked="" type="checkbox"/> <input type="checkbox"/>
Self-Regulatory	64	Simple Self-Regulatory	<input checked="" type="checkbox"/> <input type="checkbox"/>
self-regulatory	60	Self-Regulated Learning Strategies	<input checked="" type="checkbox"/> <input type="checkbox"/>
Add Key			

Concepts Question Bank

- some concept A (18)
- some concept B (18)
- some concept C (18)

Id	Question Text	
572	Choose yes. (Concept: some concept C)	<input checked="" type="checkbox"/> <input type="checkbox"/>
573	This statement is false. True or false? (Concept: some concept C)	<input checked="" type="checkbox"/> <input type="checkbox"/>
574	Only <i>one</i> of the answers below is telling the truth while the others are all liars. Pick the answer that is telling the truth. (Concept: some concept C)	<input checked="" type="checkbox"/> <input type="checkbox"/>
575	What number says "hello" if you hold a calculator upside down? (Concepts: some concept A, some concept B, some concept C)	<input checked="" type="checkbox"/> <input type="checkbox"/>
576	Please select a genre. (Concepts: some concept A, some concept C)	<input checked="" type="checkbox"/> <input type="checkbox"/>
577	Speed skating originated in which country?	<input checked="" type="checkbox"/> <input type="checkbox"/>

Figure 5. GIFT's SAS survey context interface for the EMAP

One can also see that this survey context includes four independent questionnaires that are linked to the context name that is used when creating a lesson or course. For this instance, the identified surveys are used at the beginning of a lesson to update learner model attributes that are acted upon by the EMAP quadrants as configured in the PCAT. With an established EMAP configuration completed in the PCAT, an array of course materials and objects with associated metadata files, and the construction of question banks for check-on-learning practices, the next step a course developer takes is using the CAT to build out a sequence of interactions and transitions that will guide a learner automatically through lesson materials.

Course Authoring Tool (CAT)

The GIFT CAT is the final authoring interface a course developer interacts with once all of their materials and assessments have been appropriately configured for EMAP run-time. It is in this environment that an

author designates the sequence of interaction a learner will experience for a specific lesson. The lesson is composed of defined transitions that dictate what is presented next (see GIFT documentation for a full list of available transitions and their associated descriptions). The transitions of interest for the EMAP are Present Survey and Merrill's Branch Point. The Present Survey transition enables an author to select a questionnaire present in the SAS that will be delivered to a learner at any point in the course flow. For EMAP purposes, a course developer might call for surveys upfront to collect information on learner model attributes, such as measuring an individual's motivation for a topic and applying a pre-test assessment to establish initial performance states that can dictate lesson flow. This requires an author to designate a survey context present within the SAS that determines what surveys and concept question banks are made available (Figure 6). These referenced surveys can then be selected when a Present Survey transition is entered. As one can see in Figure 6, a motivation survey and knowledge pre-test are both entered as individual transitions to start the lesson. A pre-test is unique because it can be used to modify a user's experience within a lesson. An author can specify performance criteria for bypassing components of instruction based on outcomes associated with this pre-test.

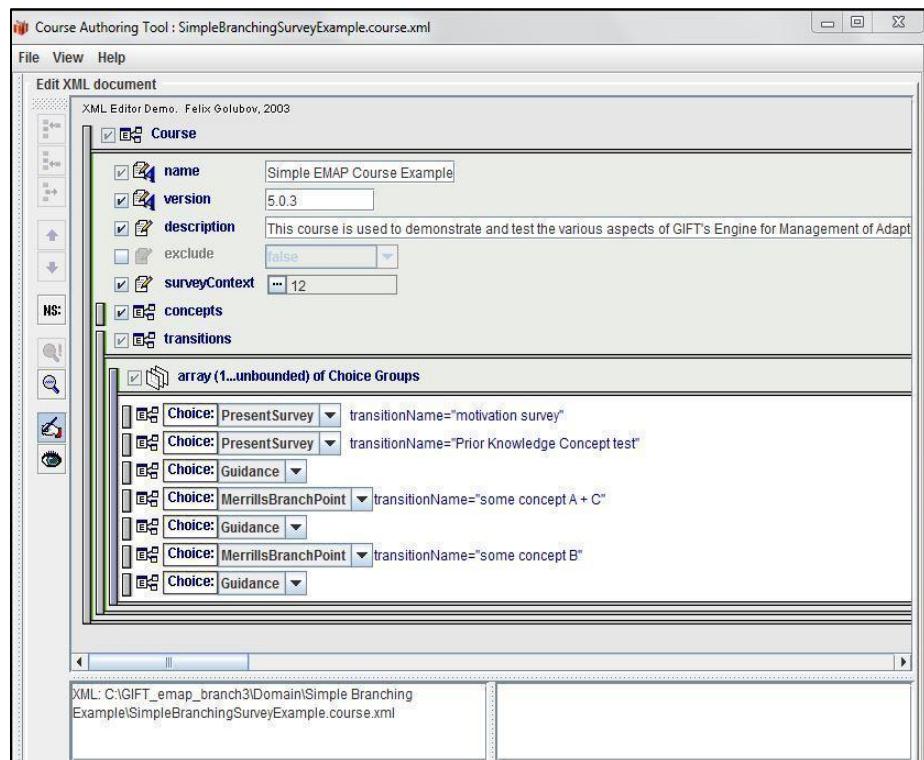


Figure 6. GIFT's CAT

With defined surveys used to inform learner attributes and a pre-test to gauge initial knowledge levels, the next step for an author is establishing Merrill Branch Points. These branch point transitions are managed by the configurations established in the PCAT and MAT, as described above. Within the CAT, the author selects specific concepts used to build the quadrants of the CDT for a given branch point. Within a single course, one can define multiple branch point transitions. In figure 6, one can see two branches were entered, covering a total of three concepts to be instructed. Once the author has selected the concepts a branch point associates with, the author then has the opportunity to configure interactions across the quadrants of the CDT. The CAT configurations allow a user to deselect certain quadrants that would be bypassed when a learner enters that branch transition. In Figure 7, one can see the visual breakdown of the quadrants where the practice field was not checked, thus bypassing that interaction.

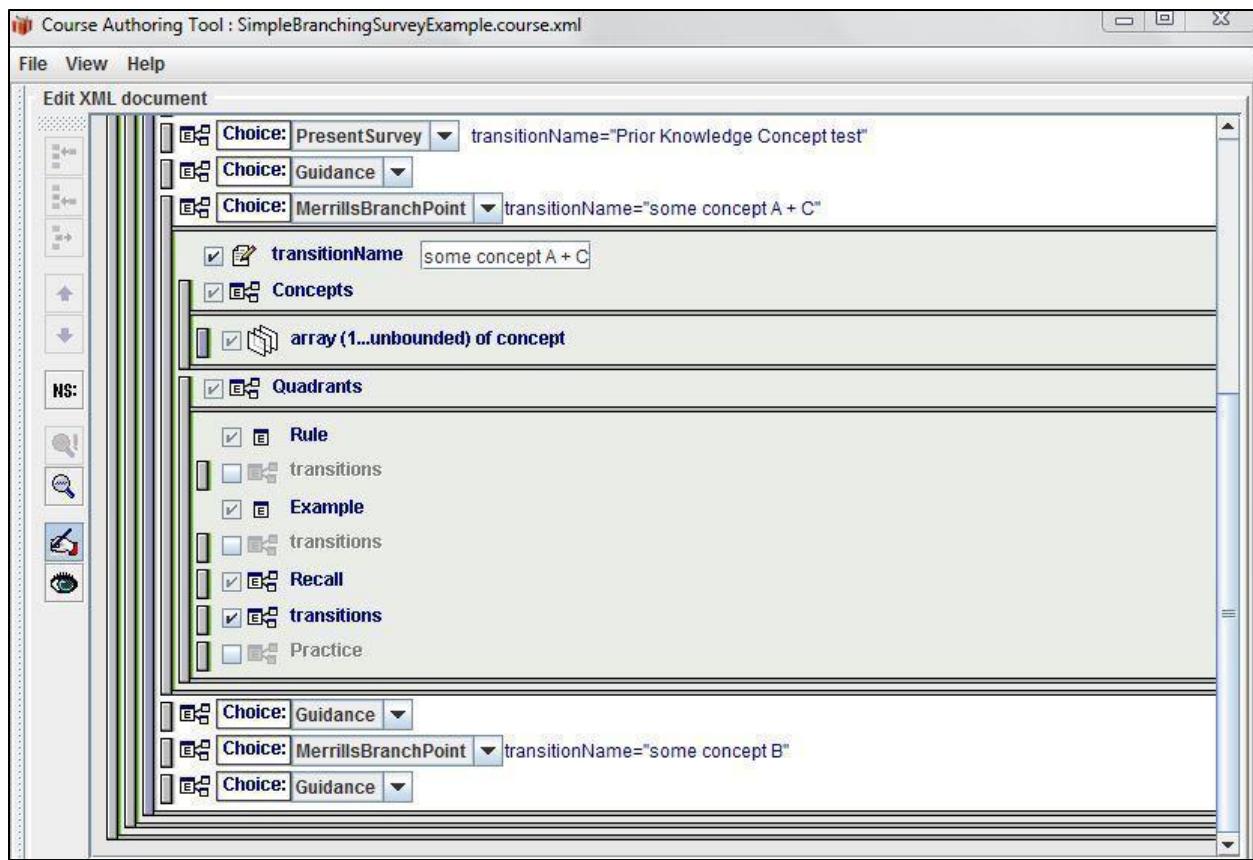


Figure 7. GIFT's CAT: Merrill's Branch Point Transition

The last configurations an author can make within the CAT is specifying what will be delivered to a learner within the Recall quadrant of the CDT (Figure 8). Once a user specifies the survey context referenced in the SAS and the concepts a Merrill's Branch Point is used to manage, the next step is defining the number of questions to deliver for knowledge state assessment purposes and the scoring rules that determine the state value to communicate out to the learner module. When defining the number of questions to deliver, the inputs are used to identify matches within the established question banks that are built in the survey context interface within the SAS. It first looks for questions linked to a concept and then it looks for metadata values used to classify a difficulty ranking. In the example in Figure 8, the user inputs the delivery of 1 question for the associated concepts across all three difficulty levels of easy, medium, and hard.

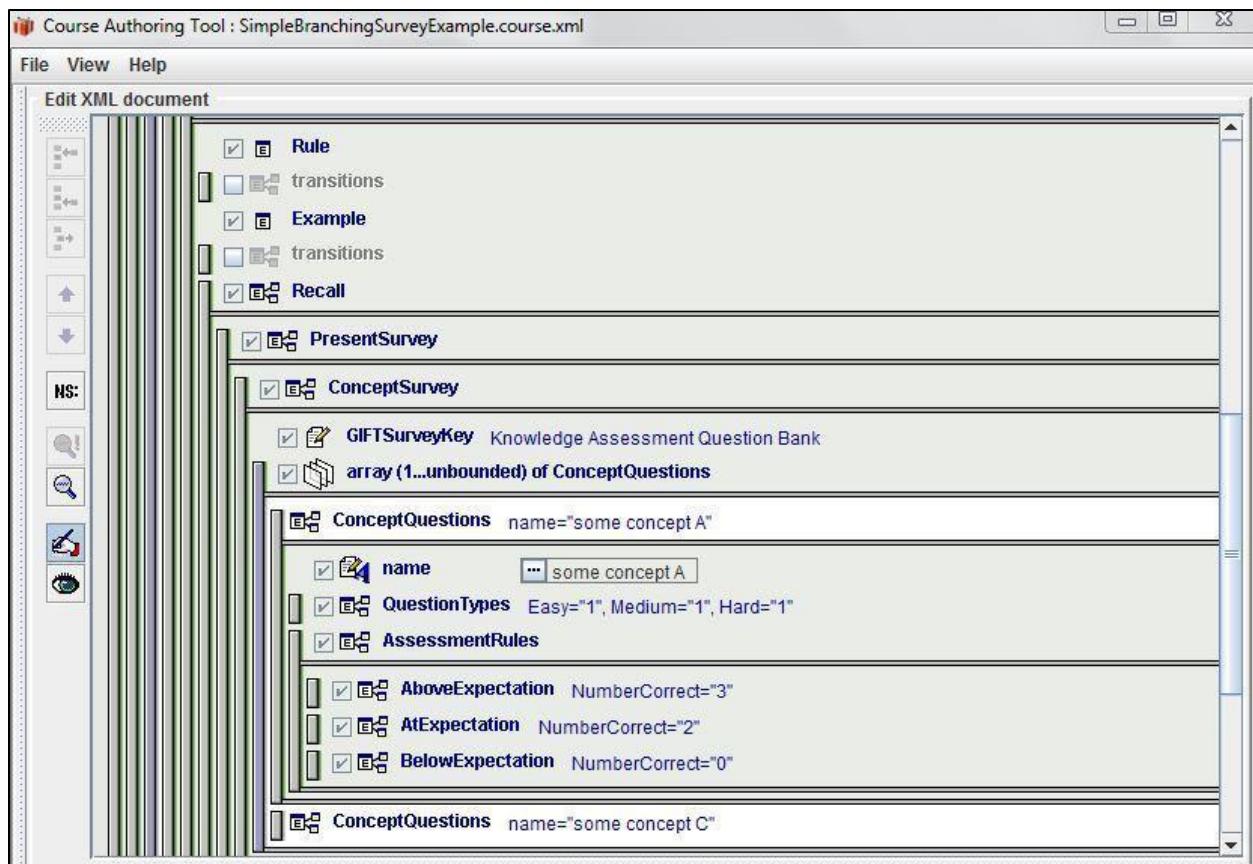


Figure 8. GIFT's CAT: Recall Assessment Scoring Rules

Below the question types, the assessment rules are defined. In the current state of the EMAP development, assessment rules are simple condition statements linked to the number of questions a learner gets scored correctly. These condition statements are important because they are used to determine if the learner proceeds out of the Recall quadrant and into the Practice quadrant, if that interaction is available, or if a learner triggers a Remediation Loop. Remediation Loops work in the same fashion for the Recall and Practice quadrants. It is triggered by a form of assessment across both the knowledge and skill states maintained in GIFT's learner model. If a concept is scored at “below expectation” or “at expectation” the EMAP initiates remediation by routing the learner to a Rule or Example quadrant for the presentation of additional learning objects.

The selected path of remediation is based on the performance state resulting from an administered assessment. For “below expectation” outcomes, the user is delivered both Rule and Example materials focused on the specific concept found below criteria. For “at expectation” outcomes, the user is delivered just Example materials before a follow-on assessment. In the event that multiple forms of content exist for a single concept, an algorithm has been established that performs conflict resolution for initial content selection as well as secondary remediation selection. This is important, as an author can build in multiple representations of a single concept, thus providing more options to personalize a lesson around. All remediation paths are reconfigurable. The current prioritization logic in place is as follows:

- (1) If possible, don't present domain content that has already been presented.

- (2) Maximize the needed coverage of concepts by selecting the fewest content to present at that time.
- (3) Maximize the appropriateness of the content by selecting the best match of EMAP learner state attributes to the available metadata attributes.
- (4) If available, use the content's paradata (i.e., usage data about learning resources) to trim the number of content choices that cover the same set of needed concepts.
- (5) Choose randomly from the content choices that cover the same set of needed concepts.

The EMAP's Intended Function

When designing the EMAP, two primary end-users were considered. These included traditional educators and training developers that would use the EMAP toolsets to author adaptive courseware materials they could distribute across classes and training organizations, and it included researchers and scientists within the learning sciences community that have interest in adaptive instruction, individual differences, and pedagogical heuristics. Making the distinctions between these two user groups is important because it will dictate the primary authoring tools they will interact with.

The EMAP for Educators and Training Developers

The main difference between these two user sets is their interest and background associated with individual differences based research. A simple assumption with our design is that the primary group of educators and training developers building course materials with the EMAP will not be experts in pedagogical theory and practice, thus placing an emphasis on GIFT to accommodate this lack of knowledge. In fact, for this specific user group, the goal is to have an empirically driven pedagogical configuration established within the PCAT that would require no manipulation. It will have support for all the learner attributes available within GIFT along with pedagogical strategies to implement for a given learner across the type of interactions experienced within the CDT quadrants.

As the goal of the EMAP is to support streamlined creation of personalized educational content, the primary interactions this user group will engage in are housed within the MAT, SAS, and CAT. As mentioned before, the PCAT will be off limits to these individuals because it will be operated in a default capacity as it is intended to be empirically driven. However, a primary responsibility of these authors is to tag available content, practice scenarios, and assessments with LOM descriptors that the PCAT is configured around. A training developer will need to create a metadata descriptor file for each piece of content and associate those files with the various quadrants of the CDT. As these authors are assumed to be experts in the field they are building a course around, their main responsibility is bringing appropriate materials to use for instruction and to set them up so that GIFT can manage their delivery.

Following interaction in the MAT, an author is then responsible to use the SAS to establish the surveys that will be used to inform learner model attributes. An author is also then responsible to build concept question banks that are ultimately used for pre-/post-test assessments as well as checks-on-learning conducted within the Recall quadrant. Following, this class of authors would then interact in the CAT to build out the sequence of transitions a learner would progress through, including the number of Merrill's Branch Points to be experienced. Once interaction in the CAT is complete, this user group can then run an EMAP managed course and distribute it as required.

The EMAP for the Learning Sciences Research Community

When considering the authoring experience across the EMAP tools for the learning sciences community, an assumption is that this user group is interested in the creation of experimental groups for the intentions of running studies to support feedback and individual differences based research. The main differences between these individuals and educators/training developers are that the learning sciences researchers will interact heavily with the PCAT to build out specific configurations in support of their research questions. This involves modifying current attributes available in the EMAP as well as adding additional variables with the goal of assessing its effectiveness in informing adaptive pedagogical approaches. Within the PCAT, an author references a learner model attribute as well as metadata descriptors associated with the ideal type of content to deliver to that specific learner. For research purposes, the learning sciences user group will be responsible for adding and removing available attribute tags for both the learner model and metadata variables and associated values. Each of the available tags are referenced in an enumeration file in GIFT's source code that can be modified to support the type of learner attribute and metadata a researcher desires the EMAP to act on. These files can be located at the following directory off of GIFT root:

GIFT\src\mil\arl\gift\common\enums\MetadataAttributeEnum.java and
GIFT\src\mil\arl\gift\common\ enums\ LearnerStateAttributeNameEnum.java.

For each of the attributes represented in the learner state attribute enumerations file, a user will need to author an additional .java file that specifies the available tags made available for that variable when manipulating the PCAT authoring tool. Many examples are available to work from in the GIFT\src\mil\arl\gift\common\enums\ folder. Following any manipulation within GIFT's source code, that user will need to recompile the program to update variables available in the authoring process.

In implementing the EMAP for support in the learning sciences research community, there are also a couple more assumed dependencies associated with the authoring tools. While an author will be required to modify enumeration files in the source code to introduce variables of interest not currently supported in the baseline version of GIFT, an author will also be required to make additions in the SAS for the purpose of collecting individual differences metrics used to inform variable states that dictate strategy selection. In addition, GIFT also supports tracking and inferring upon affective based data for the purpose of diagnosing cognitive and emotional states experienced during a learning event. None of the current EMAP configurations are built to support acting on this type of assessment. The PCAT would need to be modified for this very purpose, along with identifying required classifiers that would be developed within the learner and sensor modules of GIFT.

Future Work

In its current baseline state, the EMAP provides an automated lesson manager for a GIFT course with some caveats. For example, the course author has but only one option for a “check on learning” within the Recall CDT quadrant and that is an assessment composed of items selected at runtime from a designated question bank. Other forms of knowledge assessment could be given, such as creating a static pre-authored survey useful for research settings in which the consistency across users is desired or using AutoTutor (Nye, Graesser & Hu, 2014) web services integrated with GIFT. Using AutoTutor allows GIFT to hold a conversational discourse in natural language between one or more agents and the learner. This interaction can be used to elicit a learner’s comprehension of prior instructed concepts in their own words, which provides evidence for deeper understanding of a concept and its relationships.

As additional EMAP functionality and content type support is expanded, GIFT can increasingly serve as the course delivery medium for third party applications such as Tools for Rapid Development of Expert Models (TRADEM; Brown, Martin, Ray & Robson, 2015). TRADEM is designed to rapidly assist in the transformation of domain content repositories into a hierarchical expert model. Ultimately, the user can

select to export the model into a simple GIFT course containing a Merrill's branch point course transition. The course package includes at least one Rule and one Example based auto-generated, metadata tagged PowerPoint with slides that contain the information found in the expert model. In a future release of TRADEM, it will automatically populate the SAS question bank with a set of questions related to the course concepts, which will be used in the Recall CDT quadrant's "check on learning." Overall, this process will enable GIFT authors to quickly generate a simple course given a content repository containing many different files in numerous formats.

Continuing with the theme of auto-generated EMAP informed courses, one might have or want to build an ontology representation of domain concepts and objectives where learning objects can be embedded. Such an ontology would represent the knowledge and skills associated with a given domain, while also providing direct linkages to content/problems/scenarios that can be used to instruct and assess those associated concepts. In this manner, GIFT plans to leverage the a concept applied in the SCALE program (Spain et al., 2013) where the leaf nodes in an established ontology related to land navigation are linked with reusable learning objects (RLOs) such as images, PDFs, question banks, training scenarios, etc. Using an ontology with established learning objects can be informed by EMAP relevant metadata descriptors, thereby exposing those assets to the data mining used during GIFT course execution. All of this would facilitate an educator to easily, and in some cases autonomously, build courseware that is adaptive and linked with an ontology. A possible issue with this approach is controlling inputs and performing semantic reasoning that allows linking concepts that are the same but labeled differently possibly due to different data sources, authors, organizations (e.g., "map reading," "reading a map," "read map," "understanding a map," etc.).

Finally, as mentioned above, another major push with the EMAP is translating its authoring processes into an intuitive web-based interface that creates a transparent workflow of establishing all configurations required to run an adaptive GIFT managed lesson. This will require the application of human factors informed design that adheres to heuristics identified in the usability literature. Creating this authoring workflow will require extensive research focused on formative evaluations that create iterative development cycles to make the user experience as seamless as possible.

Conclusion

In this chapter, we presented the tools and methods formalized through the development of GIFT's EMAP. Authoring adaptation in GIFT is dependent on the functions made available in the pedagogical module. The EMAP was built as an instructional framework that guides pedagogical authoring and implementation within GIFT. The eMAP is structured around David Merrill's CDT and is designed to support adaptive instruction based on the tenets of knowledge and skill acquisition. The framework is designed to assist with two facets of lesson creation. First, it is designed to serve as a guiding template for subject matter experts when constructing intelligent and adaptive course materials that adhere to sound instructional design principles. Second, it serves as a framework to support instructional strategy focused research to examine pedagogical practices and the influence of individual differences on learning outcomes. We also described the fundamental components that make up the EMAP, followed by the authoring workflow associated with its implementation.

References

- Brown, D., Martin, E., Ray, F. & Robson, R. (2015). *Using GIFT as an adaptation engine for a dialogue-based tutor*. Paper presented at the Generalized Intelligent Framework for Tutoring (GIFT) Users Symposium (GIFTsym2).

- Goldberg, B., Brawner, K. W., Sottilare, R., Tarr, R., Billings, D. R. & Malone, N. (2012). Use of Evidence-based Strategies to Enhance the Extensibility of Adaptive Tutoring Technologies. Paper presented at the Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC) 2012, Orlando, FL.
- Merrill, M. D. (1994). *The descriptive component display theory*: Educational Technology Publications, Englewood Cliffs, NJ.
- Mitchell, J. L. & Farha, N. (2007). Learning Object Metadata: Use and Discovery. In K. Harman & A. Koohang (Eds.), *Learning Objects: Standards, Metadata, Repositories, and LCMS* (pp. 1-40). Santa Rosa, CA: Informing Science Press.
- 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.
- Sottilare, R., Brawner, K. W., Goldberg, B. & Holden, H. (2013). The Generalized Intelligent Framework for Tutoring (GIFT). In C. Best, G. Galanis, J. Kerry & R. Sottilare (Eds.), *Fundamental Issues in Defense Training and Simulation* (pp. 223-234). Burlington, VT: Ashgate Publishing Company.
- Spain, R., Mulvaney, R. H., Cummings, P., Barnieu, J., Hyland, J., Lodato, M. & Zoellick, C. (2013). Enhancing Soldier-Centered Learning with Emerging Training Technologies and Integrated Assessments. Paper presented at the Interservice/Industry Simulation Training & Education Conference (I/ITSEC), Orlando, FL.
- VanLehn, K. (2006). The behavior of tutoring systems. *International Journal of Artificial Intelligence in Education*, 16(3), 227-265.
- Wang-Costello, J., Goldberg, B., Tarr, R. W., Cintron, L. M. & Jiang, H. (2013). Creating an Advanced Pedagogical Model to Improve Intelligent Tutoring Technologies. Paper presented at The Interservice/Industry Training, Simulation & Education Conference (I/ITSEC).

Chapter 27 Tiering, Layering and Bootstrapping for ITS Development

Eric Domeshek, Randy Jensen, and Sowmya Ramachandran
Stottler Henke Associates, Inc.

Introduction

Intelligent tutoring system (ITS) authoring tools aim to lower the cost of code and content development, maintenance, and reuse. We discuss three techniques for cost-containment: *tiering*, *layering*, and *bootstrapping*. Our discussion focuses on the critical task of assessment authoring for *situated tutors* (Schatz, Oakes, Folsom-Kovarik and Dolletschi-Lazar, 2011). Situated tutors are a class of ITSs where training is conducted in a scenario-playing experiential environment with intelligent adaptive instruction, including micro-adaptation within scenarios and/or macro-adaptation across scenarios (Shute, 1993). Exercise environments are often quite complex—e.g., simulations of helicopter flight controls, ship battle stations, sensor suites, or command and control systems—and often include components for interacting with simulated teammates, customers, or adversaries. Student assessment is complicated by the nuances of the domain and task, the need to track activity in such complicated simulations, and the need to generate simulation behaviors to create particular learning opportunities. Based entirely on the data and cues available from the training environment, automated assessment mechanisms are responsible for producing judgments of performance at a fidelity that meets training objectives by being sufficiently comparable to human instructor assessments. The complexity of the assessment mechanism in this kind of environment often translates to significant development costs, and thus the need for authoring techniques aimed at reducing costs by structuring the process, component, and content development tasks.

A tiered structure of authoring tools offers a way to tailor knowledge elicitation and engineering for different classes of experts, ranging from those with domain expertise or instructional knowledge, to authors with skills in domain or task modeling, logical and symbolic reasoning, basic scripting, or even advanced programming. A layered approach to modeling allows for composition of model components. It promotes reuse of general knowledge where feasible, while allowing for context-specific knowledge to fill gaps as needed. A bootstrapping approach involves generalizing assessment knowledge from specific instances to scenario-independent mechanisms. Bootstrapping techniques we have applied include incremental rule condition generalization and student action templates created by demonstration and generalization.

These techniques fit an ITS development approach emphasizing incremental example-driven evolution over upfront complete model development. We aim to gain the advantages of rapid/cheap initial capability while still ensuring that instructional unit costs taper over time. We describe our experience building ITS authoring tools that embody approaches to tiering, layering, and bootstrapping.

Related Research

Tiered authoring is an intuitive solution to the challenge of ITS authoring and, not surprisingly, has been implemented in one form or another by a variety of authoring tools. Murray (1999) discusses meta-authoring tools as a potentially effective approach to addressing the usability and power trade-off. Meta-authoring tools are a means for creating special-purpose authoring tools using general-purpose authoring tools. The latter are designed to be applicable to a wide variety of domains and support several types of pedagogical approaches and thus would present a larger degree of authoring complexity. The idea is that

highly skilled authors could use these tools to create special-purpose authoring tools that are targeted at a specific domain and a subset of pedagogical styles (Qiu & Riesbeck, 2005; Hsieh, Halff & Redfield, 1999). Limiting the scope of the tool in this manner makes it possible to design these authoring tools to be more usable and less demanding in terms of authoring skills. Meta-authoring is an example of the tiered authoring approach that we discuss below. For a more recent example, Nye, et al. (2014) describe a tool that uses a tiered approach for augmenting web content with AutoTutor-like dialogues.

Layered authoring of ITS content is primarily intended to enable and promote reuse. This fits one of the authoring tool methods enumerated by Murray (1999). However, given our bias toward example-driven situated tutor development, we focus on reuse across scenarios rather than across entire tutor applications. Layering is *not* aimed at reusing preexisting media or courseware as in REDEEM (Major, Ainsworth & Wood, 1997) or the Shareable Content Object Reference Model (SCORM) standard (ADL, 2009), nor is layering primarily concerned with reusing computational or user interface components (e.g., as in SIMQUEST; deJong et al., 1998). Layering, as presented here, would not make sense in the context of authoring a fully general domain model capable of solving any problem the tutor might pose to a student.

Bootstrapped acquisition of domain knowledge has been gaining traction in recent years. A common approach is to use machine learning algorithms to learn initial domain knowledge and refine it on an ongoing basis (Kumar, Roy, Roberts & Makhoul, 2014; Aleven, McLaren, Sewall & Koedinger, 2006). More recently, SimStudent advances the concept even further where the ITS is an active learner, i.e., it learns from an initial set of demonstrated solutions and refines its knowledge by actively validating it on examples while asking for feedback and help, much like a student. Another bootstrapping approach is limited to using student performance data to improve a tutor's assessment or student modeling knowledge while the initial knowledge itself is handcrafted (Baker, Corbett & Aleven, 2008; Barnes & Stamper, 2008). However, such bootstrapping is primarily envisioned as an automated process, whereas we emphasize the more pragmatic approach of keeping authors in the loop to deal with the commonly required representational shifts.

Discussion

Tiered Authoring

The challenge of ITS authoring lies in developing user-friendly tools that allow subject matter experts or instructional designers to create complex pieces of knowledge. The targeted authors typically do not possess the kinds of computational/logical modeling skill required to create the knowledge that informs ITSs. This skill gap is often too large to be bridged by authoring tools. One way to reduce this gap is to limit the complexity (breadth and/or depth) of knowledge provided to the tutor, thereby reducing modeling complexity. However, this comes at the cost of reducing the "intelligence" of the ITS. An alternative is to partition the space of the knowledge to be authored into sections that can be authored by different people with different skillsets. One approach is to partition the knowledge into modules, each of which might require a different skill set for authoring. For example, an author with expert modeling skills may author performance assessment rules while an instructional designer might configure the pedagogy. An alternative way to partition is to develop tiers of knowledge with one tier combining and specializing another. Templates are an example of intermediary structures or abstractions that can be combined together and instantiated to capture the knowledge required for assessment and tutoring. A tiered approach to authoring provides a way to divide and distribute the task so as to match the skillset and knowledge of the variety of contributors collaborating on the ITS.

The EarthTutor ITS and authoring tool illustrates this approach. This ITS was designed for NASA to teach remote sensing image processing, a domain in which students analyze satellite data using an image

processing application. The objective of EarthTutor is to teach students to use image processing tools by completing exercises related to specific questions about an image. EarthTutor structures an exercise as a series of cards, each containing interactive behaviors embedded in HTML pages. The interactive behaviors consist of questions and real-world tasks the student must complete in the host application. Embedded in these behaviors is the logic for monitoring the student's actions, presenting feedback, and updating the student model.

EarthTutor provides a tiered tool suite for authoring these behaviors. At the foundational level, advanced authors use a graphical flow chart tool to combine ITS and host-application primitives into hierarchies of reusable parameterized assessment behaviors. A novice tier authoring tool, then, allows less skilled authors to use previously defined flow charts from the behavior library to create interactive cards for exercises. The novice tool enables authors to select *behavior templates* from the behavior library, instantiate their parameters, and embed them in a card with other HTML content. Adding an instantiated template to a card indicates (1) that the flow chart linked to the template should be executed when the card is displayed, and (2) that the student interface should replace the template with a user interface (UI) component (defined by the advanced author in the flow chart). This approach allows novice authors to tailor tutoring behaviors to their own pedagogical needs using parameters, but the interface is reduced to what you see is what you get (WYSIWYG) HTML and simple forms. This novice tier tool also lets authors define a hierarchical course structure in which a course contains labs and labs contain cards. The author can set properties for the courses, labs, and cards such as prerequisites and student modeling parameters.

This two-tiered authoring architecture allows subject matter experts to create image processing exercises with automated intelligent tutoring support by piggybacking on the more advanced authors who populate the behavior library. Since the templates are designed to be reusable objects, the work invested in creating them can be amortized over many exercises.

Figure 11 shows the authoring interface for creating behavior templates. In this example, the author has specified the steps for opening a specific image file using the application's menu. Executing this behavior will show the student the necessary steps to find and open a file, monitor their actions, and provide feedback if they open the wrong file.

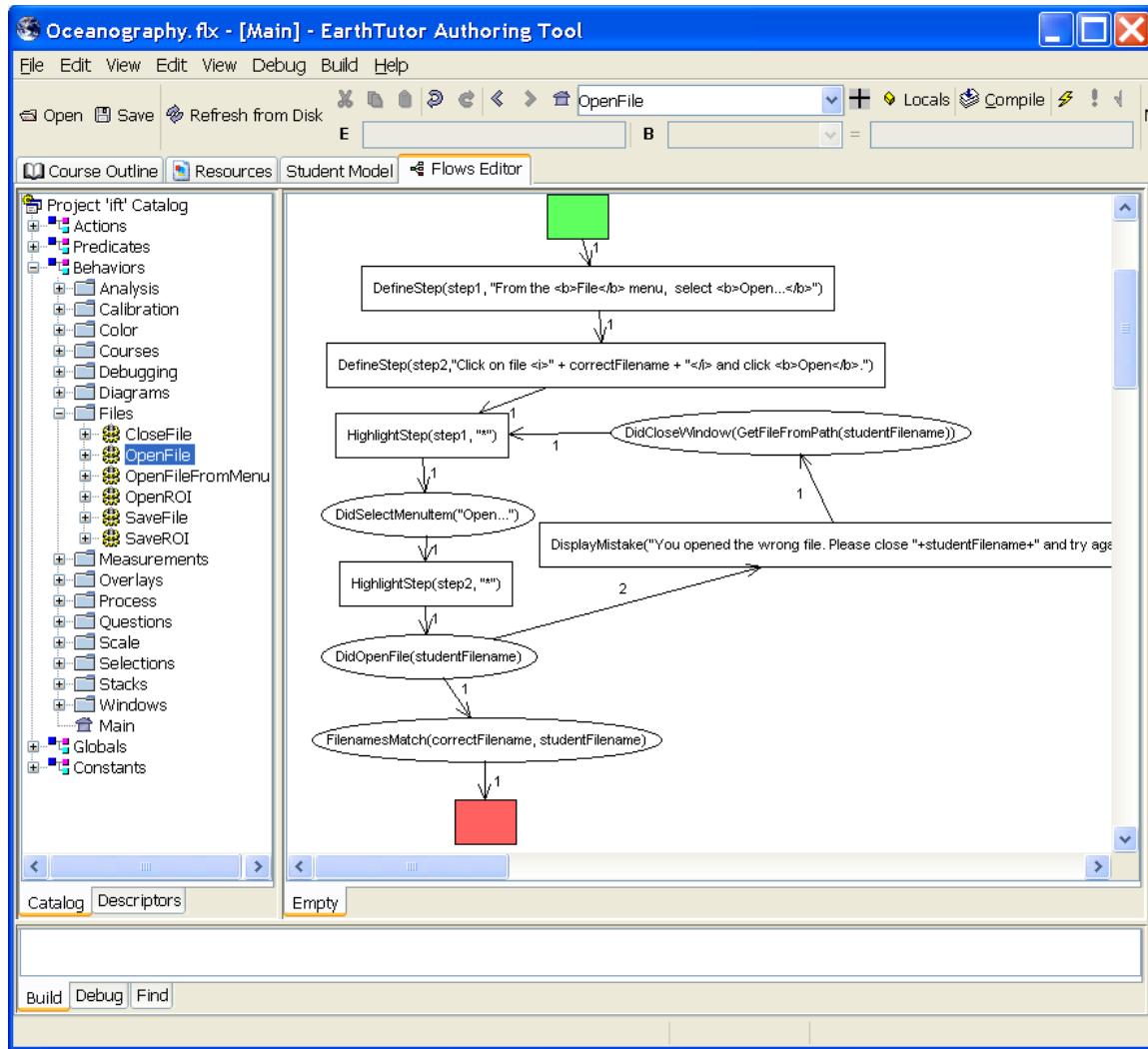


Figure 11. Flowchart behavior templates are created in EarthTutor's expert authoring interface.

Figure 12 shows a novice author creating a card for an exercise and embedding previously created behavior templates, including the one for opening a file using the menu. Here, the author has written introductory text and selected two behavior templates, including the one shown above for opening a file. The author has instantiated these templates using simple form-based graphical user interfaces (GUIs). When this card is shown to the student, the templates will be replaced by the GUI that shows the steps for opening a file, and student activity will be monitored as specified in the flowchart above.

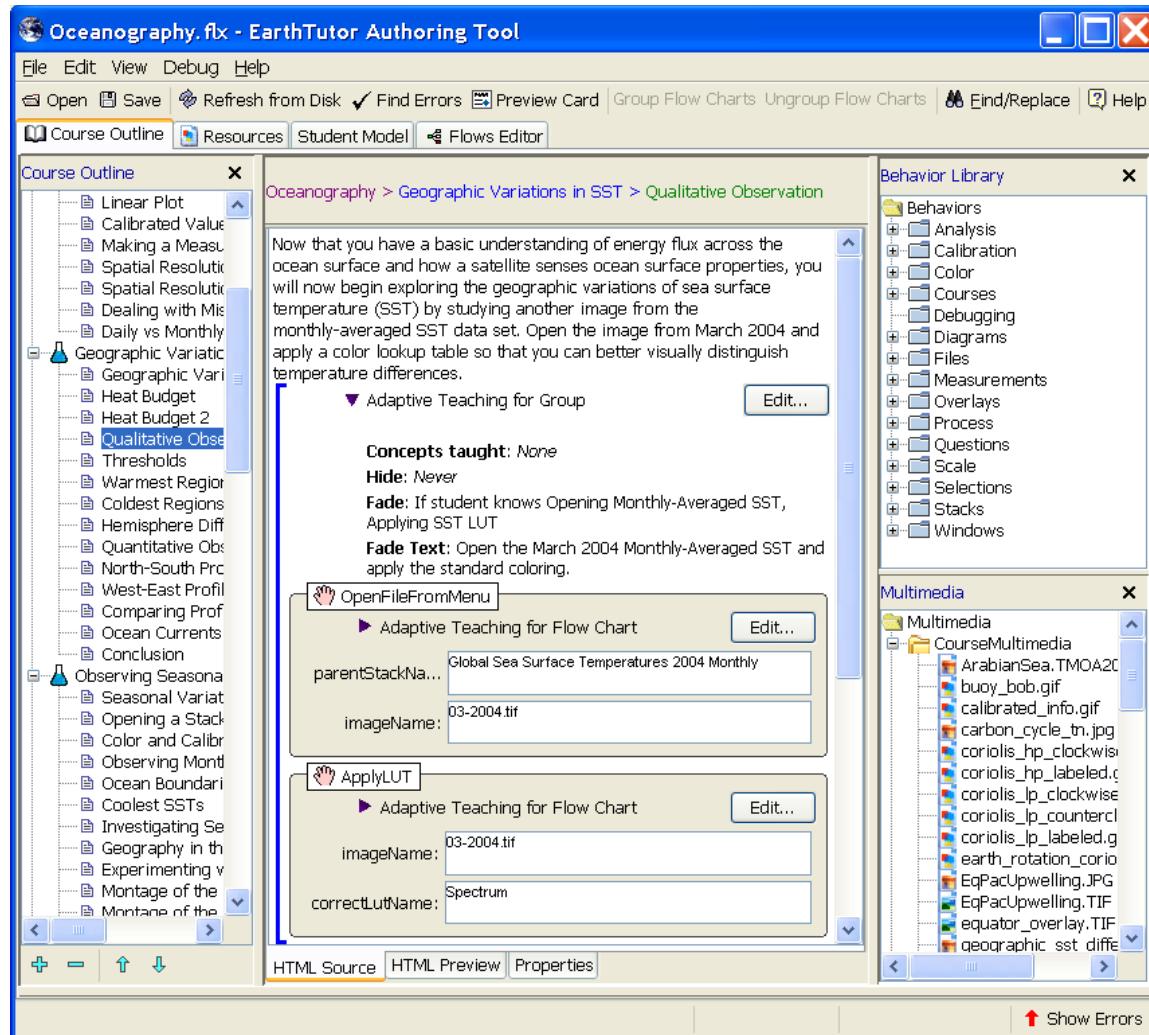


Figure 12. HTML exercise cards are created in the novice authoring interface by instantiating behavior templates.

We are using this tiered authoring approach for an ITS being developed to train Navy Information Technology (IT) support staff (ITADS). This is a simulation-based ITS designed to provide hands-on experience with troubleshooting skills and maintenance procedures. The knowledge required for automated assessment of performance—especially for troubleshooting exercises—requires complex modeling and will be constructed by developers or very advanced authors. Once the assessment knowledge has been modeled, less advanced authors will create scenarios that reference this model and tie to frozen sets of virtual machines supporting the simulation. Using simple form-based editors, novice authors can edit student-visible text associated with the scenario and with the model-linked coaching; they can also copy and adapt expert-developed scenarios. The ITS is also designed to use Socratic dialogues as a pedagogical strategy for coaching students. We take a tiered approach to authoring these dialogues as well. Advanced authors use a dialogue authoring tool to create a variety of dialogue structures. Novice authors copy and modify dialogues using form-based editors.

Layered Authoring

Tiering primarily addresses the provision of different authoring tools for different kinds of content most efficiently created by different kinds of authors. The prototypical approach of templating allows a higher volume of content to be generated more quickly by lower-skilled authors, guided and constrained by patterns established by higher-skill (and more expensive) authors. Layering, in contrast, focuses on picking apart a single kind of content (or at least a single view of content that may be composed from related elements) into pieces that have different scopes of applicability or levels of generality. The object is to achieve more reuse of authored content.

Our prototypical example of layering comes from a system that entwines *simulation*, *assessment*, and potential tutor *interventions* into a composite authoring view of linked content. Our Medical Emergency Team Tutored Learning Environment (METTLE) ITS teaches diagnosis, emergency response, and task-specific coordination appropriate for responding to chemical, biological, and radiological attacks. The target trainee is an emergency room physician. In a scenario based on an anthrax attack, the doctor is coached through an initial diagnostic session with a mystery patient in their emergency room (ER). The web-based system supports text-based diagnostic interviewing, media-based physical examination, and form-based ordering of tests and treatments. A cast of other characters can be consulted or may intervene during the scenario, including an ER nurse, a hospital administrator, and staff members at ERs of other nearby hospitals. The tutor provides proactive and reactive hints and feedback, and can also carry out extended Socratic dialogues to review diagnostic logic.

METTLE adopts a theater metaphor in which an exercise scenario is viewed as a sort of dynamic play. A METTLE scenario has a cast, each member of which is assigned a set of behaviors that we think of as “lines” in a nonlinear “script,” to be used when triggered by student activity or other scenario events. These behaviors can vary across the scenes of the scenario and based on the state of the character. Lines, then, can be assigned by role, scenario, scene, or state, (or, in the most flexible case, based on some combination of those factors). Lines can include (1) a cue (trigger conditions), (2) a response (the character’s scripted actions), (3) side-effects on scenario or character state, and (4) contextual tutor evaluations and comments (including hints, prompts, and feedback).

METTLE allows for composition of scripts and even individual script lines from different sources. For instance, a set of default behaviors can be defined that apply to any character in any situation (e.g., how to handle greetings, farewells, and small talk), while a more specific set of behaviors can be defined for some particular class of simulated characters (e.g., how any character assigned the “patient” role should respond to diagnostic questions). For the patient role, we defined a basic set of several hundred default script lines, providing a reusable set of named rules with cues and “normal” responses covering many standard diagnostic interview questions, examination actions, diagnostic tests, and so on.

When scripting any particular patient for any particular scenario, a subset of these default rules can be extended with situationally important responses, state changes, and tutoring. For instance, a patient with anthrax really only differs from a normal healthy adult on a small set of key diagnostic indicators. Authors can compose scenario-appropriate diagnostic question/answer script lines by taking the trigger from the role-general form of the behavior (the question stays the same), while overriding the response to fit the scenario (the answer is tuned to fit the results that would be expected for an anthrax patient). Entirely new rules can be added for behaviors that only make sense in the context of the scenario (or some scene or character state). For instance, if an important aspect of the case is how the patient got the disease, then a back-story can be introduced with a set of custom question and answer behaviors bearing on their recent activities, who they were with, and how those people are faring.

Consider a pair of example behaviors used in the anthrax scenario. First, there is a standard diagnostic interview question—appropriate in cases where the underlying issue might be an infectious disease—that checks if anyone else the patient knows is suffering from a similar problem. In the generic patient script there is a line named **Complaint-Others** that is specified with cues such as “*Do you know anyone else with the same symptoms?*” and a default answer of “No.” In the anthrax scenario, this is an important question and so additions and modifications are layered onto the default behavior. For starters, its answer is overridden so that the patient says: “*Yeah, my cousin John has come down with some fluey thing since we last saw each other. My wife says his wife took him to Memorial Hospital today.*” The triggering of this behavior is tied to a curriculum point labeled **ED-Diagnosis-Infection** and in the anthrax scenario a proactive prompt is associated to be used by the tutor if this behavior has not been triggered 5 minutes into the scenario: “*You might consider asking whether Ryan knows anyone else who has what he has.*” Our second example is a totally new script line introduced for this scenario. Once it is revealed that the patient’s cousin is also sick, there should be a follow-up line of questioning about the cousin. Accordingly, this scenario introduces a new line named **Others-Cousin-When** with cues that include “*When did you last see your cousin?*” eliciting the answer “*We went to a basketball game together with another friend of mine maybe 5 or 6 days ago.*”

These examples illustrate composition of aggregate scripts from behaviors defined at different layers such as for generic characters, generic patients, and some particular patient in a scenario. They also illustrate composition of individual script lines from fragments defined at different layers, such as a question/answer behavior defined for generic patients being overridden with an answer appropriate to a particular patient and associated tutoring interventions.

METTLE behaviors are composed from an extensible application-specific rule condition/action language. Extensions to that language can be viewed as an expert level authoring tier similar to EarthTutor’s advanced authoring of behavior templates. In addition, it might turn out that different tools are appropriate for different layers, or that different classes of authors are best suited to providing different layers of content. Nonetheless, when it comes to building up behaviors in layers, the primary issue is not division of labor, but rather content reusability—across exercises, courses, and possibly even domains.

Bootstrapped Assessment

Bootstrapped authoring, as applied to automated assessment, is an incremental development process where the cost of authoring is reduced with successive spirals or releases. Starting with example-based scenario-specific content and training mechanisms, authors incrementally generalize to create successively more reusable components. Each iteration offers cost savings over the last, coupled with wider reusability for the next.

Before proceeding to examples, we explicate what we mean by generalized assessment mechanisms in the context of a situated tutor. A common tradeoff in designing automated assessment is the choice between an example-based or model-based approach. Example-based assessment makes inferences from the case-specific conditions that apply in a particular scenario, without regard for how the same concepts would appear or be assessed in other scenarios. Because example-based assessment mechanisms can be essentially hard-coded with unique knowledge associated with a specific scenario, learner, or context, they are often easy to rapidly prototype. This can be very productive for the early stages of development when requirements are still being refined. However, as the number of scenarios grows, the example-based approach must be essentially replicated for each new scenario.

In contrast, a model-based approach seeks to capture more general knowledge that reduces the cost of authoring new scenarios. In the broadest sense, an assessment model attempts to represent knowledge,

skills and aptitudes (KSAs) and how they're applied in scenarios, without relying on unique scenario-specific knowledge. In practice, there are numerous approaches to model-based assessment, ranging from those that represent recurring but recognizable constraints on good performance, to those that aim to represent a comprehensive space of possible actions together with the underlying cognitive states that produce those actions. Regardless of the precise formulation, we emphasize the goal of *scenario-independence*. On one hand, the effort required to achieve scenario-independent assessment doesn't easily scale down to the early development stages where prototyping is useful. So in the short term of initial prototyping, a purely model-based approach is inherently more costly and time-consuming to implement than a purely example-based approach. However, in the longer term, a generalized assessment model reaps authoring cost benefits precisely because of the scenario-independence. A generalized model can also theoretically be abstracted further, to shed the specific constraints of a given simulation or exercise environment, and yield cost savings for transitions to other platforms.

Given the practical benefits of example-based methods in the short run and model-based methods in the long run, the bootstrapping approach seeks a transition from the former to the latter in the course of assessment authoring. This combines the expedient of an example-based approach for early development, with the future authoring benefits and cost savings associated with a generalized model. The concept of bootstrapped content authoring can be applied over successive development spirals of a scenario-based ITS, in tandem with expansions in either or both the collection of scenarios or the core ITS assessment capabilities.

This bootstrapping approach was employed in the development of a game-based trainer for the Army's US Military Academy (USMA) at West Point, called Intelligent Game-based Evaluation and Review (InGEAR). InGEAR is integrated with a tactical decision-making game called Follow Me, which is used at West Point to teach small unit leader tactics in dynamic, experiential scenarios. The project objective was to extend the reach of instructors and allow self-directed learning for cadets using the game environment. Before InGEAR was developed, Follow Me was used entirely with facilitated classroom learning, where all performance assessment and feedback in exercises was the province of human instructors. The USMA instructional staff designed an existing set of scenarios to exercise tactical concepts with varying degrees of difficulty, and assessed cadets' performance by applying accumulated knowledge of scenario dynamics. For InGEAR, this existing scenario knowledge provided an excellent baseline for a rapid prototyping effort in the first spiral of development. Example-based assessments were developed within 4 months of the project start, following the lead of established instructional knowledge.

One of the benefits of rapid prototyping in this manner is that it produces a useable training capability early on. However, with InGEAR the objective was to deliver scenario-independent mechanisms that could assess the same tactical concepts when relevant in future scenarios to be created or modified by the USMA instructional staff after the InGEAR development effort. The combination of short-term prototyping goals and long-term project goals motivated a bootstrapping evolution from an initial set of example-based assessments to an eventual set of generalized scenario-independent assessments using a constraint-based model.

An example of this evolution involves the assessment of cover and concealment in tactical movement. Initially with existing Follow Me scenarios used at West Point, instructors were so intimately familiar with the terrain and enemy positions that they could immediately point to good and bad areas of cover and concealment. Following this lead, the initial example-based assessments in the first spiral used scenario-specific annotations to score areas of terrain, applying a figure of merit for the quality of cover and concealment in significant areas. This was easy to develop quickly, and it provided a sample working assessment to review with instructors (along with automated feedback and other capabilities). It also served as an effective primer for the development team to quickly gain an understanding of the domain, which facilitated the ongoing collaboration with both the USMA staff and the developer of Follow Me.

However, this example-driven approach could not be easily extended to future scenarios, so the next spiral required a more general model-based approach to assessing cover and concealment.

In order to develop a scenario-independent assessment for cover and concealment, the methodology was to review existing scenarios where the tactical principles were applied, and to abstract the key concepts across settings. From that, a mechanism could be constructed to reason about the merits of a tactical position with respect to those concepts, in any given scenario. The key concepts in this case involve visibilities in relation to actual or likely enemy positions, and visibilities in specific terrain (e.g., the inherent level of exposure on a ridgeline versus a wooded area). For this application, the game environment already dynamically calculates visibilities between units and between terrain positions. The screenshot in Figure 13 shows an example of terrain visibility in the game (represented as a pixelated overlay) from the position of a particular machine gun unit (also shown with its sector of fire as a wedge shaped graphic).



Figure 13. The Follow Me game shows machine gun section visibilities and sectors of fire.

During an exercise, instances of detection by enemy units trigger game notifications, contributing to half of the generalized assessment for cover and concealment. However, it is more complex to implement a real-time assessment of the quality of a position in terms of terrain exposure. To support such assessment, we constructed an authoring utility to pre-process the terrain database for any given scenario by generating exposure scores for all positions (represented as terrain tiles). These scores can then be used during execution for real-time assessment, without requiring heavy processing during the exercise and without requiring explicit manual instructor annotation of the terrain in authoring. The resulting generalized assessment for cover and concealment was scenario-independent, with minimal requirements on authors seeking to activate this assessment for a new scenario. From a methodological standpoint, the implementation benefited from the earlier knowledge acquired with the example-based implementation, which accelerated the development of the subsequent more general mechanism. As a further benefit, the general assessment's performance could be compared with the earlier example-based versions as well.

For each scenario-independent assessment implemented for InGEAR, the final step to support authoring was to produce a specification for the parameters required to configure and apply the assessment mechanism in a scenario. In some cases, the parameters are thresholds for time, distances, survivability, or other factors that instructors determine will delineate performance standards (such as pass/fail). In other cases, the parameters involve a simple specification of a game artifact, such as an objective area to be secured as part of a tactical task.

Recommendations and Future Research

The three approaches discussed in this chapter, tiering, layering, and bootstrapping, hold promise for addressing the trade-off of power vs. usability in the design of authoring tools, while enabling cost-savings through content reuse and restructuring. Further research is required to build such tools and validate them for a variety of ITSs. The Generalized Intelligent Framework for Tutoring (GIFT) can facilitate this research by providing a unified framework for collaboration on this research.

GIFT provides a decomposition of typical ITS functionality that aligns well with a range of applications and capabilities. For instance, the architecture presented by Ragusa, Hoffman, and Leonard (2013) has many broad correspondences to the architecture of the ITADS system mentioned earlier: both separate management/monitor functions from the tutor user interface, which is separate from any simulation/game modules (which are linked to the ITS through an interface module); both have user management and learning management modules; and both have domain, learner, and pedagogy modules.

Domain knowledge—specifically performance assessment rules—can be specified in the GIFT framework within extensible markup language (XML) domain knowledge files (DKFs). GIFT provides a Domain Knowledge File Authoring Tool (DAT), an XML editing tool for creating and editing these rules. The DKF—and its associated DAT—provide a means to define *assessments* and *state transitions*. **Assessments** use a hierarchy of *tasks*, *concepts*, and *conditions* to cover *runtime performance* assessment (during exercises) and *scoring rules* (aggregate after-exercise scores). **State transitions** itemize *changes* in learner state that are of interest (including, of course, assessed performance states), each with a list of *strategies* the tutor might use to respond to those changes.

Within GIFT’s general module breakdown and domain modeling framework, we see several possible extensions that might support tiering, layering, and bootstrapping.

An obvious way to support tiered authoring within this framework is to allow parameterized rules and create a GUI-based authoring tool in addition to the DAT for novice authors to instantiate parameters. Another useful capability would be to support multiple simultaneous authors so that the task of rule creation can be distributed more fluidly. With these changes, expert authors could create complex logic while novice authors could create simpler rules. This capability should be supported by associated integration and testing tools for the overall set of rules. A more advanced approach might be to provide the capability to create flowcharts representing branching sequences of assessments and state transitions (e.g., to represent procedural tasks). An expert tier authoring tool could be developed for creating such flowcharts as a part of a DKF specification, while a novice tier authoring tool supported selection and instantiation of templates.

Layered authoring, as exemplified in METTLE, could also be introduced into the GIFT framework. One challenge here is our example’s relatively tight coupling between simulation/game construction, assessment authoring, and tutor intervention specification. However, if it is most natural for a scenario-focused author to think about exercise behavior, performance evaluation, and coaching in tandem then authoring tools should provide a view that couples those structures, even though an underlying

architecture might divide the simulation/game from the assessment engine from the tutor utterances. At the same time, the tools should provide a view that helps authors understand the contextually composited form of a behavior or rule, even though it combines new and reused pieces from different scopes. Again, this view should be available irrespective of how the generic underlying ITS architecture wants to divide up the included, reused, and overridden bits of knowledge.

Bootstrapped assessment authoring may also be facilitated with the GIFT framework, by adding structure for regression testing, to be integrated with the analysis testbed methodology. Naturally if assessment mechanisms will undergo an evolution as they are incrementally generalized, then some form of regression testing is desirable to verify that the assessment results from a generalized mechanism match those produced from earlier example-based assessments in a battery of specific scenarios. The GIFT framework may be an effective place to introduce such testing artifacts, because its domain module is designed to consume assessment outputs from an instrumented exercise environment, while being abstracted from the internals of the implementation in the environment. This inherently supports the abstraction between the GIFT domain module and pedagogical module. This same abstraction is relevant to a potential additional function for the GIFT analysis testbed methodology, which seeks to refine and validate learning outcomes in different conditions, such as an authored tutor versus traditional classroom learning. This comparison methodology would be useful for validating an evolving assessment approach developed in a bootstrapping fashion—to compare an initial baseline of example-based assessment mechanisms to subsequent more generalized iterations or spirals

References

- ADL (2009). SCORM® 2004 4th Edition Content Aggregation Model (CAM) Version 1.1. *Advanced Distributed Learning (ADL)*. Retrieved from: <http://www.adlnet.gov/scorm/scorm-2004-4th/>.
- Aleven, V., McLaren, B., Sewall, J. & Koedinger, K. (2006). The cognitive tutor authoring tools (CTAT): Preliminary evaluation of efficiency gains. *Intelligent Tutoring Systems Lecture Notes in Computer Science*, Vol. 4053, 61-70.
- Baker, R., Corbett, A. & Aleven, V. (2008). More accurate student modeling through contextual estimation of slip and guess probabilities in Bayesian knowledge tracing. *Intelligent Tutoring Systems Lecture Notes in Computer Science*, Vol. 5091, 406-415.
- Barnes, T. & Stamper, J. (2008). Toward automatic hint generation for logic proof tutoring using historical student data. *Intelligent Tutoring Systems Lecture Notes in Computer Science*, Vol. 5091, 373-382.
- de Jong, T., van Joolingen, W.R., Swaak, J., Veermans K., Limbach R., King S., and Gureghian D. (1998). Combining human and machine expertise for self-directed learning in simulation-based discovery environments. *Journal of Computer Assisted Learning*, 14(3), 235-246.
- Hsieh, P. Y., Halff, H. M. & Redfield, C. L. (1999). Four easy pieces: Development systems for knowledge-based generative instruction. *International Journal of Artificial Intelligence in Education (IJAIED)*, 10, 1-45.
- Kumar, R., Roy, M. E., Roberts, R. B. & Makhoul, J. I. (2014). Towards Automatically Building Tutor Models Using Multiple Behavior Demonstrations. *Intelligent Tutoring Systems Lecture Notes in Computer Science*, Vol. 8474, 535-544.
- MacLellan, C. J., Koedinger, K. R. & Matsuda, N. (2014). Authoring Tutors with SimStudent: An Evaluation of Efficiency and Model Quality. *Intelligent Tutoring Systems Lecture Notes in Computer Science*, Vol. 8474, 551-560.
- Major, N., Ainsworth, S., and Wood, D. (1997). REDEEM: Exploiting symbiosis between psychology and authoring environments. *International J. of Artificial Intelligence in Education*. 8(3-4), 317-340.
- Murray, T. (1999). Authoring intelligent tutoring systems: An analysis of the state of the art. *International Journal of Artificial Intelligence in Education (IJAIED)*, 10, 98-129.
- Nye, B. D., Rahman, M. F., Yng, M., Hays, P., Cai, Z., Graesser, A. & Hu, X. (2014). A tutoring page markup suite for integrating shareable knowledge objects (SKO) with HTML. *Proceedings from Intelligent Tutoring Systems (ITS) 2014 Workshop on Authoring Tools*.
- Qiu, L., and Riesbeck, C. (2005). The design for authoring and deploying web-based interactive learning environments. *World Conf. on Educational Multimedia, Hypermedia and Telecommunications*.

- Schatz, S., Oakes, C., Folsom-Kovarik, J. T. & Dolletski-Lazar, R. (2012). ITS + SBT: A review of operational situated tutors. *Military Psychology* 24(2), 166-193.
- Shute, V. J. (1993). A macroadaptive approach to tutoring. *Journal of Artificial Intelligence in Education*, 4(1), 61-93.

CHAPTER 28 Expanding Authoring Tools to Support Psychomotor Training Beyond the Desktop

Robert A. Sotilare, Scott J. Ososky, and Michael Boyce
US Army Research Laboratory

Introduction

Today, intelligent tutoring systems (ITSs) are generally authored to support desktop training applications with the most common domains being mathematics, computer programming, and physics. The success of using game-based platforms (e.g., Virtual BattleSpace 3 and VMedic) to train the cognitive aspects of military tasks (e.g., problem solving and decision making for land navigation, force-on-force battle tactics, and combat casualty care) have also demonstrated the efficacy of games as desktop training platforms when combined with measures of success and sufficient feedback to the learner.

In recent years, implementations of game-based tutors (Goldberg, Sotilare, Brawner & Holden, 2012) using the Generalized Intelligent Framework for Tutoring (GIFT; Sotilare, Brawner, Goldberg & Holden, 2012; Sotilare, Goldberg, Brawner & Holden, 2012) have demonstrated adaptive tutoring for desktop training applications similar to those shown in Figure 14. Measures of tutor-learner interactions shown in Figure 14 may or may not be available during psychomotor tasks being trained in the operational environment (e.g., embedded training or training in the wild). A goal is to integrate GIFT with more dynamic virtual simulations to support more natural learner interaction associated with the psychomotor elements of training tasks and thereby promote higher transfer of skills to the operational environment. This chapter begins to explore how authoring systems might be enhanced to support tutoring beyond the desktop for more dynamic physical tasks.



Figure 14. Prototypical elements of a desktop tutor-user interface

As their name suggests, adaptive training systems (e.g., ITSs) offer more flexibility during instruction. Instruction is tailored to the needs and preferences of individual learners. Given the variability of learner attributes across the general population, this creates a greater demand for domain content authoring to support tailored training experiences. Finding efficient methods to create new content and to reuse existing content (e.g., training content in existing training simulations) is a critical element in making adaptive training affordable and ubiquitous (Sottilare, 2015). Tools and methods are needed to automate large portions of the authoring process. Before we can automate the authoring process, we first need to define the authoring process for domain modeling and then examine what is unique about authoring for psychomotor domains.

Expert models (sometimes called ideal student models), scenario generation, content curation (search, retrieval, and selection), and learner assessment are all candidates for automating authoring processes. In order to operationalize automated authoring processes for psychomotor tasks, first we need to represent the dimensions of and then define measures for those tasks.

Modes of Psychomotor Tasks and their Impact on Authoring

In examining modes of dynamic interaction, we begin by modeling the type and degree of physical interaction. The degree of physical interaction in training as compared to how it is performed in the operational environment may impact transfer or the degree to which knowledge and skills developed in training are used in the operational environment. The authoring processes for all training environments include the development of the following:

- instructional goals, objectives, and concepts to be learned
- directed graphs to represent paths through the training material represented by concepts, assessments, and instructional decisions based on learning theory

We have defined four levels of physical interaction in support of adaptive training: static, limited dynamic, enhanced dynamic, and full dynamic. Each is discussed in terms of its ability to support training in the psychomotor domain and its impact on the authoring process.

Static Tutoring Mode for Low Dynamic, High Cognitive Tasks

Static training environments (e.g., desktop computer training; see Figure 14) allow the learner to train for primarily cognitive tasks with little dynamic interaction. Desktop environments are unsuitable for training purely physical tasks, but may be used to reinforce knowledge acquisition during the rules, examples, and recall quadrants defined by component display theory (CDT; Merrill, Reiser, Ranney, and Trafton, 1992).

Since the training tasks associated with desktop environments are primarily cognitive, the authoring process is primarily focused on delivering content to facilitate decision making and problem solving in the form of scenarios or graded problem sets (e.g., easy, moderate, difficult). There is much less focus on capturing any physical learner data or measures other than those needed to classify learner performance, engagement, cognitive load, and emotional states. These states are used to drive instructional decisions by the ITS.

This mode is most closely aligned with authoring processes common to traditional ITSs to support tutoring in mathematics, physics, and computer programming. No assessments are required to compare and contrast detailed physical movements of the learner to an expert model. Cognitive task analyses may

be required to develop a cognitive model to assess and compare the learner's decision-making and problem-solving processes relative to those of an expert.

Limited Dynamic Tutoring Mode

Limited dynamic tasks allow for full gestures and limited motion in a restricted area determined by the range of the sensors. Movement and tracking of the learner from standing positions to kneeling, sitting, or supine positions is supported so the range of physical tasks is broader than in static tasks. A prototype was developed in 2014 by SRI for the US Army Research Laboratory to support limited dynamic training (Figure 15) through multimodal sensing and tailoring of instruction driven by GIFT. The sensor suite (hardware and algorithms) included a Microsoft Kinect to support gesture and pose estimation, a high-resolution web camera to support assessment of emotional states based on facial markers and gaze estimation, and a microphone to provide speech interaction and support stress evaluation via tonal analysis.



Figure 15. Prototype for a limited dynamic training environment

Limited dynamic environments support hybrid (cognitive, affective, psychomotor) tasks where a larger degree of interaction with the training environment and other learners is critical to learning, retention, and transfer to the operational environment. Decision-making and problem-solving tasks may be taught easily in a limited dynamic mode along with tasks requiring physical orientation (e.g., land navigation), but certain aspects of the environment are difficult to reproduce (e.g., running over uneven terrain). Small unit training scenarios may be possible by reproducing the individual training suites in Figure 15 and combining them in a shared synthetic environment.

The impact on authoring is the assessment of physical actions, which may include tracking of fine motor movements for some tasks (e.g., interaction with equipment). The ability of the ITS to track, assess, and respond to learner movements becomes critical to supporting training in a limited dynamic mode, and authoring in this mode is correspondingly more complex than in the static mode. This difficulty increases dramatically for team-level tasks where there is a high degree of interdependence in pursuing goals. In the team scenario, models for team processes (e.g., coordination, communication, and leadership) and team states related to performance and learning must be developed and then assessed in real time during training.

Enhanced Dynamic Tutoring Mode

Enhanced dynamic environments support tasks where freedom of movement and a high degree of interaction with other learners are critical to learning, retention, and transfer to the operational environment. Building clearing and other team-based tasks may be taught easily in an enhanced dynamic mode. The impact of this mode on the authoring process is similar to the limited dynamic mode, but more complex based on the higher degree of movement in the training environment (e.g., live, augmented reality, or mixed reality). This requires sensors with greater ranges or networks of sensors. As with the limited dynamic mode, authoring is complicated in team-based scenarios where multiple learners must be tracked over wider ranges in instrumented spaces. Team processes and states must also be modeled in this mode.

Full Dynamic Tasks in the Wild

Full dynamic mode transfers tutoring to the operational environments and could also be called embedded training or training in the wild. Tutoring would go with the learner wherever the learner goes. Full dynamic mode is critical to support tasks where a very high degree of freedom of movement and a high degree of interaction with other learners are critical to learning, retention, and transfer to the operational environment.

It is anticipated that psychomotor and social tasks may be best taught in full dynamic mode or an environment more closely resembling the operational environment. Research has shown that retrieval of learned information is better when the original learning context is reinstated during task performance and that contextual dependencies also extend to perceptual-motor behavior (Ruitenberg, De Kleine, Van der Lubbe, Verwey, and Abrahamse, 2012). This supports the notion that a misalignment between physical dynamics in training tasks will slow transfer of psychomotor skills during operations, and that a better alignment of the physical aspects of training tasks with how they will be performed on the job will result in more efficient transfer of motor skills.

Authoring is complicated in this mode as sensor-based assessments of motor movements are currently limited to location. Sensor suites will need to be developed to support more detailed assessments of position, location, orientation, and other physical states.

Measuring Learner Performance in the Psychomotor Domain

Sometimes called the doing or action domain, tasks in the psychomotor domain are associated with physical tasks (e.g., marksmanship and sports like golf, baseball, and soccer) or manipulation of a tangible interface (e.g., driving or piloting vehicles and remotely piloting a vehicle), which may include physical movement, coordination, and the use of the motor skills along with cognitive elements (e.g.,

decision making and problem solving). Simpson's psychomotor taxonomy (1972, Figure 16) lists seven levels of psychomotor skill development from perception (low) to origination (high).

- **Origination (high):** creating new movement patterns to fit a particular situation
- **Adaptation:** skills well developed and can be modified to fit special requirements
- **Complex Overt Response:** skillful performance of complex movements
- **Mechanism:** learned responses have become habitual
- **Response:** early stages in learning complex skill; imitation; trial & error
- **Set:** readiness to act
- **Perception (low):** ability to use sensory cues to guide motor activity

Figure 16. Simpson's (1972) psychomotor domain

Psychomotor tasks encompass physical movement, coordination, and the use of the motor-skill areas. The development of these skills requires practice and is measured in terms of speed, precision, distance, procedures, or techniques in execution. While this domain is well represented in military training, research is needed to build adaptiveness into these training systems and thereby optimize deep learning. A goal of this research is to reduce the *time to competency* to allow time for over-training and deeper learning experiences which transfer more efficiently to the operational environment.

For each type of physical task and associated training scenario, we can represent measures of skill development in a similar hierarchical fashion. In its simplest form, training is about asking learners to perform a task (with associated goals) under specific conditions (environment) to an established set of standards (measures), and finally, provide feedback about their performance to support improved learning and potential for greater performance in the future. Associated with each task is a set of required skills. By way of example, let's examine a land navigation task:

- **Task:** plan and navigate a route from point A (east) to point B (west)
- **Associated goals:** determine one's position on a map at 30 minute intervals as one navigates from one position (point A) to another (point B)
- **Measures:** note the variance between actual position and marked position on the map at each 30-minute interval and the time to complete course
- **Conditions:** consists of a single individual learner wearing a global positioning system (GPS) tracker walking on hilly, forested terrain with restricted visibility; no watch or compass is available
- **Performance Standard:** navigate course in 3 hours or less
- **Physical Skills Required:** demonstrate endurance, speed, and balance for navigating over uneven terrain
- **Cognitive Skills Required:** demonstrate map reading, assessment of position based on landmarks, and the position of the sun

We examine a land navigation task in terms of physical behaviors and cognitive skills starting with low skills and working toward examples of high skills. In terms of Simpson's psychomotor domain, we also observe specific goals and measures that might be required to support interactive tutoring beyond the desktop (e.g., in an operational environment as embedded training or in the wild as part of a distributed learning application). Across all of the levels of psychomotor development, we have identified the need to capture performance-related behaviors (e.g., observing and organizing), but we also discuss the challenges and potential impact on the authoring process.

An ITS must be able to acquire data about the learner's choices and use these data to assess progress toward goals as measured against an expert model or other standard established for the task under training. The author must be able to identify and acquire key behavioral measures at each level of the psychomotor taxonomy. The author must also be aware of and manage cognitive load, and specifically, working memory during instruction (Sweller, Van Merriënboer & Paas, 1998). Sottilare & Goldberg (2012) suggest that comprehensive modeling of the learner during instruction is key to successfully managing cognitive load by either injecting difficulties during tutoring to engage the learner or reducing difficulty so the learner can realize success. ITSs will require the ability to distinguish one psychomotor level from another to determine progress of the learner based on either expected results based on past performance, on organizational standards, or in comparison to the expert behaviors described in the ITS expert model.

Perception

Perception is the “organization, identification, and interpretation of sensory information in order to represent and understand the environment” (Schacter, 2011). In our example task, land navigation, the learner is taking in information about the terrain and observing the position of the sun, and using this information to estimate current position and choose future actions (e.g., routes). Perception behaviors include, but are not limited to choosing, describing, detecting, and differentiating (Simpson, 1972) based on sensory input and judgment. While we may not be able to directly observe the “interpretation of sensory information,” we can track the resulting behaviors stemming from decision making (e.g., learner moves off to the left toward the road below). Greater insight may also be teased out through reflect dialogue with the ITS.

Set

Sometimes called mindsets or dispositions, *set* includes mental, physical, and emotional dispositions that predetermine a person's response to current conditions (Simpson, 1972). In the case of our land navigation task, the learner is assumed to have prerequisite skills (e.g., map reading), which drive reactions to current conditions. Since these cognitive skills are needed to successfully complete the task, these are part of the mental set. A learner's motivation and enthusiasm to complete the task is part of the emotional set, and finally, the physical set might include a readiness to complete the task based on sufficient sleep and nutrition. Each of these dispositions can either enable or inhibit the learner's ability to perform. Set behaviors include starting, displaying, reacting, responding, and volunteering (Simpson, 1972). It is likely that long-term modeling of learner experiences can provide insight to the learner's mental disposition based on meeting prerequisites for the task under training. Specific behavioral measures to determine emotional disposition may include semantic and/or tonal analysis of learner responses. Finally, the physical set may be determined through query or physiological sensing.

Response

During the *response* stage of learning complex tasks, models are critical in skill development. It may be useful to the learner to observe others successfully performing the task of determining position based on the position of the sun at various times of day to determine direction. Trial and error through guided practice allows the learner to apply knowledge (e.g., heuristics) and eventually reduce errors as the learner develops enhanced mental and physical models. This assumes time is available to support discovery learning. In the case of land navigation, the learners may use the sun throughout the day to determine that they are traveling west toward point B. Over time, they will become more skilled at judging the time of day, and thereby the position of the sun and its relationship to a westerly course. Response behaviors include assembling, measuring, decomposing, manipulating, fixing, mixing, and organizing (Simpson, 1972). It is likely that measures of response will be domain-specific.

Mechanism

During the *mechanism* stage, learned responses are now habitual and physical movements can be performed with a growing degree of confidence. In our land navigation example, learners running over uneven terrain the first time would be slower and more deliberate in their movements, while learners who have practiced and habituated this skill will run more easily and with much less conscious thought. This reduces the cognitive workload during this action and allows this resource to be applied to other elements of the task. Mechanism behaviors include many of the same behaviors as in response, but are displayed at a higher level of automaticity (Simpson, 1972). Measures of mechanism will be similar to response, but the speed and accuracy of learner actions will have increased based on deliberate practice.

Complex Overt Response

During the *complex overt response* stage, the learner displays highly skillful performance of physical actions that involve complex movement patterns. At this level of proficiency, the learner does not hesitate, and is accurate and highly coordinated. They perform required actions with a minimum expenditure of energy. In our land navigation example, the ability to move easily and almost effortlessly over uneven terrain has developed to a high level of confidence and speed. Complex overt response behaviors include many of the same behaviors as in response and mechanism, but are displayed at a higher level of automaticity (Simpson, 1972). Measures of complex overt response will be similar to mechanism and response, but the speed and accuracy of learner actions will have increased based on deliberate practice.

Adaptation

In *adaptation*, the learner's skills are so well developed that the learner can change movement patterns to fit special requirements or unexpected situations. As the term adaptation suggests, the learner's behaviors include changing, altering, rearranging, reorganizing, revising, and varying movement to meet new situations that may never been encountered by the learner previously (Simpson, 1972). In our land navigation example, the learner could encounter obstacles (e.g., near vertical paths and rivers) in route to point B that may require adaptation of the more basic "moving over uneven terrain" skill. Physical pattern recognition will be needed for the ITS to recognize standards (most likely) physical actions and their variants.

Origination

During *origination*, the learner arranges, combines, composes, develops, designs, and creates. With an emphasis on creativity based on highly developed skills, the learner crafts new movement patterns to fit a particular scenario, a set of conditions, or a specific problem (Simpson, 1972). Again, physical pattern recognition will be needed for the ITS to recognize standards (most likely) physical actions and their variants, but the ITS will also need to recognize when physical behaviors have evolved to become sufficiently different to be classified as “new.”

Implications for Authors and GIFT

This chapter has examined the expansion of authoring in support of tutoring in psychomotor domains. The primary implications for authoring in the psychomotor domain are in developing mode-specific measures and sensor suites (hardware and algorithms) to support the assessment of individual motor movements, team processes, and team states. There are many opportunities for research to support these challenges. For example, research is needed to support assessments of fine motor movements at a distance. Likewise, opportunities to evaluate commercial tools (e.g., smart glasses) to provide instruction, feedback, and enhanced interaction between the learner and the environment should also be pursued.

An examination of the expansion of authoring, however, would not be complete without addressing the impact that those new methods, tools, and technologies may have on the authoring experience from a user-perspective. While automated authoring processes are a goal of ITSs, in general, it is likely that the burden of physically creating the tutor will fall to human hands for the foreseeable future. The act of creating an intelligent tutor, whether through computer programming or a guided user interface, is a process with which many potential authors (e.g., subject matter experts, training facilitators) may be unfamiliar. Developing an artificially intelligent tutor represents a new content creation activity, one for which new human mental models are required. ITS authoring shares some superficial similarities with other content creation activities such as developing a slide deck designing a web page. Though, there are aspects of tutor creation unique to ITSs (e.g., content selection based on learner model) for which new authoring processes must be defined. By extension, the nature of the psychomotor domain presents a specific set of challenges in authoring tutors for learners in dynamic environments as well as the testing and evaluation of those tutors.

Authoring for Different Interfaces

Instructional content is typically authored with PC-based software tools; courses created with those tools are usually accessed by learners using PC-based hardware (as evidenced by low dynamic trainers). More sophisticated creation tools support cross-platform compatibility to access course content on phones and tablets. However, if adaptive tutoring is to truly move beyond the desktop, then tutor authoring tools must also accommodate non-traditional interfaces, such as embedded systems, augmented reality, or haptic systems.

Recall the land navigation example. In full dynamic mode, for instance, learners are moving through an open environment. The constraints dictate that no watch or compass is available. Learners must also attend to environmental features and reference area maps. The characteristics of the land navigation task represent potential limitations for the author with respect the type of content and communication that can be delivered to the learner — it may not be practical or appropriate to add a smartphone to the learner’s physical and cognitive load. Therefore, authors must identify an appropriate modality by which instructional interventions may be communicated to the learner, as well as determine how to implement that solution with their tutor creation tools. Traditional software tutor authoring tools will need to extend

their functionality be able to create content for and interoperate with available devices that may be embedded with learners in the field. Research will also be needed to attend to the *usability* of authoring tools, concurrent with the development of new functionality.

Authoring for Different Environments

Real-world environments introduce new constraints such as noise, glare, temperature, or physical obstacles. Those environments features vie for the attention of the learner and represent constraints that authors must consider when developing adaptive tutors.

For instance, examine the interaction with the talking avatar shown in Figure 1. The author may desire to carry the avatar interaction into a dynamic environment in order to create a consistent narrative across various stages of training. In developing an *enhanced dynamic* component to the land navigation course, the author plans to use an augmented reality headset for use in daytime and nighttime scenarios, including simulated night vision capabilities. The author realizes learners' visual bandwidth may be overloaded with information from the task itself and decides to use auditory communication for instructional interventions without the visual avatar component.

Even at this stage in the design, additional decisions must be made. The author can *push* auditory messages to the learner at the point of need, but risks the learner being unable to attend to the message while moving about. Alternatively, the author can notify the learner that guidance is available (i.e., *pull*), at the risk of the learner ignoring the guidance or listening to it when it is no longer useful. Thus, as training environments increase in approximation of real-world settings, authors will need to consider how features of the training environment impact the design of the tutor.

Test and Evaluation of Psychomotor Domain Tutors

Current GUI-based tutor authoring tools leverage visual design interfaces that allow authors to preview course content (from the learner's perspective) as it is being created. This method of content creation has been referred to as *what you see is what you get* (WYSIWYG) and is the common method of content creation for documents, slide decks, and web pages. That method is somewhat inadequate for creating training in dynamic environments, which may include unique hardware interfaces and complex physical environments, described in the previous sections, respectively. Authors, however, will still require the means to review and update training material to ensure, for example, sensors and communication between the learner and tutor function as intended or that branches of the tutor are accessed based on corresponding learner models.

Suppose that the author(s) in the land navigation example decided to overlay an interactive avatar in the augmented reality display. The author, designing the course from a desktop computer, must make some decisions regarding the size, position, and transparency of the physical avatar within the learner's display. The avatar must be configured in such a way that it provides an instructional benefit to the learner, while not detracting from necessary information in the visual field. The author might reference literature to determine the configuration, go through a testing process with pilot users, or make an educated guess and hope for the best. To that end, authoring tools can support authors by embedding *best practices* into GIFT's development interfaces to allow authors to work from a series of default options.

The cornerstone of adaptive tutoring is instructional interventions and branching content based on learner data. Authors may also want to know if their interventions are triggered at the correct moments within the course, as well as trace the tutor content model to ensure that all permutations of the content are reachable under intended learner states. To test those elements of the tutor within a static environment, an author might simply need to answer a pre or post-test survey with the desired responses and examine the result.

Testing the same tutor functionality in the land navigation example by creating GPS and other physical sensor data is labor intensive and potentially impractical. Therefore, a robust set of GIFT authoring tools should also include the capability to automatically simulate sensor data in order to generate a set of learner states to test the adaptive portions of the tutor.

Finally, the value of an adaptive tutor will be diminished without mechanisms by which the effectiveness of the tutor can be evaluated. The degree to which training for psychomotor tasks is effective may be dependent upon the level of physical interaction (i.e., *mode*). Further, the data required to assess the effectiveness of the training may overlap with the data collected for the learner model. There are opportunities to minimize effort by building training effectiveness *hooks* into authoring tools, thus creating a more comprehensive solution. Enabling authoring tools in GIFT with a forward-looking perspective toward training effectiveness may also create opportunities to embed data collection tools into live (non-training) performance assessments. Such features may provide an easier path to expanding authoring tools in support of psychomotor tasks beyond training into transfer tasks and long-term skill development.

References

BIOGRAPHIES

Editors

Keith Brawner

Dr. Keith Brawner is a researcher for the Learning in Intelligent Tutoring Environments (LITE) Lab within the US Army Research Laboratory's Human Research & Engineering Directorate (ARL-HRED). He has 8 years of experience within US Army and Navy acquisition, development, and research agencies. He holds a Masters and PhD degree in computer engineering with a focus on intelligent systems and machine learning from the University of Central Florida. The foci of his current research are in machine learning, active and semi-supervised learning, realtime datastream processing, affective computing, adaptive training, and semi/fully automated user tools for adaptive training content.

Art Graesser

Professor Art Graesser is a professor in the Department of Psychology and the Institute of Intelligent Systems at the University of Memphis and is an Honorary Research Fellow at the University of Oxford. His primary research interests are in cognitive science, discourse processing, and the learning sciences. More specific interests include knowledge representation, question asking and answering, tutoring, text comprehension, inference generation, conversation, reading, memory, emotions, computational linguistics, artificial intelligence, human-computer interaction, learning technologies with animated conversational agents (such as AutoTutor and Operation ARA), and automated analyses of texts at multiple levels (such as Coh-Metrix, and Question Understanding AID (QUAID)). He served as editor of the journal *Discourse Processes* (1996–2005) and *Journal of Educational Psychology* (2009–2014). His service in professional societies includes president of the Empirical Studies of Literature, Art, and Media (1989–1992), the Society for Text and Discourse (2007–2010), the International Society for Artificial Intelligence in Education (2007–2009), and the Federation of Associations for Behavioral and Brain Sciences Foundation (2012–13). In addition to receiving major lifetime research achievements awards from the Society for Text and Discourse and University of Memphis, he received an award in 2011 from American Psychological Association on Distinguished Contributions of Applications of Psychology to Education and Training.

Xiangen Hu

Dr. Xiangen Hu is Dunavant professor in the Department of Psychology and Department of Electronic and Computer Engineering at The University of Memphis (UofM), senior researcher at the Institute for Intelligent Systems (IIS) at the UofM, and visiting professor at Central China Normal University (CCNU). Dr. Hu received his MS in applied mathematics from Huazhong University of Science and Technology, MA in social sciences, and PhD in cognitive sciences from the University of California,

Irvine. Currently, Dr. Hu is the director of the Cognitive Psychology program at the UofM, the Director of Advanced Distributed Learning (ADL) Center for Intelligent Tutoring Systems (ITS) Research & Development, and the senior researcher in the Chinese Ministry of Education's Key Laboratory of Adolescent Cyberpsychology and Behavior. Dr. Hu's primary research areas include mathematical psychology, research design and statistics, and cognitive psychology. More specific research interests include general processing tree (GPT) models, categorical data analysis, knowledge representation, computerized tutoring, and advanced distributed learning. Dr. Hu receives funding for the above research from the US National Science Foundation (NSF), US Institute for Education Sciences (IES), ADL of the US Department of Defense (DoD), US Army Medical Research Acquisition Activity (USAMRAA), ARL, US Office of Naval Research (ONR), UofM, and CCNU.

Robert Sottilare

Dr. Robert A. Sottilare leads adaptive training research at the Simulation & Training Technology Center (STTC) within ARL-HRED. The focus of his research is automated authoring, instructional management, and analysis tools and methods for intelligent tutoring systems (ITSs). His work is widely published and includes recent articles in the *Journal for Defense Modeling & Simulation*, *Cognitive Technology* and the *Educational Technology Journal & Society*. Dr. Sottilare is the co-creator of the Generalized Intelligent Framework for Tutoring (GIFT). Prior to his work in adaptive training, he was active in distributed training and learning technologies. He received his doctorate in modeling and simulation from the University of Central Florida with a focus in intelligent systems. In January 2012, he was honored as the inaugural recipient of the US Army Research Development & Engineering Command's Modeling & Simulation Lifetime Achievement Award.

Authors

Vincent Aleven

Dr. Vincent Aleven is an Associate Professor in Carnegie Mellon University's (CMU) Human-Computer Interaction Institute, and has over 20 years of experience in research and development of advanced learning technologies, such as ITSs and educational games. Major themes in his research are self-regulated learning, metacognition authoring tools, and the use of tutoring technology in ill-defined domains. Dr. Aleven and colleagues created the Cognitive Tutor Authoring Tools (CTAT), a suite of efficient, easy-to-learn, and easy-to-use authoring tools for intelligent tutoring systems (<http://ctat.pact.cs.cmu.edu>), including a new paradigm called "example-tracing tutors" that make tutor authoring 4–8 times as cost-effective. CTAT tutors have been built for a wide range of domains, including mathematics (at the elementary school, middle school, and high school level), science (chemistry, genetics), engineering (thermodynamics), language learning (Chinese, French, and English as a Second Language), and learning of intercultural competence. Dr. Aleven is a member of the Executive Committee of the Pittsburgh Science of Learning Center (PSLC), an National Science Foundation (NSF)-sponsored research center spanning CMU and the University of Pittsburgh. He is a co-founder of Carnegie Learning, Inc., a Pittsburgh-based company that markets Cognitive Tutor™ math courses. He was the program committee co-chair of the 2010 International Conference on Intelligent Tutoring Systems. He is co-editor in chief of the *International Journal of Artificial Intelligence in Education*. He has been or is principal investigator (PI) on 7 major research grants and co-PI on 10 others. He has over 200 publications to his name.

Benjamin Bell

Dr. Benjamin Bell is a principal with Aqru Research and Technology, LLC. Dr. Bell's research has addressed the authoring and efficacy of simulation for training and education across a spectrum of applications, including K-12, higher education, and military training. He has led funded research from a diverse array of sponsors, including the DoD, Federal Aviation Administration (FAA), National Air and Space Administration (NASA), and the National Baseball Hall of Fame. He has held leadership positions in the private sector, and previously served on the faculty of Teachers College, Columbia University. Dr. Bell is an associate editor for the *IEEE Transactions on Human-Machine Systems* and an assistant adjunct professor at Embry Riddle Aeronautical University. He holds a PhD from Northwestern University and Master's degrees from Embry Riddle and Drexel University. Dr. Bell is a graduate of the University of Pennsylvania.

Karissa Berkey

Karissa Berkey is currently a senior at Stetson University in DeLand, FL. She will be graduating in May 2015 with a major in psychology and a minor in education. During her time at Stetson, she has been a member of the Psi Chi International Psychology Honor Society, Student Government Association, Student Ambassadors, and Pi Beta Phi Women's Fraternity, and served as the International Admissions Assistant. She completed her Senior Project in the areas of social anxieties and interaction, self-awareness, and perception, and has since pursued a second research study in similar areas in relation to social Greek organizations. Karissa plans to attend graduate school for clinical psychology in fall 2015.

Stephen Blessing

Dr. Stephen Blessing is currently an Associate Professor of Psychology at the University of Tampa. He has over 20 years of experience in the field of ITSs, starting with developing the Demonstr8 authoring tool while an intern at Apple Computer. He worked for 5 years at Carnegie Learning, creating the cognitive models for their high school math tutors. While there he started work on their Cognitive Tutor Software Development Kit, which allowed for the rapid creation of their model-tracing tutors. Dr. Blessing has maintained his research interest in authoring tools for ITSs, co-editing a book on the topic. He has collaborated with Dr. Stephen Gilbert on the Extensible Problem-Solving Tutor (xPST), another authoring tool that aims to make tutor creation easier and more affordable. He has an interest not only in how ITSs are used in traditional classroom environments, but also how they may be used in informal learning environments as well. He is currently testing an iPad-based tutor in a children's museum.

Amy Bolton

Dr. Amy Bolton is a Program Officer at ONR. She manages several programs within the Capable Manpower Future Naval Capability. Capable Manpower is a multi-million dollar per year science and technology (S&T) program that addresses the human system integration topics of manpower, personnel, training, and human system design. Products from Dr. Bolton's programs have had success transitioning to both the Navy and Marine Corps contributing to enhanced warfighter readiness across the Naval Enterprise. Dr. Bolton's research interests include adaptive training, human behavior modeling, human system design, and Live, Virtual, and Constructive training. She holds a PhD in applied experimental and human factors psychology from the University of Central Florida. Dr. Bolton has published more than 50 technical publications including four invited book chapters. Publications were on the topics of computational modeling, training technology and methodology, cognitive performance and resilience to stress, augmented cognition, and transitioning S&T into the acquisition process.

Michael Boyce

Dr. Michael W. Boyce is a Postdoctoral Research Associate under the direction of Dr. Sottilare supporting the Learning in Intelligent Tutoring Environments (LITE) Laboratory and the Advanced Simulation Branch for the Army Research Laboratory Simulation and Training Technology Center (ARL STTC). As a part of his postdoc, Dr. Boyce has been tasked with developing an adaptive tutoring environment to help support the Augmented Reality Sandtable. His research interests involve user interaction design, human systems integration, and human performance assessment. Dr. Boyce has his doctorate from the University of Central Florida, Applied / Experimental Human Factors Psychology Program.

Zhiqiang Cai

Mr. Zhiqiang Cai is a Research Assistant Professor with IIS at UofM. He has a MS degree in computational mathematics received in 1985 from Huazhong University of Science and Technology, P. R. China. His current research interests are in algorithm design and software development for tutoring systems and natural language processing (NLP).

Joseph Cohn

Dr. Joseph Cohn is a Commander in the US Navy's aerospace experimental psychologist (AEP) community. He is currently assigned as Deputy Director to the Office of the Under Secretary of Defense for Acquisition, Technology, and Logistics' (OUSD(AT&L)) Human Performance Training and BioSystems Directorate, with oversight for both the DoD's Human Research Protection Programs and the DoD's Human Systems and Medical research portfolios. He has previously served in the ONR's Human and Bioengineered Systems Division, as a Military Deputy and Program Officer and was ONR's first Deputy Director of Research for Science, Technology, Engineering and Mathematics (DDoR STEM). Dr. Cohn also served as a program manager at the Defense Advanced Research Projects Agency (DARPA) directing basic and applied research projects that delivered cutting edge biomedical and information technology products, including deployable brain-imaging technologies, advanced brain-system interfaces, technologies that inoculate warfighters against stress, and a digital tutoring system that reduced by an order of magnitude the time required to train novices to perform at the expert level. He has co-authored over 80 publications, chaired numerous panels and workshops and been an invited speaker to national and international conferences on human systems research. He has co-edited a three-volume book series focusing on all aspects of training system development, and a single-volume book on enhancing human performance in high risk environments and is working on a book entitled *Modeling Sociocultural Influences on Decision Making*. He is a Fellow of the American Psychological Association, the Society of Military Psychologists, and Associate Fellow of the Aerospace Medical Association.

Ron Cole

Dr. Ron Cole is Principal Scientist and President of Boulder Language Technologies, Inc. He received a BA in psychology from the University of Rochester and an MA and PhD in psychology from the University of California at Riverside. He established the Center for Spoken Language Understanding (CSLU) at the Oregon Graduate Institute, where he envisioned and managed development of the CSLU toolkit, with over 32,000 installations in 136 countries. The CSLU Toolkit was used as the research, development, and runtime platform to teach vocabulary to profoundly deaf children through spoken dialogue interaction with an animated computer character. The results of this multidisciplinary research effort were featured on ABC TV's Prime Time and the NSF Home Page on 03/2001–04/2001. He also

co-founded the Center for Spoken Language Research at University of Colorado and established three successful companies. He has been principal investigator or co-PI on over \$40 million in individual investigator peer-reviewed grants from NSF, National Institutes of Health (NIH), and Department of Education. His research with Wayne Ward at Boulder Language Technologies led to development of My Science Tutor, an ITS in which children learn spoken dialogues with a virtual tutor, with learning gains equivalent to expert human tutors. During the past 20 years, he has worked diligently to stimulate and sustain international collaboration in computer science and engineering. In 1997, with Jose Fortes, he organized the NSF-sponsored Workshop on International Collaboration in Computer Science. Subsequently, he organized several NSF-sponsored workshops with Jose Fortes, Jaime Carbonnell, and others in the US, Argentina, Chile, and Mexico to promote international collaboration in computer science. Several of these workshops led to new projects, initiatives and programs. He also managed a 2-year project sponsored by the NSF and EU to survey the state of the art of the field of human language technology, which resulted in an edited volume with contributions from over 90 authors.

Mark G. Core

Mark G. Core is a research scientist at the Institute for Creative Technologies (ICT) at the University of Southern California. He specializes in artificial intelligence (AI) in education working in ill-defined domains such as negotiation, cultural awareness, and leadership. He received his PhD from the University of Rochester in 2000 under the direction of Prof. Lenhart Schubert, and was a research fellow at the University of Edinburgh working with Prof. Johanna Moore until joining ICT in 2004. He worked in the area of computational linguistics specifically natural language understanding, and discourse analysis. At the University of Edinburgh, he undertook analysis of successful human tutoring dialogues and while at ICT he has focused on analysis of learner writing. At ICT, he is also working on an evolving tutoring architecture incorporating technologies such as expert modeling, open learner modeling, experience manipulation, explainable AI, natural language understanding, and natural language generation. Recent areas of research include authoring tools for tutoring systems, and physician training including interview and diagnosis skills.

Jianmin Dai

Jianmin Dai received his PhD in system engineering from the Huazhong University of Science and Technology in China in 2006, and served as a Postdoctoral Fellow on the Writing Pal project with Danielle McNamara from 2008–2011. He is currently an Assistant Research Professor in charge of system design and computer programming for ITSs (e.g., iSTART, Writing Pal) and NLP tools (e.g., Coh-Metrix, TERA, Writing Assessment Tool) for the Science of Learning and Educational Technology Lab (SoLET) at Arizona State University (ASU). His primary interests are in research and development (R&D) for ITSs and game-based education technology. His research focus is on the application of NLP and machine learning in ITSs and game-based education systems.

Sandra Demi

Sandra Demi is a Senior Research Programmer at the Human-Computer Interaction Institute at CMU and has over 20 years' experience working in the software field as a developer, tester, and technical writer. For the last 10 years, she has been responsible for quality assurance and creating installers for the CTAT project, as well as contributing to development of the software, documentation, and supporting users. Prior to joining the project, she worked as a tester for an electronic diary used in clinical trials for the pharmaceutical industry, a developer on a nuclear-waste tracking system, a developer for a web proxy server, the lead tester for the transaction-processing engine in IBM's WebSphere application, and as a

technical writer for a suite of design-synthesis tools for electrical engineers. Her technical expertise includes test automation, software usability, and a variety of programming languages including Java, JavaScript, Python, ActionScript, C and PL/SQL. She has a MS in information science from the University of Pittsburgh.

Eric Domeshek

Dr. Domeshek is an AI Project Manager at Stottler Henke Associates, Inc. where he leads and supports projects applying AI technology to problems in training and decision support. He has worked on a wide range of ITSs and related training, education, and simulation environments spanning applications to military tactics, medical diagnosis, engineering systems management, business decision-making, and historical analysis. He is particularly interested in exploration of Socratic tutoring techniques and the development of authoring tools. He currently leads work on the authoring tools for the Intelligent Tutoring Authoring and Delivery System (ITADS) ITS in development for the US Navy. Dr. Domeshek received his PhD in computer science from Yale University, focused on case-based reasoning. For his dissertation, he developed representations of decision rationale for social situations, intended to support case retrieval; this included extensive representations of characters' relationships, traits, and motivational structures. He served as research faculty at the Georgia Institute of Technology College of Computing where he contributed to the development of a line of case-based design aids. He was also an assistant professor at Northwestern University, developing goal-based scenario training systems at the Institute for the Learning Sciences.

Hannah Freeman

Hannah Freeman, MSc, is a Policy Analyst in the Office of the Assistant Secretary of Defense (Research & Engineering)'s (OASD(R&E)) Human Performance, Training, and BioSystems Directorate. In this role, she provides strategic guidance to the DoD's Human Systems S&T Senior Executives, directly supporting the long-term investment of over \$3B in basic and applied research activities. Ms. Freeman also supported the department's S&T leadership through Reliance 21; supporting the development and implementation of policy strengthening the harmonization and efficiency of the DoD S&T joint planning and coordination process. She earned a dual BA degree in Hispanic studies and international studies (Russia and Eastern Europe) at Illinois Wesleyan University. Ms. Freeman earned her MSc in political science - conflict studies from the London School of Economics & Political Science.

Libby Gerard

Libby Gerard, EdD, is a Research Scientist in the University of California (UC), Berkeley Graduate School of Education. Her research examines how innovative learning technologies can capture student ideas and help teachers and principals use those ideas to make decisions about classroom instruction and assessment. Her recent projects explore the use of automated scoring of student written essays and student created drawings to provide guidance to students and support the teacher. She designs and leads teacher and principal professional development by using student assessment data to inform instructional customization and resource allocation. Prior to being a research scientist, she was a postdoctoral scholar researcher at UC Berkeley where she coordinated the Mentored and Online Professional Development in Science (MODELS) project, and was a fellow of the Technology Enhanced Learning in Science (TELS) Center at Mills College. She also taught preschool and elementary school in Oakland, CA, and in Alessandria, Italy. Her research is published in leading peer-reviewed journals including *Science*, *Review of Educational Research* and *Journal of Research in Science Teaching*.

Stephen Gilbert

Stephen B. Gilbert received a BSE from Princeton in 1992 and PhD from MIT in 1997. He has worked in commercial software development and run his own company. He is currently an assistant professor in the Industrial and Manufacturing Systems Engineering Department at Iowa State University, as well as Associate Director of ISU's Virtual Reality Application Center and its Graduate Program in Human Computer Interaction. His research interests focus on technology to advance cognition, including interface design, intelligent tutoring systems, and cognitive engineering. He is a member of IEEE and ACM. He is currently the PI on two contracts supporting ARL's STTC in future training technologies.

Benjamin Goldberg

Benjamin Goldberg, PhD, is a member of the LITE Lab at ARL-HRED's STTC in Orlando, FL. He has been conducting research in the M&S community for the past 5 years with a focus on adaptive learning and how to leverage AI tools and methods for adaptive computer-based instruction. Currently, he is the LITE Lab's lead scientist on instructional management research within adaptive training environments. Dr. Goldberg is a PhD graduate from the University of Central Florida in the program of M&S. Dr. Goldberg's work has been published across several well-known conferences, with recent contributions to the Human Factors and Ergonomics Society (HFES), Artificial Intelligence in Education and ITS proceedings, and the *Journal of Cognitive Technology*.

Neil Heffernan

Dr. Neil Heffernan is a Professor of Computer Science and Co-Director of the Learning Science & Technologies Program at Worcester Polytechnic Institute (WPI). For his dissertation from CMU, he built the first ITS that incorporated a model of tutorial dialogue. This system was shown to lead to higher student learning, by getting students to think more deeply about problems. It is based upon detailed studies of students, which produced basic cognitive science research results on the nature of human thinking and learning. He has written over 60 strictly peer-reviewed publications, and received multiple awards from professional associations. Since coming to WPI, he has received over a dozen major grants from NSF including the prestigious CAREER award, the US Department of Education, ONR, the US Army, the Massachusetts Technology Transfer Center, the Bill and Melinda Gates Foundation, and the Spencer Foundation worth over 13 million dollars. Recently, his work was cited in the National Educational Technology Plan and featured in the *NY Times Sunday Magazine*.

Michael Hoffman

Michael Hoffman is the lead software engineer on the GIFT project with over 9 years of software development experience and a Master's of Science degree from the University of Central Florida. He has been responsible for ensuring that the development of GIFT meets the evolving customer requirements in addition to supporting both intelligent tutoring for computer based training and intelligent tutoring technology research of the growing user community. He manages and contributes support for the GIFT community through various mediums including the GIFT portal (www.GIFTTutoring.org), annual GIFT Symposium conferences, and various technical exchanges. In addition, he excels in integrating third-party capabilities such as software and hardware systems that enable other organizations to integrate GIFT into their training solutions.

Heather Holden

Heather K. Holden, PhD, is an Assistant Professor in the School of Information Technology for Mount Washington College. She is a former researcher for the LITE Lab within ARL-HRED. The focus of her research is in AI and ITS application to education and training; technology acceptance; and human-computer interaction. Dr. Holden's doctoral research evaluated the relationship between teachers' technology acceptance and usage behaviors to better understand the perceived usability and use of job-related technologies. Her work has been published in the *Journal of Research on Technology in Education*, the *International Journal of Mobile Learning and Organization*, the *Interactive Technology and Smart Education Journal*, and several relevant conference proceedings. Her PhD and MS were earned in information systems from the University of Maryland, Baltimore County. Dr. Holden also possesses a BS in computer science from the University of Maryland, Eastern Shore.

Tanner Jackson

G. Tanner Jackson is a research scientist in the Research and Development Division at Educational Testing Service (ETS) in Princeton, NJ. Tanner received a PhD degree in cognitive psychology in 2007 and a MS degree in cognitive psychology in 2004—both from UofM. He also received a BA degree in psychology from Rhodes College in 2001. After completing a Postdoctoral Fellowship at UofM (2008–2011), he continued his research as an Assistant Research Professor within the Learning Sciences Institute at ASU (2011–2013). His current work at ETS focuses on innovative assessments and student process data. His main efforts involve the development and evaluation of conversation-based formative assessments (through ETS strategic initiatives) and game-based assessments (working in collaboration with GlassLab). Additionally, he is interested in how users interact with complex systems, and leverages these environments to examine and interpret continuous and live data streams, including user interactions across time within an assessment system.

Matthew E. Jacovina

Matthew E. Jacovina is a Postdoctoral Scholar working with Dr. Danielle McNamara in the Science of Learning and Educational Technology Lab (SoLET). He received his PhD in cognitive psychology in 2011 working with Dr. Richard Gerrig at Stony Brook University and subsequently worked as a Postdoctoral Fellow with Dr. David Rapp at Northwestern University. He studies the cognitive processes that guide comprehension and communication, focusing on situations in which success is complicated by mismatches between discourse content and prior knowledge, preferential biases, or time pressure. He is interested in how individual differences influence success in these situations, and how educational technology can leverage these understandings to individualize and improve learning. He is currently working on the optimization of iSTART-2 and Writing Pal, game-based tutoring systems teaching reading and writing strategies.

Randy Jensen

Randy Jensen is a group manager at Stottler Henke Associates, Inc., working in training systems since 1993. His research areas include adaptive training, distributed learning, game-based training, behavior modeling, and NLP. He has led projects to develop ITSs and automated after-action review tools for the Army, Air Force, Navy, and Marines. Recent work includes a model-based performance assessment capability for training troubleshooting skills in an ITS for the US Navy. He also recently led the development of a game-based trainer for small unit tactical decision-making at the United States Military

Academy at West Point. Mr. Jensen holds a BS with honors in symbolic systems from Stanford University.

Lewis Johnson

Dr. Lewis Johnson co-founded Alelo in 2005 as a spinout of the University of Southern California. Under his leadership, Alelo has developed into a major producer of innovative learning products focusing on communication skills. Alelo has developed courses for use in a number of countries around the world, all using the Virtual Role-Play method. Dr. Johnson is an internationally recognized leader in innovation for education and training. In 2012, he was keynote speaker at the International Symposium on Automated Detection of Errors in Pronunciation Training in Stockholm. In 2013, he was keynote speaker at the International Association of Science and Technology for Development (IASTED) Technology Enhanced Learning Conference and the SimTecT conference, and was co-chair of the Industry and Innovation Track of the Artificial Intelligence in Education (AIED) 2013 conference. In 2014, he was keynote speaker at the International Conference on Intelligent Tutoring Systems, and was Distinguished Lecturer at the National Science Foundation. When not engaged in developing disruptive learning products, Lewis and his wife Kim produce Kona coffee in Hawaii.

Irvin Katz

Irvin R. Katz is Director of the Cognitive Sciences Research Group at ETS in Princeton, NJ. He earned a PhD in cognitive psychology from CMU in 1988. In addition to ETS, he has held positions at Keio University in Yokohama, Japan; the US Bureau of Labor Statistics; the US Census Bureau; and George Mason University. Throughout his 25-year career at ETS, he has conducted research that applies and develops theories of cognitive learning and reasoning to issues of educational assessment. Dr. Katz is also a human-computer interaction practitioner with more than 30 years of experience in designing, building, and evaluating software for research, industry, and government. The Cognitive Science Research Group that he directs comprises 12 scientists who conduct research and development at the forefront of educational assessment, using cognitive theory in the design of assessments, building cognitive models to guide interpretation of test-takers' performance, and researching cognitive issues in the context of assessment. Moving beyond traditional (e.g., multiple-choice) tests, the group investigates reliable and valid assessment (both summative and formative) using innovative, highly interactive digital environments such as online games, virtual labs or other simulations, and human-agent conversation-based interactions.

Ken Koedinger

Dr. Kenneth Koedinger is Professor of Human-Computer Interaction and Psychology at CMU. His research has contributed new principles and techniques for the design of educational software and has produced basic cognitive science research results on the nature of student thinking and learning. Dr. Koedinger is a co-founder of Carnegie Learning (carnegielearning.com <<http://carnegielearning.com>>) and the CMU Director of LearnLab (learnlab.org <<http://learnlab.org>>). LearnLab is supporting Big Data investigations in education and, more generally, leverages cognitive and computational approaches to support researchers in investigating the instructional conditions that cause robust student learning. See pact.cs.cmu.edu/koedinger.html <<http://pact.cs.cmu.edu/koedinger.html>> for more information.

H. Chad Lane

H. Chad Lane is an Associate Professor of Educational Psychology and Informatics at the University of Illinois, Urbana-Champaign (UIUC). His work focuses on the application of AI and entertainment technologies to educational problems. He has published over 40 papers in areas including educational games, pedagogical agents, scaffolding/feedback, and virtual environments for learning. Prior to joining UIUC, he was the Director for Learning Sciences Research at the USC ICT. He received his PhD in Computer Science in 2004 from the University of Pittsburgh where he studied intelligent tutoring systems and the learning sciences. In 2013, Chad served as the Program Co-Chair for the 16th International Conference on AIED. He also serves on the executive committee of the AIED Society (an elected position), as an associate editor for several major educational technology journals, and as an advisor for the NSF Cyberlearning CIRCL center. More information is available on his website: <http://hchadlane.net>.

James Lester

James C. Lester is Distinguished Professor of Computer Science and Director of the Center for Educational Informatics at North Carolina State University. His research focuses on transforming education with technology-rich learning environments. Using AI, game technologies, and computational linguistics, he designs, develops, fields, and evaluates next-generation learning technologies for K-12 science, literacy, and computer science education. His work on personalized learning ranges from game-based learning environments and ITSs to affective computing, computational models of narrative, and natural language tutorial dialogue. The adaptive learning environments he and his colleagues develop have been used by thousands of students in K-12 classrooms throughout the US. He received his BA (Highest Honors, Phi Beta Kappa), MSCS, and PhD in computer science from the University of Texas at Austin. He received his BA in history from Baylor University. He has served as Editor-in-Chief of the *International Journal of Artificial Intelligence in Education* and Program Chair for the International Conference on Intelligent Tutoring Systems, the International Conference on Intelligent User Interfaces, and the International Conference on Foundations of Digital Games. The recipient of a NSF CAREER Award, he is a Fellow of the Association for the Advancement of Artificial Intelligence (AAAI).

Marcia Linn

Marcia C. Linn is Professor of Development and Cognition, specializing in S&T in the Graduate School of Education, UC Berkeley. She is a member of the National Academy of Education and a Fellow of the American Association for the Advancement of Science (AAAS), the American Psychological Association, and the Association for Psychological Science. She has served as President of the International Society of the Learning Sciences, Chair of the AAAS Education Section, and on the boards of the AAAS, the Educational Testing Service Graduate Record Examination, the McDonnell Foundation Cognitive Studies in Education Practice, and the NSF Education and Human Resources Directorate. Awards include the National Association for Research in Science Teaching Award for Lifelong Distinguished Contributions to Science Education, the American Educational Research Association Willystine Goodsell Award, and the Council of Scientific Society Presidents first award for Excellence in Educational Research.

Danielle S. McNamara

Danielle S. McNamara is a Professor in the Psychology Department at ASU and director of the Science of Learning and Educational Technology laboratory. She focuses on educational technologies and discovering new methods to improve students' ability to understand challenging text, learn new

information, and convey their thoughts and ideas in writing. Her work integrates various approaches and methodologies including the development of game-based ITSs (e.g., iSTART, Writing Pal), the development of NLP tools (e.g., iSTART, Writing Pal, Coh-Metrix, the Writing Assessment Tool), basic research to better understand cognitive and motivational processes involved in comprehension and writing, and the use of learning analytics across multiple contexts. More information about her research and access to her publications are available at soletlab.com.

Noboru Matsuda

Dr. Noboru Matsuda is research faculty at the Human-Computer Interaction Institute at CMU. His primary research interest is in an application of cutting-edge technologies to build an effective learning technology for all students. To achieve this goal, Dr. Matsuda studies the transformative theory of advanced educational technology as well as cognitive theories of learning and teaching. Dr. Matsuda received a PhD in intelligent systems from the University of Pittsburgh in 2004. Dr. Matsuda has developed a number of ITSs in math (arithmetic, geometry theorem proving and algebra equations), C language, and the formal specification language Z. In recent years, Dr. Matsuda has been leading the SimStudent project (www.SimStudent.org) where the research team develops an AI that learns problem-solving skills through guided-problem solving (aka peer tutoring) and worked-out examples (aka learning by self-explanation). Applications of SimStudent include (1) developing an innovative authoring system for cognitive tutors by using SimStudent as an intelligent apprentice that learns subject matter knowledge from authors, (2) understanding the theory of learn by teaching by using SimStudent as a synthetic peer that students can teach, and (3) advancing theory of learning by running simulations using SimStudent.

Camillia Matuk

Camillia Matuk is Assistant professor of Educational Communication and Technology at New York University's Steinhardt School of Culture, Education, and Human Development. Her interests are in the design of technologies for teaching, learning, and collaboration. Recently, she has been involved in researching how tools within online learning environments can support classroom science inquiry, and how they can encourage teachers to design and refine their instruction. Matuk has a PhD in learning sciences from Northwestern University, an MSc in biomedical communications from the University of Toronto, and a BSc in biological sciences from the University of Windsor. She completed a postdoctoral fellowship with the TELS center at UC Berkeley.

Tanja Mitrovic

Dr. Antonija (Tanja) Mitrovic is a full professor and the Head of the Department of Computer Science and Software Engineering at the University of Canterbury, Christchurch, New Zealand. She is the leader of Intelligent Computer Tutoring Group (ICTG). Dr. Mitrovic received her PhD in computer science from the University of Nis, Yugoslavia, in 1994. Prof. Mitrovic is president of the International Society of Artificial Intelligence in Education. She is an associate editor of the following journals: *International Journal on Artificial Intelligence in Education*, *IEEE Transactions on Teaching and Learning Technologies*, and *Research and Practice in Technology Enhanced Learning* (RPTEL). Dr. Mitrovic's primary research interests are in student modeling. ICTG has developed a number of constraint-based intelligent tutoring systems in a variety of domains, which have been thoroughly evaluated in real classrooms, and proven to be highly effective. These systems provide adaptive support for acquiring both problem-solving skills and meta-cognitive skills (such as self-explanation and self-assessment). Although most of the ITSs developed by ICTG support students learning individually in areas such as database querying (SQL-Tutor), database design (EER-Tutor and ERM-Tutor), and data normalization (NORMIT),

there are also constraint-based tutors for object-oriented software design and collaborative skills, various engineering topics (thermodynamics, mechanics), training to interpret medical images and language-learning. ICTG has also developed the Authoring Software Platform for Intelligent Resources in Education (ASPIRE), a full authoring and deployment environment for constraint-based tutors. Recent research includes affect-aware tutors and motivational tutors. She has authored over 200 peer-reviewed publications.

Bradford Mott

Bradford Mott is a Senior Research Scientist in the Center for Educational Informatics at North Carolina State University. He received his PhD in computer science from North Carolina State University, where his research focused on intelligent game-based learning environments. His research interests include AI and human-computer interaction, with applications in educational technology. In particular, his research focuses on game-based learning environments, intelligent tutoring systems, computer games, and computational models of interactive narrative. His research has been recognized with best paper awards and he has contributed to several award-winning video games, including one that received a game of the year award. He has many years of software development experience from industry, including extensive experience in the video game industry, having served as Technical Director at Emergent Game Technologies where he created cross-platform middleware solutions for Microsoft's Xbox and Sony's PlayStation video game consoles.

Tom Murray

Dr. Tom Murray is a Senior Research Fellow in School of Computer Science at the University of Massachusetts Amherst. His current research areas include supporting social deliberative skills in online contexts, and text analytics for cognitive developmental levels. He has also published in the areas of ITS authoring tools, adaptive hypermedia, intelligent learning environments, and knowledge engineering. He is also publishes papers in the field of integral theory on embodied epistemology, contemplative dialogue practices, and applied ethics. Murray has degrees in educational technology (EdD, MEd), computer science (MS), and physics (BS). He is on the editorial review boards of two international journals, *the International Journal of Artificial Intelligence in Education* and *Integral Review* (as an Associate Editor).

Benjamin Nye

Benjamin D. Nye is a research assistant professor at UofM at the IIS. His current focus is on ITS architectures, with a focus on lowering barriers to developing and adopting ITS technology. His primary research project is the ONR STEM Grand Challenge, where he is researching natural language tutoring modules called sharable knowledge objects (SKOs). He is also involved in cognitive agent-based architectures. His thesis topic was "Modeling Memes: A Memetic View of Affordance Learning," which examined memes theoretically and computationally through a model that synthesized Shannon Information Theory and Observational Learning from Bandura's Socio-Cognitive Learning Theory.

Brent Olde

Dr. Brent Olde is a Lieutenant Commander in the US Navy. He is currently assigned as a Program Officer and Division Deputy at ONR's Human & Bio-Engineered Systems Division. He manages several S&T programs; primarily Live, Virtual, and Constructive (LVC) training; Unmanned Aerial Systems (UAS) Selection, Interface, and Training; and STEM ITSs. He received his undergraduate degree at the University of Missouri - Columbia and his PhD in experimental psychology at UofM. Upon completion

he was commissioned as a Lieutenant in the US Navy, completed primary flight training in 2003, and was designated an US Navy AEP. He has completed tours at NAVAIR 1.0, Program Manager (PMA205 - Training Systems); NAVAIR 4.6, Human Systems Research and Engineering Department; Naval Postgraduate School, Assistant Professor; and Naval Aerospace Medicine Institute (NAMI), Fleet Support Division Officer.

Andrew Olney

Andrew Olney is presently an Associate Professor in the Institute for Intelligent Systems/Department of Psychology at UofM and Director of the IIS. Dr. Olney received a BA in linguistics with cognitive science from University College London in 1998, an MS in evolutionary and adaptive systems from the University of Sussex in 2001, and a PhD in computer science from UofM in 2006. His primary research interests are in natural language interfaces. Specific interests include vector space models, dialogue systems, unsupervised grammar induction, robotics, and ITSs. Dr. Olney frequently serves as program committee member and journal reviewer in the fields of cognitive science, AI, and education. Together with his collaborators, Dr. Olney has been awarded \$9.3 million from federal funding agencies including the NSF, the Institute for Education Sciences, and the DoD. His research has been featured in *WIRED Magazine*, the *New York Times*, the *Wall Street Journal*, the Discovery Science Channel, and BBC Radio 4. Dr. Olney was awarded first place in an international robotics competition for the PKD Android (AAAI, 2006) and received the Early Career Research Award from UofM.

Scott Ososky

Dr. Scott Ososky is a Postdoctoral Research Fellow at the STTC within ARL-HRED. His current research examines mental models of adaptive tutor authoring, including user experience issues related to tools and interfaces within the adaptive tutor authoring workflow. His prior work regarding mental models of human interaction with intelligent robotic teammates has been published in the proceedings of the Human Factors and Ergonomics Society, HCI International, and SPIE Defense & Security annual meetings. Dr. Ososky received his PhD and MS in modeling & simulation, as well as a BS in management information systems from the University of Central Florida.

Philip Pavlik

Philip I. Pavlik Jr. is currently an Assistant Professor of Psychology at UofM's IIS. Dr. Pavlik received a BA from the University of Michigan in Economics and a PhD from CMU where he studied cognitive psychology with John Anderson (developer of the Adaptive Control of Thought—Rational (ACT-R) cognitive modeling system) and received a neuroscience certificate from the Center for the Neural Basis of Cognition. With Anderson, Pavlik has pioneered changes in the ACT-R theory that have allow his research to use this theory to quantitatively optimize the learning of information for tasks such as flashcard learning. From this foundation, his work with Ken Koedinger has developed to focus on problem solving, schema learning, optimal transfer, effects of motivational constructs, and student strategy use. His methodologies include theory development, experimentation, mathematical modeling, and educational applications. Pavlik has received more than 2.2 million dollars in grant awards from the Institute for Educational Sciences, NSF, and other sources.

Octav Popescu

Octav Popescu is a Senior Research Programmer/Analyst in CMU's Human-Computer Interaction Institute, where he is in charge of TutorShop, the learning management system part of the CTAT project.

He has more than 25 years of experience working on various projects involving natural language understanding and ITSs. He holds an MS in computational linguistics and a PhD in language technologies from CMU.

Charles Ragusa

Charles Ragusa is a senior software engineer at Dignitas Technologies with over 14 years of software development experience. After graduating from University of Central Florida with a BS in computer science, Mr. Ragusa spent several years at SAIC working on a variety of R&D projects in roles ranging from software engineer and technical/integration lead to project manager. Noteworthy projects include the 2006 DARPA Grand Challenge as an embedded engineer with the CMU Red Team, program manager of the SAIC Common Driver Trainer (CDT)/Mine Resistant Ambush Protected (MRAP) Independent Research & Development (IR&D) project, and lead engineer for Psychosocial Performance Factors in Space Dwelling Groups. Since joining Dignitas Technologies in 2009, he has held technical leadership roles on multiple projects, including his current role as the principal investigator for the GIFT project.

Sowmya Ramachandran

Dr. Sowmya Ramachandran is a Research Scientist at Stottler Henke Associates where her work focuses on the application of AI and machine learning to improve education and training. She leads research and development of ITSs and ITS authoring tools for a diverse range of military and civilian domains. Dr. Ramachandran headed the development of ReadInsight, an intelligent tutor for teaching reading comprehension skills to adult English speakers. She also led the development of an intelligent tutor for training Tactical Action Officers in the Navy. This system uses NLP technologies to assess and train tactical action officers (TAOs) and is currently in operational use at the Surface Warfare Officers School. She is currently leading the development of an ITS for training US Navy Information Systems Technicians in troubleshooting and maintenance skills. Dr. Ramachandran holds a PhD from The University of Texas at Austin. For her dissertation, she developed a novel machine learning technique for constructing Bayesian network models from data.

Steven Ritter

Steven Ritter is Chief Scientist at Carnegie Learning. Dr. Ritter received his doctorate in cognitive psychology from CMU and worked with John Anderson and others to develop and evaluate the ITSs that became the basis for Carnegie Learning's products. He was one of the co-founders of Carnegie Learning. Dr. Ritter is the author of numerous papers on the design, architecture, and evaluation of educational technology and served on the education board of the Software and Information Industry Association. His evaluation work has been recognized by the What Works Clearinghouse as fully satisfying their requirements for rigorous evaluation. In his role as chief scientist, Dr. Ritter directs all projects regarding research on the effectiveness of Cognitive Tutor products and guides development projects focused on improving the effectiveness of mathematics curricula. Dr. Ritter also serves as Chief Product Architect, setting the direction of future Cognitive Tutor products.

Jonathan Rowe

Jonathan Rowe is a Research Scientist in the Center for Educational Informatics at North Carolina State University. He received the PhD and MS degrees in computer science from North Carolina State University. He received the BS degree in computer science from Lafayette College. His research is in the areas of AI and human-computer interaction for advanced learning technologies, with an emphasis on

game-based learning environments. He is particularly interested in intelligent tutoring systems, user modeling, educational data mining, and computational models of interactive narrative. He has led development efforts on several game-based learning projects, including Crystal Island: Lost Investigation, which was nominated for Best Serious Game at the Unity Awards and Interservice/Industry Training, Simulation and Education Conference (IITSEC) Serious Games Showcase and Challenge. His research has also been recognized with several best paper awards, including best paper at the Seventh International Artificial Intelligence and Interactive Digital Entertainment Conference and best paper at the Second International Conference on Intelligent Technologies for Interactive Entertainment.

Andrew R. Ruis

A.R. Ruis is a member of the Epistemic Games Group at the Wisconsin Center for Education Research and a fellow of the Medical History and Bioethics Department at the University of Wisconsin-Madison. He received his BS and BA from the University of California, Davis, and his MA and PhD from the University of Wisconsin-Madison.

Jonathan Sewall

Jonathan Sewall is a Project Director in the Human-Computer Interaction Institute at CMU. For the last 11 years, he has been the technical lead on the CTAT project, which aims to create tools that speed the development of ITSs. Mr. Sewall has more than 32 years of experience in system design, development, integration and testing. His technical expertise includes Java, C, JavaScript, C++, HTML and other programming languages. His career experience has included developing software for the Joint Chiefs of Staff's strategic missile warning system; debugging network software, training operators and handling problems in the Internet's Network Operations Center; designing and building a data base system for personal computer application distribution; creating a system to automate oil terminal operations; and building server software for retrieving and displaying electronic medical records. His roles have ranged from system tester to software debugger to designer to project manager.

Dylan Schmorrow

Dr. Schmorrow is the Chief Scientist at Soar Technology (SoarTech) where he is leading the advancement of research and technology tracks to build intelligent systems for defense, government, and commercial applications that emulate human decision making in order to make people more prepared, more informed and more capable. He has led numerous initiatives that transformed promising technologies into operational capabilities and he successfully transitioned several significant prototypes to operational use. His past service includes the Deputy Director, Human Performance, Training, and BioSystems at the Office of the Secretary of Defense, Program Manager for DARPA, Research Scientist and Branch Head at the Naval Air Warfare Center, Chief Scientist for Human-Technology Integration at the Naval Research Lab, Assistant Professor at the Naval Postgraduate School, Program Officer at ONR, and Executive Assistant to the Chief of Naval Research. He received a commission in the US Navy in 1993 as a Naval aerospace experimental psychologist and completed naval flight training in 1994. He retired as a US Navy Captain in 2013 after twenty years of service where he was both an aerospace experimental psychologist and an acquisition professional leading research and development programs.

David Shaffer

David Williamson Shaffer is a Professor at the University of Wisconsin-Madison in the Department of Educational Psychology and a Game Scientist at the Wisconsin Center for Education Research. Before

coming to the University of Wisconsin, Dr. Shaffer taught grades 4–12 in the United States and abroad, including 2 years working with the Asian Development Bank and US Peace Corps in Nepal. His MS and PhD are from the Media Laboratory at MIT, and he taught in the Technology and Education Program at the Harvard Graduate School of Education. Dr. Shaffer was a 2008–2009 European Union Marie Curie Fellow. He studies how new technologies change the way people think and learn, and his most recent book is *How Computer Games Help Children Learn*.

Anne Sinatra

Anne M. Sinatra, Ph.D. is a Research Psychologist and Adaptive Tutoring Scientist in the LITE Lab within ARL-HRED. The focus of her research is in cognitive psychology, human factors psychology, and adaptive tutoring. She has specific interest in how information relating to the self and about those that one is familiar with can aid in memory, recall, and tutoring. Her dissertation research evaluated the impact of using degraded speech and a familiar story on attention/recall in a dichotic listening task. Her post-doctoral work examined the self-reference effect and personalization in the context of computer-based tutoring. Her work has been published in the journal *Interaction Studies*, and in the conference proceedings of the Human Factors and Ergonomics Society and Human-Computer Interaction International. Prior to her current position, Dr. Sinatra was a Visiting Assistant Professor of Cognitive Psychology at Stetson University and completed 2 years as an Oak Ridge Associated Universities/ARL Post-Doctoral Fellow in ARL's LITE Lab. Dr. Sinatra received her PhD and MA in applied experimental and human factors psychology, as well as her BS in psychology from the University of Central Florida.

Erica L. Snow

Erica L. Snow is a graduate student in the Department of Psychology and the Learning Sciences Institute at ASU. Her academic background includes a psychology BS (2007) and a cognitive psychology MA (2014). She is currently pursuing a doctoral degree in the area of cognitive science. Her current research explores the interplay of students' learning outcomes, learning behaviors, and individual differences within ITSs and educational games. She is particularly interested in how methodologies from AI, educational data mining, and learning analytics can be applied to discover patterns in students' logged interactions with computer-based learning environments.

Ronald Tarr

Ronald W. Tarr is a Senior Research Faculty Member at the University of Central Florida and Program Director of the Research in Advanced Performance Technologies and Educational Readiness (RAPTER) Lab at the Institute for Simulation and Training (IST). He leads a team of interdisciplinary researchers who function as analysts, planners, integrators and designers of the advanced applications of simulation and learning technologies for enhancing human performance.

Robert Taylor

Robert Taylor is a Senior Research Software Engineer in the Center for Educational Informatics at North Carolina State University. His primary focus is designing and implementing game-based learning environments and intelligent cyberlearning systems that leverage video game and cloud-based computing technologies. His work includes creating cutting-edge AI research platforms and deploying software systems of commercial-quality and scalability. Thousands of students across the United States have used these adaptive learning environments for STEM education in elementary school classrooms. He received his ME and BS degrees in engineering mathematics and computer science from the University of

Louisville. The majority of his career has focused on designing and implementing commercial software solutions that range in scale and complexity from mobile applications to enterprise-class software. His professional interests include video game technologies, content authoring tools, ITSs, and AI technologies.

Martin van Velsen

Martin van Velsen is a Senior Research Engineer in the Human-Computer Interaction Institute and graduate student in the Language Technologies Institute at CMU. He is the lead visualization developer for the CTAT authoring research group. Martin also works full time on research projects of a wildly varied nature, some which are: neurosurgery simulations, large-scale AI architectures, virtual humans, and training simulations. He serves as technical adviser to many leading specialists in the field of serious games, simulations, and digital entertainment. As a digital storytelling expert, he serves as research consultant to various entertainment companies including Disney Research and Paramount Pictures. He has been a speaker and panel host for various entertainment technology gatherings. Most recently, he took part as a panelist at the PAX East gaming convention, but he has also organized such scientific forums as a panel on Authoring Interactive Narrative at the Stanford Spring Symposium. Over the last 18 years, he has been responsible for shepherding open-ended research projects toward viable products that can be deployed by such organizations as DARPA, Air Force Research Laboratory, and ONR. Finally, he is an award-winning artist, published fiction author, engineer, and a researcher in the field of interactive narrative.

Wayne Ward

Dr. Wayne Ward is Principal Scientist and Chief Financial Officer at Boulder Language Technologies, Inc. He received a BA in mathematical science and psychology at Rice University and an MA and PhD in psychology at University of Colorado. He is also a Research Professor at the Computational Language and Education Research Center for the University of Colorado. Previously, he was appointed as a Research Computer Scientist at CMU. Dr. Ward developed and maintains the Phoenix system, a parser and dialogue manager designed specifically for semantic information extraction from spoken dialogues in limited domains. Phoenix is distributed as freeware by Boulder Language Technologies and by CMU. Dr. Ward then led the effort to incorporate these and additional technologies into the Virtual Human Toolkit, a toolkit for developing conversational systems using animated agents.

Diego Zapata-Rivera

Diego Zapata-Rivera is a Senior Research Scientist in the Cognitive and Learning Sciences Center at ETS in Princeton, NJ. He earned a PhD in computer science (with a focus on AI in education) from the University of Saskatchewan in 2003. His research at ETS has focused on the areas of innovations in score reporting and technology-enhanced assessment (TEA) including work on adaptive learning environments and game-based assessments. His research interests also include evidence-centered design, Bayesian student modeling, open student models, conversation-based tasks, virtual communities, authoring tools, and program evaluation. Dr. Zapata-Rivera has produced over 100 publications including journal articles, book chapters, and technical papers. He has served as a reviewer for several international conferences and journals. He has been a committee member and organizer of international conferences and workshops in his research areas. He is a member of the Board of Special Reviewers of the User Modeling and User-Adapted Interaction journal and an Associate Editor of the *IEEE Transactions on Learning Technologies Journal*. Most recently, Dr. Zapata-Rivera has been invited to contribute his expertise to projects sponsored by the National Research Council, NSF, and NASA.

INDEX

Abelson, H., 27
Abraham, A., xiv
Abrahamse, E.L., 348, 354
Adams, D., 89, 91, 280
Adams, D.M., 89
Adamson, D., 269, 278
adaptive training, xiii, 6, 161, 162, 163, 164, 346, 355, 356, 357, 361, 362
adaptive tutoring, vi, vii, xii, xiv, 86, 92, 236, 255, 258, 278, 345, 352, 353, 358, 370
Adcock, A., 152, 160, 169, 178
Adenowo, A., 42
Adesope, O.O., 151, 160, 251, 275, 280
ADL, iv, 334, 343, 356
affective state, 123, 166
affordances, 27, 37, 51, 52, 180, 288, 313, 314
Ahuja, N.J., 42
Ainsworth, S., xiv, 27, 29, 120, 143, 144, 189, 190, 191, 298, 315, 316, 343
Aist, G., 251, 294, 298
Albright, D., 248, 250
Aleahmad, T., 142, 143
Aleixandre, M., 251
Aleven, Vincent, iii, xiii, 3, 7, 9, 27, 29, 40, 42, 52, 54, 55, 56, 57, 61, 62, 63, 68, 71, 74, 75, 76, 78, 87, 89, 90, 91, 92, 93, 109, 120, 137, 141, 143, 167, 169, 177, 178, 179, 181, 182, 189, 190, 229, 234, 237, 238, 239, 240, 255, 257, 259, 261, 262, 263, 264, 265, 266, 268, 269, 270, 272, 273, 274, 275, 278, 279, 280, 281, 291, 294, 297, 303, 304, 309, 315, 334, 343, 356
Allen, L.K., 110, 111, 118, 119, 120, 121, 149
Almeida, S.F., 55, 63, 92, 144, 281, 299
Amburn, C., 233, 239
Anderson, J. R., xiii, 28, 63, 90
Anderson, J.R., iii, x, xiii, 28, 50, 54, 63, 74, 82, 84, 85, 87, 90, 138, 143, 144, 163, 167, 180, 189, 227, 228, 229, 238, 264, 266, 270, 274, 278, 279, 291, 292, 293, 297, 298, 301, 309, 313, 315, 367, 368
Anderson, L.W., xiii
Andersson P., 28

Arastoopour, G., 180, 181, 189
architecture, iii, ix, xi, 4, 33, 35, 36, 39, 44, 48, 59, 62, 63, 82, 84, 88, 89, 92, 93, 123, 131, 135, 139, 142, 144, 148, 149, 155, 183, 189, 220, 221, 223, 235, 240, 261, 266, 269, 271, 272, 273, 274, 276, 281, 302, 303, 313, 317, 335, 342, 354, 359, 368
Aris, T.N.M., 45
Army Learning Model, v, xiv
Army Research Laboratory, ii, i, iv, v, viii, ix, xiii, xiv, 3, 7, 29, 42, 44, 45, 61, 89, 121, 144, 147, 149, 150, 178, 191, 209, 210, 225, 227, 255, 256, 259, 278, 280, 281, 283, 290, 301, 317, 318, 345, 347, 354, 355, 356, 358, 361, 362, 367, 370
Arnab, S., 160
Arroyo, I., 109, 120, 151, 159, 163, 268, 275, 276, 278
Articulate Global, 216, 218, 224
Artstein, R., 224
Ashley, K., 62, 120, 239
ASPIRE, iv, xiv, 3, 7, 29, 68, 69, 71, 79, 80, 81, 85, 86, 88, 178, 179, 181, 191, 239, 256, 280, 291, 298, 304, 307, 308, 312, 313, 316, 365
authoring tools, i, iv, v, vi, viii, ix, xii, xiii, xiv, 1, 3, 4, 6, 7, 9, 10, 11, 13, 16, 19, 20, 22, 26, 27, 29, 31, 32, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 47, 49, 54, 59, 61, 62, 67, 68, 69, 70, 71, 74, 77, 84, 87, 89, 95, 96, 97, 99, 100, 102, 105, 106, 107, 109, 110, 112, 113, 116, 117, 118, 119, 120, 121, 123, 124, 125, 127, 131, 133, 135, 136, 137, 140, 143, 144, 147, 149, 150, 151, 152, 153, 155, 156, 157, 158, 159, 160, 161, 169, 170, 175, 177, 178, 179, 180, 181, 182, 185, 188, 189, 195, 196, 197, 199, 209, 211, 216, 217, 218, 219, 220, 222, 223, 225, 227, 229, 236, 237, 238, 255, 256, 257, 258, 259, 261, 264, 265, 266, 267, 269, 271, 272, 275, 277, 278, 283, 284, 285, 287, 289, 290, 291, 292, 295, 296, 298, 299, 301, 303, 304, 305, 309, 310, 311, 312, 313, 314, 315, 316, 317, 320, 329, 330, 333, 334, 338, 342, 343, 345, 352, 353, 356, 357, 359, 360, 366, 368, 370, 371

- AutoTutor, v, xiii, 3, 4, 28, 33, 38, 43, 45, 62, 120, 133, 135, 136, 147, 148, 152, 159, 160, 169, 170, 174, 178, 179, 190, 191, 196, 197, 199, 200, 201, 202, 203, 204, 205, 207, 208, 209, 210, 223, 225, 238, 239, 256, 315, 330, 332, 334, 355
 Axelrod, R., 106
 Azevedo, R., 61
 Baek, J.Y., 185, 190
 Baer, W., 199, 209
 Bagley, E.A., 180, 181, 189
 Baker, A., iv, xiii, 43, 49, 62, 87, 90, 112, 120, 148, 149, 163, 166, 167, 225, 245, 251, 270, 273, 279, 334
 Baker, R., 238, 343
 Baker, R.S., xiii, 62, 90, 120, 149, 167, 278, 279
 Bakhtin, M., 242, 251, 252
 Bannan-Ritland, B., 103, 106
 Barab, S.A., 181, 189
 Barba, C., 43
 Barlow, S.T., 251
 Barnes, T., 144, 273, 281, 334, 343
 Barnieu, J., 332
 Barrows, H.S., 211, 224
 Battistini, L., 169, 177
 Bauer, M., 112, 121, 178, 280, 296, 299
 Beal, C., 109, 120, 151, 159
 Beauchat, T., 233, 239
 Bebbington, A., 92
 Beck, I., 251
 Beck, J.E., 138, 139, 143, 244
 behavior, vii, ix, x, xii, xiv, 12, 16, 23, 34, 36, 37, 47, 52, 54, 56, 59, 63, 68, 74, 75, 76, 77, 84, 85, 86, 87, 88, 91, 93, 101, 103, 109, 112, 118, 119, 121, 125, 133, 137, 138, 139, 144, 149, 151, 152, 155, 156, 157, 158, 159, 161, 163, 164, 165, 166, 167, 185, 188, 190, 218, 220, 221, 222, 223, 224, 228, 240, 251, 257, 261, 262, 263, 264, 266, 267, 268, 269, 270, 271, 277, 279, 281, 292, 294, 299, 302, 305, 307, 309, 312, 315, 332, 333, 335, 336, 337, 338, 339, 342, 348, 349, 350, 351, 357, 362, 370
 Belenky, D.M., 29, 92, 280
 Bell, Benjamin, 6, 31, 33, 34, 35, 37, 39, 41, 42, 45, 96, 103, 242, 251, 357
 Benbya, H., 27
 Bennett, R.E., 37, 42, 169, 177
 Bennett, W., 42
 Bereiter, C., 27
 Berenfeld, B., 108
 Berkey, Karissa, 257, 283, 357
 Berkowitz, M., 44
 Berland, L., 242, 251
 Berman, S.R., 144
 Bernett, D., 264, 279
 Best, R., 121
 Bharathy, G.K., 62
 Bhogal, R.S., 251
 Billings, D.R., 149, 332
 Billington, I., 37, 42
 Billington, S., 37, 42
 Biswas, G., 61, 281, 297
 Bjork, R., 144
 Blackmore, J., 107
 Blank, G.D., 61
 Blankenship, E., 90
 Blankenship, L.A., 82, 90, 177
 Blessing, Stephen, xiv, 7, 9, 27, 29, 42, 44, 68, 71, 82, 84, 85, 89, 90, 91, 109, 120, 121, 136, 137, 143, 144, 160, 170, 177, 179, 191, 257, 272, 275, 278, 291, 292, 293, 294, 295, 296, 297, 298, 299, 301, 307, 311, 312, 315, 316, 357
 Bloom, Benjamin, x, xiii, 50, 52, 61
 Blyth, P., 225
 Bogoni, L., 240
 Bohemia Interactive Simulations, 222, 224
 Boiney, J., 165, 167
 Bolanos, D., 252
 Bolstad, C.A., 92
 Bolter, J.D., 190
 Bolton, Amy, 149, 161, 357
 Bonn, D., 92
 Bonnell, R.D., 45
 Boonthum, C., xiii, 57, 62, 110, 111, 120, 121, 182, 190, 209, 238, 239
 Booth, R.J., 186, 191
 Bope, E., 44
 Borek, A., 264, 272, 278
 Boschman, F., 106
 Boucher, J., 225
 Boullosa, J., 152, 160
 Bourdeau, J., xiv
 Boyce, Michael, 258, 345
 Boyce, S., 164, 167, 358
 Boyd, P., 218, 224
 Boyle, C.F., 293, 297
 Brandon, R.D., 110, 121
 Bransford, J.D., 42
 Brantley, J.W., 169, 178
 Brawner, Keith, i, iii, iv, x, xiii, xiv, 3, 7, 29, 32, 33, 35, 40, 41, 42, 43, 44, 45, 58, 63, 123, 136, 145, 147, 149, 150, 151, 153, 159, 160, 177, 196, 225, 227, 255, 258, 259, 261, 269, 280, 281, 283, 285, 290, 317, 332, 345, 354, 355
 Brennan, K., 299
 Brew, C., 107
 Bricker, L., 242, 251
 Brown, A.L., 27, 106, 189, 239
 Brown, D., 135, 331
 Brown, J., 42, 238
 Brown, J.S., 42, 43
 Brown, M., 106
 Bruner, J., 27
 Brunskill, E., 144, 238

- Brusilovsky, P., 36, 42, 91, 92, 143, 280, 305, 315
 Buchenroth-Martin, C., 252
 Buckley, B.C., 178
 Bunzo, M., 255, 259, 291, 298
 Burelson, W., 278
 Burke, C.S., 63
 Burkett, C., 61, 209
 Burnette, T., 298
 Burns, H.L., 42
 Burton, A.M., 33, 36, 42, 305, 315
 Burton, R., 42
 Butcher, K., 141, 143
 Butler, A.C., 240
 Butler, H., 169, 177, 190, 209, 210, 229
 Byström, K., 27
 Cade, W., xiii, 62, 238, 239
 Cai, Zhiqiang, xiii, xiv, 43, 47, 62, 111, 120, 123, 133, 136, 147, 149, 150, 152, 160, 179, 186, 190, 196, 197, 199, 201, 209, 210, 234, 236, 238, 240, 343, 358
 Callaway, C.B., 151, 159
 Camp, P.J., 298
 Campbell, D.J., 27
 Campione, J.C., 27, 106
 Cannon-Bowers, J., 135
 Capeheart, A., 298
 Capps C.G., 42
 Carpenter, R., 224
 Carter, E., 61
 Cedillos, E.M., 45
 Cen, H., 138, 141, 143, 168, 273, 278
 Chand, D., 278
 Chang, H., 106
 Chang, K.M., 99, 106, 138, 143
 Chase, C.C., 231, 238, 264, 279
 Chauncey, A., 61
 Chen, Z., 209, 239, 240
 Chesler, N.C., 180, 181, 189
 Chi, M.T.H., 251
 Chilana, P.K., 27
 Childers, D.L., 62
 Chin, C., 251
 Chin, D.B., 238, 242
 Chipman, P., 28, 43, 62, 135, 159, 238
 Chiu, J.L., 106
 Chklovski, T., 232, 238
 Choi, J., 250
 Cintron, L.M., 234, 240, 317, 318, 332
 Clark, D., 61, 106, 189, 252
 Clark, R.C., 61, 143
 Clark, R.E., 42, 90
 Clarke-Midura, J., 169, 177, 178, 181, 190
 Clemente, J., 163, 167
 Cobb, P., 27, 190
 Code, J., 120, 169, 177, 185, 187
 cognitive modeling, xiv, 74, 89, 161, 256, 259, 278, 304, 367
 cognitive state, 50
 Cohen, M., 106, 290
 Cohen, P., v, xii, xiii, xiv, 54, 62, 77, 78, 91, 102, 106, 109, 120, 187, 229, 230, 231, 258, 262, 288, 291, 311, 315
 Cohen, P.R., xiii, 259
 Cohen, W., 239
 Cohen, W., 315
 Cohen, W.W., xiv, 62, 91, 239, 280, 298, 315
 Cohn, Joseph, 44, 149, 161, 162, 165, 167, 168, 358
 Cole, Ron, 196, 197, 241, 252, 358
 Collier, W., 189
 Collins, A., 27, 29, 42, 43
 Commons, M.L., 28
 Component Display Theory, x, 3, 62, 234, 317, 318, 319, 320, 322, 323, 326, 327, 329, 330, 331, 332, 346, 368
 Conejo, R., 61
 Confrey, J., 27, 190
 Conklin, J., 28
 Conley, M.W., iii, xiii, 181, 190
 Connolly, T., 120
 Conole, G., 106
 Conrad, F.G., 170, 178, 297
 consistency, 235, 287, 288, 330
 Constantin, A., 28
 Converse, S.A., 251
 Conway, M., 298
 Cook-Greuter, S.R., 28
 Cooper, H., 143, 190, 275, 281
 Copeland, J., xiii
 Corbett, A., xiii, 62, 63, 90, 92, 120
 Corbett, A.T., iii, xiii, 50, 54, 62, 63, 74, 76, 84, 87, 90, 92, 112, 120, 138, 143, 163, 167, 228, 238, 264, 265, 266, 268, 270, 274, 278, 279, 280, 292, 293, 297, 301, 315, 334, 343
 Core, Mark, 62, 225, 257, 301, 304, 309, 310, 311, 359
 Corrigan, S., 178
 Cosgrove, D., 298
 Cox, M.T., 62
 Craig, S.D., 190, 209
 Crismond, D., 298
 Cristea, A., 28
 Crossley, S.A., 110, 111, 119, 120
 Crouch, C.H., 291, 298
 Croy, M., 273, 281
 CTAT, v, 3, 27, 54, 61, 68, 69, 71, 72, 74, 75, 76, 77, 78, 79, 82, 84, 85, 86, 87, 88, 89, 143, 179, 181, 189, 229, 230, 231, 236, 256, 259, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272, 273, 274, 275, 276, 277, 278, 291, 292, 294, 303, 304, 309, 311, 313, 315, 343, 356, 359, 367, 369, 371
 Cue, Y., 152, 159, 225

- Cummings, P., 332
 Cunningham, K., 279
 Cviko, A., 106
 Cycorp, 232, 238
 Cypher, A., 90, 315
 Cytrynowicz, M., 144
 D'Angelo, C.M., 106, 189
 D'Mello, S.K., xiii, 43, 45, 62, 160, 190, 209, 238,
 239, 240, 279, 281, 315
 Dabbagh, N.H., 95, 96, 103, 105, 106
 Dahmann, J., 62
 Dai, Jianmin, 68, 109, 359
 Damelin, D., 108
 data analytics, 165, 275
 data mining, 92, 117, 156, 159, 275, 278, 279, 296,
 331, 368, 370
 Davenport, J., 178
 Davidson, M., 44
 Davis, E.A., 96, 104, 106, 369
 Day, J., 92
 De Antonio, A., 163, 167
 de Carvalho, A., 62
 de Freitas, S., 160
 de Jong, T., 44, 45, 343
 De Kleine, E., 348, 354
 de Raedt, L., 92
 De Wever, B., 107
 Deaton, J., 43
 Dede, C., 169, 177, 181, 190
 DeFalco, J.A., 32, 45, 148, 149
 deJong, L., 242, 251, 334
 del Blanco, Á., 120
 de-la Cruz, J.L.P., 61
 DeLeeuw, K.E., 91
 DeLine, R., 298
 Demi, Sandra, 90, 257, 261, 278, 359
 Dennis, S., 111, 120, 190, 210, 228, 239
 Derry, S., 43
 Desmarais, M.C., 144, 163, 166, 167
 Devasani, S., 90, 278, 297, 298, 315
 Development, iv, xiv, 13, 23, 28, 29, 42, 43, 84, 106,
 133, 149, 152, 160, 161, 167, 168, 176, 178, 182,
 185, 217, 248, 259, 265, 299, 318, 330, 333, 343,
 356, 357, 360, 362, 363, 364, 365, 368, 369
 DiazGranados, D., 63
 DiCerbo, K., 178
 Dickison, D., 139, 143, 144
 Dieterle, E., 181, 190
 diSessa, A., 27
 Dligach, D., 250
 Dobson, W., 43, 44
 Dolletski-Lazar, R., 333, 344
 domain model, iii, iv, vii, 3, 34, 79, 86, 218, 232,
 257, 262, 264, 267, 269, 270, 308, 334
 domain modeling, iii, viii, 40, 342, 346
 Domeshek, Eric, 257, 333, 360
 Donnelly, D.F., 96, 106
 Dow, S., 185, 190
 Downing, B., 252
 Dozzi, G., 62, 239
 Drake, M., xiii
 Duguid, P., 42
 Dunwell, I., 160
 Durkin, K., 89, 91, 280
 Durlach, P., xiii, 62, 224, 225
 Duschl, R., 242, 251
 Dwyer, D., 43
 Dyke, G., 269, 278
 Eagle, M., 273, 281
 Early, S., 42, 90
 Easterday, M.W., 61
 Eastmond, E., 299
 Edelson, D., 106
 educational games, 121, 181, 188, 191, 356, 363, 370
 Eduworks, 233, 238
 Edwards, J.E., 168
 Eggan, G., 255, 259, 291, 298
 Eksin, C., 62
 Elkins, D., 135
 Elson-Cook, M., xiii
 EMAP, 317, 318, 319, 320, 322, 323, 324, 325, 326,
 328, 329, 330, 331
 Emme, D., 44
 Emonts, M., 215, 224
 Engestrom, Y., 28
 Erduran, S., 251, 252
 Estes, F., 90
 Evanini, K., 169, 177
 Evans, C., 96, 106
 expert model, v, ix, 6, 32, 35, 78, 79, 109, 257, 292,
 307, 330, 346, 350
 expert modeling, 32, 305, 334, 359
 Eylon, B.S., 96, 99, 105, 106, 107
 Fancsali, S.E., 144
 Fano, A., 45
 Farha, N., 320, 332
 Fasse, B., 298
 feedback, iv, vii, viii, ix, x, 4, 13, 14, 15, 16, 18, 20,
 34, 37, 40, 47, 49, 51, 52, 53, 54, 56, 57, 58, 59,
 60, 63, 68, 73, 74, 76, 77, 82, 85, 86, 88, 89, 92,
 96, 98, 99, 100, 102, 103, 104, 109, 110, 111, 113,
 115, 117, 118, 119, 137, 139, 140, 142, 151,
 180, 188, 195, 213, 214, 216, 221, 223, 228, 229,
 230, 236, 241, 244, 247, 258, 263, 264, 265, 266,
 270, 272, 274, 276, 281, 283, 284, 287, 288, 293,
 296, 302, 305, 307, 309, 310, 311, 312, 318, 330,
 334, 335, 338, 340, 345, 349, 352, 363
 Feldon, D., 90
 Feng, M., 63, 92, 144, 281, 299
 Feng, S., 55, 63, 92, 144, 199, 209, 281
 Ferguson, W., 27
 Fernández-Manjón, B., 120

- Fillmore, C., 251
 Fischer, K., 28
 Fisher, C.R., 45
 Fleming, P., 109, 120, 307, 315
 Fletcher, D.F., iv, vi, xiii, 162, 167
 Fletcher, J.D., xiii
 Florin, C., 240
 Folsom-Kovarik, J.T., 333, 344
 Forbell, E., 62, 225
 Forbus, K.D., 251, 271, 279
 Forlizzi, J., 264, 279
 Forsyth, C., 177, 190, 199, 209, 210
 FOSS, 242, 243
 Foster, D., 165, 167
 Fournier-Viger, P., 61, 238
 Fowler, S., 43
 Fowlkes, J.E., 43
 Francis, M.E., 45, 186, 191, 290
 Franklin, S., 7
 Frederiksen, J., 43
 Fredriksen, A., 250
 Freeman, Hannah, 33, 149, 161, 360
 Freeman, J., 42
 Friedland, L., 224
 Friedman-Hill, E., 263, 279
 Furman, M., 107
 Furtak, E.M., 101, 107
 Gagne, R.M., xiii
 Gagné, R.M., 234, 238
 Gandhe, S., 220, 224
 Gandy, M., 190
 Ganoe, C., 279
 Garc, D., 152, 160
 Gasevic, D., 278
 Gentner, D., 28
 Gerard, Libby, 68, 95, 99, 106, 107, 108, 360
 Germany, M.L., 90, 189, 190, 191, 209
 Gersh, J.R., 28
 Geyer, A., 167, 168
 Gholson, B., 209
 Gick, M.L., 229, 238
 GIFT, iii, iv, v, vi, vii, viii, ix, x, xi, xii, xiv, 3, 4, 6,
 7, 29, 35, 40, 41, 44, 45, 59, 60, 61, 68, 69, 70, 84,
 88, 89, 105, 109, 119, 123, 124, 125, 126, 127,
 128, 130, 131, 132, 133, 134, 135, 136, 142, 144,
 147, 148, 149, 150, 151, 159, 167, 177, 189, 197,
 208, 211, 212, 223, 224, 234, 237, 250, 255, 256,
 258, 259, 261, 265, 266, 267, 269, 271, 272, 274,
 275, 276, 277, 281, 283, 284, 285, 286, 287, 288,
 289, 290, 292, 295, 296, 299, 302, 303, 311, 313,
 314, 315, 317, 319, 320, 321, 322, 323, 324, 325,
 326, 327, 328, 329, 330, 331, 332, 342, 343, 345,
 347, 352, 353, 354, 356, 361, 368
 Gilbert, R.B., 160
 Gilbert, Stephen, viii, xiv, 68, 71, 82, 84, 85, 89, 90,
 92, 151, 170, 177, 255, 257, 259, 272, 275, 278,
 291, 292, 293, 294, 296, 297, 298, 311, 312, 315,
 357, 361
 Gildea, D., 252
 Goguadze, G., 91, 280
 Goldberg, Benjamin, iii, iv, vi, x, xii, xiii, xiv, 3, 6, 7,
 29, 35, 40, 41, 42, 43, 44, 45, 47, 50, 58, 59, 61,
 62, 63, 121, 123, 135, 136, 148, 149, 150, 151,
 159, 160, 177, 178, 209, 210, 225, 234, 240, 255,
 257, 258, 259, 261, 281, 283, 290, 301, 317, 318,
 319, 320, 332, 345, 350, 354, 361
 Goldman, R., 225
 Goldman, S., 279
 Goldstein, D.S., 279
 Goldstone, R., 189
 Gong, Y., 139, 143
 González-Brenes, J.P., 143
 González-Calero, P.A., 62
 Goodwin, G.F., 63
 GooruLearning, 233, 238
 Gordon, A.S., 43
 Gorman, G., 224
 Gott, S., 91
 Graesser, Art, i, iii, iv, v, x, xiii, xiv, 3, 7, 11, 28, 29,
 33, 34, 38, 42, 43, 44, 45, 47, 48, 57, 61, 62, 89,
 109, 111, 120, 121, 123, 133, 135, 136, 147, 148,
 149, 150, 151, 152, 153, 159, 160, 169, 170, 177,
 178, 179, 181, 182, 186, 187, 190, 191, 193, 195,
 196, 199, 201, 209, 210, 225, 227, 228, 231, 232,
 234, 238, 239, 240, 241, 251, 269, 275, 278, 279,
 280, 281, 283, 290, 301, 315, 330, 332, 343, 355
 Graf, P., 119, 121
 Grance, T., 134, 136
 Gray, J., 298
 Green, T.R.G., 90
 Greer, J., 43, 143, 315
 Griffin, B.A., 275, 280
 Grimshaw, S., 27, 143, 189, 315
 Grishman, R., 187, 190
 Grooms, J., 252
 Grossman, R., 164, 167
 Grünwald, P.D., 28
 Guess, R.H., 111, 120
 Guralnick, D., 43
 Gureghian, D., 343
 Gustha, M., 187, 191
 Guzodial, M., 29
 Guzmán, E., 61
 Hacioglu, K., 252
 Hadley, W.H., 28, 74, 82, 90, 238, 298
 Hadwin, A.F., 112, 120
 Halbert, D.C., 307, 315
 Haff, H.M., 91, 280, 316, 334, 343
 Hall, B., 62, 92, 136, 152, 159, 225, 280, 290, 357
 Halpern, D., 177, 181, 182, 190, 199, 209, 210
 Halpin, S.M., 63
 Han, L., 190

- Hanks, S., xiii, 259
- Hao, J., 178
- Harris, T., 144
- Harris, T.K., xiii, 143, 190
- Hart, J., 62, 225
- Harter, D., 43, 149, 178, 251
- Hartley, J.R., 43
- Hasselbring, T.S., 42
- Hassler, B., 116, 120
- Hausmann, R.G.M., 141, 143
- Havighurst, R.J., 43
- Hay, K., 29
- Hayes, M., 27, 143
- Haynes, B., 43, 62, 135, 159, 238
- Haynes, B.C., 28
- Haynes, S.R., 28
- Hays, M., 47, 62, 123, 136, 150, 238, 239, 315
- Hays, P., 62, 136, 150, 238, 239, 343
- Healy, A.F., 119, 121
- Heffernan, C., 28
- Heffernan, C., 144, 224, 279, 315
- Heffernan, Neil, 3, 7, 11, 28, 36, 44, 55, 63, 68, 71, 73, 74, 78, 90, 91, 92, 137, 139, 144, 164, 167, 168, 179, 190, 222, 224, 262, 263, 268, 271, 279, 281, 299, 305, 312, 314, 315, 361
- Hendrix, M., 160
- Hennessy, S., 116, 120
- Henriksen, P., 294, 298
- Hernault, H., 152, 153, 160
- heuristic, 5, 6, 13, 15, 24, 25, 232, 288
- Hickey, D., 181, 190
- Hiebert, J., 107
- Hill, R.W., 225
- Hilton, M., 179, 182, 190
- Hockenberry, M., 91, 167, 190, 279
- Hofer, R.C., 62
- Hoffman, E., 178
- Hoffman, Michael, iv, vi, xii, xiv, 133, 136, 258, 305, 317, 342, 361
- Hoffman, R.R., 315
- Hofmann, M., 136
- Holbrook, J., 298
- Holden, Heather, iii, iv, x, xiii, xiv, 3, 7, 29, 35, 40, 41, 42, 43, 44, 45, 58, 63, 89, 123, 136, 142, 144, 148, 150, 151, 160, 177, 179, 191, 209, 225, 255, 257, 258, 259, 261, 278, 280, 281, 283, 290, 317, 332, 345, 354, 362
- Holland, J., xiv, 7, 29, 92, 93, 178, 191, 239, 280, 298, 316
- Holmes, N.G., 92
- Holt, L., 44
- Holyoak, K.J., 229, 238
- Honey, M.A., 179, 190
- Hsi, S., 101, 106
- Hsieh, P., 334, 343
- Hsu, C-H, 43
- Hu, Xiangen, i, v, xiii, xiv, 6, 29, 33, 42, 43, 44, 45, 47, 48, 61, 62, 65, 67, 89, 121, 123, 133, 136, 147, 148, 149, 150, 151, 159, 160, 178, 179, 186, 190, 191, 196, 199, 201, 209, 210, 225, 227, 238, 239, 269, 275, 278, 280, 281, 283, 290, 330, 332, 343, 355, 356
- Huang, Y., 143, 240
- Hutchinson, R., 168
- Hwang, J., 250
- Hyland, J., 332
- Ingram-Goble, A., 181, 189, 190
- instructional model, iii
- intelligent tutoring system, iii, iv, v, vi, vii, viii, ix, x, xi, xii, xiii, xiv, 3, 4, 5, 6, 7, 9, 10, 11, 12, 13, 16, 17, 18, 19, 20, 26, 27, 28, 29, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 67, 68, 69, 70, 71, 77, 79, 87, 90, 91, 92, 107, 109, 112, 120, 121, 123, 128, 129, 131, 133, 134, 135, 136, 143, 144, 147, 148, 149, 151, 152, 153, 155, 156, 158, 159, 160, 165, 167, 170, 177, 178, 195, 196, 197, 209, 210, 225, 227, 229, 231, 232, 233, 234, 236, 237, 238, 239, 242, 255, 256, 257, 258, 259, 261, 264, 265, 266, 267, 268, 269, 271, 272, 273, 274, 275, 276, 277, 278, 279, 280, 281, 285, 290, 291, 292, 295, 296, 298, 301, 302, 303, 304, 305, 306, 307, 309, 310, 311, 312, 313, 314, 315, 316, 318, 322, 333, 334, 335, 337, 338, 340, 342, 343, 344, 346, 347, 350, 351, 352, 356, 359, 360, 361, 362, 366, 368
- Irby, B.J., 163, 167, 168
- Iseli, M.R., 164, 168
- Ishizuka, M., 152, 153, 160
- Isotani, S., 89
- Iwaniec, D.M., 62
- Jackson, Tanner, 43, 110, 111, 116, 119, 120, 149, 150, 169, 178, 191, 209, 281, 362
- Jacovina, Matthew, 68, 109, 110, 121, 362
- Jameson, E., 181, 190
- Jamieson-Noel, D., 120
- Jang, H., 278
- Jarmasz, J., 42
- Järvelin, K., 27
- Jarvis, M.P., 90
- Jenkins, F., 169, 177
- Jensen, Randy, 257, 333, 362
- Jeong, H., 61
- Jesukiewicz, P., 233, 238
- Jiang, H., 234, 240, 317, 318, 332
- Jimenez Castro, M., 62
- Jo, I.Y., 61
- Johnson, A., 61
- Johnson, C.W., 28
- Johnson, D., 63, 191
- Johnson, Lewis, iii, xiii, 11, 28, 33, 37, 43, 44, 152, 159, 196, 197, 211, 212, 220, 224, 363

- Johnson, M.C., ii, x, xiii, 13, 14, 20, 28, 44, 49, 50, 61, 63, 89, 91, 181, 191, 280, 311, 316
 Johnson-Laird, P.N., 28
 Johnston, J., 42
 Jona, M., 43, 44, 45
 Jonassen, D., 28
 Jones, E., 90, 279
 Jordan, D.S., 45
 Jordan, P., 18, 28, 35, 43, 45, 92, 149, 152, 159, 191, 223, 225, 281, 285, 286, 290
 Jordan, P.W., 43, 149, 159, 251
 Jordan, T., 28
 Joshi, A., 144
 Joyce, H., 224
 Junker, B., 143, 273, 278
 Jurafsky, D., 252
 Just, M.A., 168
 Just, S., 168, 188, 295, 296
 Kahler, S.E., 251
 Kali, Y., 104, 106
 Kannampallil, T.G., 28
 Karabinos, M., 264, 278
 Karam, R., 275, 280
 Karpicke, J.D., 163, 167
 Kass, A.M., 44
 Katz, Irvin, 149, 150, 169, 170, 178, 363
 Kauffman, L., 90, 279
 Keeling, H., 307, 316
 Kegan, R., 28
 Keiser, V., 315
 Kelly, A., 185, 190, 276
 Kelly, G., 251
 Kelly, K., 279
 Keogh, B., 252
 Keshtkar, F., 201, 210
 Ketelhut, D.J., 181, 190, 191
 Khajah, M., 138, 144
 Kihumba, G., 279
 Kim, J.M., 48, 55, 56, 58, 62, 63, 211, 225, 363
 Kim, R.S., 63
 King Chen, J., 100, 107
 King, B., xiii
 King, K.S., 44
 King, S., iii, xiii, 39, 44, 100, 107, 343
 Kingsbury, P., 252
 Kinnebrew, J.S., 61
 Kintsch, W., 111, 120, 190, 210, 228, 239
 Kinzer, C.K., 42
 Kittredge, R., 187, 190
 Klein, C., 63
 Klein, G., 63, 305, 315
 Klinkenberg, R., 133, 136
 Knerr, B., 43
 Knight, S., 116, 120
 knowledge component, 50, 74, 75, 87, 88, 138, 140, 141, 155, 257, 258, 262, 263, 270, 273, 274, 275
 knowledge representation, 12, 85, 91, 92, 156, 190, 197, 293, 296, 298, 301, 304, 307, 355, 356
 Ko, A.J., 27
 Kodaganallur, V., 267, 279
 Kodavali, S., 89, 90, 291, 294, 298, 311, 312, 315
 Koedinger, Kenneth, iii, v, xiii, xiv, 3, 7, 11, 27, 28, 50, 51, 54, 55, 57, 61, 62, 63, 74, 75, 76, 77, 78, 79, 82, 84, 87, 89, 90, 91, 92, 93, 112, 120, 137, 138, 143, 144, 148, 149, 159, 163, 165, 167, 168, 169, 177, 179, 181, 182, 189, 190, 196, 227, 228, 229, 230, 231, 234, 238, 239, 240, 255, 257, 259, 261, 262, 263, 264, 266, 267, 268, 270, 271, 272, 273, 274, 275, 277, 278, 279, 280, 281, 291, 292, 297, 298, 301, 303, 309, 311, 315, 334, 343, 363, 367
 Koehn, G.M., 43
 Koenig, A.D., 164, 168
 Kogut, B., 102, 107
 Kölling, M., 294, 298
 Kolodner, J.L., 62, 298, 299
 Krajcik, J.S., 104, 106, 251
 Krathwohl, D.R., xiii
 Krause, S.R., 28
 Kraut, R., 143
 Kreuz, R., xiv, 43, 209, 240
 Kreuz, R.J., xiv, 240
 Krugler, W., 252
 Kuhl, F., 62
 Kuhn, D., 242, 251
 Kulatunga, U., 242, 251
 Kulikowich, J., 291, 299
 Kumar, A.N., 62
 Kumar, P., 28
 Kumar, R., 28, 56, 62, 144, 221, 269, 279, 334, 343
 Kyle, K., 120
 Lacerda, G., 91, 298
 Lai, K., 99, 108
 Lajoie, S., xiii, 37, 43, 54, 62, 91, 255, 259, 291, 298
 Lajoie, S.P., 43, 62
 Landauer, T.K., 111, 120, 190, 210, 228, 239
 Lane, H Chad, 37, 44, 61, 62, 91, 93, 120, 144, 220, 225, 257, 279, 301, 304, 309, 310, 311, 363
 Lanfranchi, A., 250
 Lau, T.A., 91
 Laurentino, T., 62
 Lave, J., 104, 180, 190
 Le, N.T., 120
 Leake, D.B., 237, 239
 learner model, iii, vii, ix, xii, 91, 197, 234, 272, 273, 277, 279, 317, 320, 322, 323, 325, 326, 328, 329, 330, 352, 353
 Leber, B., 279
 Leber, Brett, 261
 Lee, A., 91
 Lee, E.P.K., 168
 Lee, J.J., 78, 91, 164, 168, 190, 219

- Lee, S., 225
 Leelawong, K., 291, 297
 Lehrberger, J., 187, 191
 Lehrer R., 27
 Lehrer, R., 27, 185, 190, 251
 Lei, P., 299
 Lepper, M.R., xiii
 Lerner, R., 104, 106
 Lesgold, A., xiii, 54, 62, 224, 255, 259, 291, 298
 Lesh, R.A., 185, 190
 Lester, James, 33, 44, 54, 63, 90, 91, 149, 151, 159, 167, 211, 224, 251, 279, 364
 Levinstein, I.B., 62, 120, 121, 190
 Levy, S.T., 104, 106
 Lewis, J.E., 211, 242, 251, 293, 363
 Lewis, M.W., 297
 Lhommet, M., 152, 156, 159
 Li, H., 209, 210
 Li, N., 91, 239
 Lieberman, H., 91
 Limbach, R., 343
 Lim-Breitbart, J., 107
 Lindros, J., 225
 Lindsey, R.V., 144
 Linn, Marcia, 68, 95, 96, 99, 100, 101, 102, 105, 106, 107, 108, 364
 Lister, K., 279
 Litman, D., xiii
 Litteral, D.J., 43
 Liu, O.L., 99, 107, 141, 144, 169, 178, 275
 Liu, Q., 251, 280
 Livak, T., 44
 Lizotte, D., 251
 Lobene, E., 44
 Lodato, M., 332
 Loke, S., 211, 225
 Loll, F., 109, 120
 Long, R., 44, 76, 91, 190, 233, 239, 264, 268
 Long, Y., 91, 279, 280
 Loper, M.L., 62
 Louwerse, M., 43, 120, 209
 Louwerse, M.M., 43
 Loveland, M., 178
 Lovett, M., 91
 Lovett, M.C., 280
 Lowe, J., 251
 Lu, S., 43, 120, 209
 Lua, 219, 225
 Lucas, D., 251
 Luce, C., 169, 177
 Ludvigsen, S., 106
 Luehmann, A.L., 99, 106
 Lunenburg, F.C., 163, 167, 168
 Luzuriaga, M., 107
 Lynch, C., 30, 62, 63, 120, 239, 259, 299, 316
 Ma, W., 251, 280
 Maass, J.K., 91
 MacIntyre, B., 190
 McLaren, B., 90, 279
 MacLellan, C., 91, 149
 MacLellan, C.J., 79, 91, 148, 149, 231, 239, 257, 259, 263, 280, 311, 315, 343
 Macmillan, S., 44
 MacWhinney, B., 62, 239
 Madhok, J., 107, 108
 Magerko, B., 44
 Magliano, J.P., 228, 238, 240
 Major, N., 27, 143, 190, 315, 343
 Makhoul, J.I., 334, 343
 Malave, V.L., 168
 Mall, H., 44
 Malone, N., 149, 332
 Maloney, J., 299
 Mandel, T., 144
 Mangold, L.V., 233, 239
 Mangrubang, F.R., 157, 159
 Marchiori, E.J., 109, 120
 Mark, M.A., 28, 82, 90, 238, 298, 301, 315, 359
 Markou, M., 62
 Marks, J., 264, 279
 Marsella, S.C., 152, 156, 159, 160, 225
 Martin, B., xiv, 7, 29, 30, 91, 92, 93, 178, 191, 239, 280, 298, 316
 Martin, E., 44, 135, 331
 Martin, J., xiii, xiv, 7, 29, 30, 44, 79, 81, 91, 92, 93, 135, 169, 178, 191, 231, 239, 250, 252, 261, 267, 308, 330, 371
 Martinez-Garza, M., 106
 Marx, R., 251
 Masia, B.B., xiii
 Mason, R.A., 168, 363
 Massey, L.D., 44, 45
 Mathan, S., 231, 239, 271, 280
 Mathews, M., 92
 Matsuda, Noboru, v, xiv, 54, 62, 68, 71, 77, 78, 79, 91, 148, 149, 229, 230, 231, 239, 257, 259, 262, 263, 280, 291, 298, 304, 311, 315, 343, 365
 Matuk, Camilia, 68, 95, 96, 99, 100, 101, 102, 105, 107, 108, 365
 Mayer, R.E., 61, 89, 91, 143, 280
 Mayrath, M., 169, 177, 178
 Mazur, E., 291, 298
 McCaffrey, D.F., 275, 280
 McCollum, C., 43
 McDaniel, M., 144
 McDonald, D., 30
 McElhaney, K., 99, 100, 101, 102, 107
 McGuigan, N., 316
 McGuigan, N., xiv, 7, 29, 93, 178, 191, 239, 316
 McGuire, C.L., 291, 299
 McKelvey, B., 27
 McKenney, S., 106

- McKeown, M., 244, 251
 McKneely, J.A., 28
 McLaren, B., xiii, 27, 42, 61, 63, 89, 90, 91, 278
 McLaren, B.M., 229, 238
 McLaren, B.M., xiii, 27, 40, 42, 54, 57, 61, 63, 74, 75, 76, 78, 89, 90, 91, 143, 144, 167, 169, 177, 179, 181, 189, 190, 255, 259, 262, 263, 264, 265, 275, 277, 278, 279, 280, 291, 297, 303, 309, 315, 334, 343
 McLaren, P.B., 279
 McLaughlin, E.A., 279
 McLaughlin, M.W., 96, 107, 143, 144, 163, 168, 238, 264, 273, 279
 McNamara, Danielle, 52, 56, 57, 62, 63, 68, 109, 110, 111, 112, 116, 118, 119, 120, 121, 182, 187, 190, 210, 228, 239, 359, 362, 364
 McNeill, K.L., 251
 McTear, M.F., 228, 239
 Means, B., 91
 Means, M., 87, 91, 252
 Medvedeva, O., 144
 Mehta, A., 161, 168
 Mell, P., 134, 136
 mental models, xiii, 9, 11, 13, 20, 21, 23, 27, 28, 257, 352, 367
 Mercer, A., 156, 159
 Meron, J., 152, 159
 Merrill, D., xiv, 354
 Merrill, M.D., 62, 332
 Messina, R., 30
 Metiu, A., 102, 107
 Meyer, O., 91
 Michael, J., 178
 Miettinen, R., 28
 Milik, N., xiv, 7, 29, 93, 178, 191, 239, 280, 298, 316
 Millán, E., 61
 Miller, J.R., 44
 Millis, K., 111, 120, 177, 190, 199, 209, 210
 Mirel, B., 28
 Mislevy, R., 169, 170, 178, 187, 191
 Misra, A.K., 28
 Mitamura, T., 93, 240
 Mitchell, H., 43, 120, 166, 320, 332
 Mitchell, H. H., 43, 209
 Mitchell, T.M., 168
 Mitrovic, Tanja, iv, xiv, 3, 7, 9, 11, 29, 30, 32, 44, 52, 62, 68, 71, 79, 80, 81, 85, 91, 92, 93, 169, 178, 179, 181, 190, 191, 227, 228, 229, 231, 239, 240, 266, 267, 269, 271, 278, 280, 291, 298, 304, 307, 308, 316, 365
 Mizoguchi, R., xiv, 29
 Mochol, M., 163, 168
 Monroy-Hernandez, A., 299
 Moog, R.S., 251
 Moore, D.R., 43
 Moran, T.P., 45
 Moreno, K., 120, 152, 160, 169, 178
 Moreno-Ger, P., 120
 Morgan, B., xiii, 190, 209, 238
 Morgan, P., xiii, xiv, 28, 30, 42, 63, 91, 160, 190, 191, 209, 238, 291, 299, 316
 Morris, A.K., 104, 107
 Morrison, D., 29, 210
 Mostow, J., 29, 251
 Mott, Bradford, 33, 44, 54, 63, 149, 151, 366
 Moulton, K., 42
 Moy, L., 240
 Moyer, D., 44
 Mozer, M.C., 144
 Muggleton, S., 92
 Muldner, K., 278
 Mulvaney, R.H., 332
 Munro, A., 44, 45, 316
 Murray, R.C., 144
 Murray, Tom, viii, xiv, 6, 7, 9, 22, 23, 27, 29, 31, 34, 35, 40, 42, 44, 95, 96, 97, 99, 103, 107, 109, 121, 123, 136, 137, 143, 144, 151, 158, 160, 170, 178, 179, 191, 227, 255, 258, 259, 291, 296, 298, 299, 301, 303, 304, 305, 310, 312, 313, 314, 315, 316, 333, 334, 343, 366
 Mutter, S.A., xiv, 44, 45
 MyST, 196, 241, 242, 243, 244, 245, 246, 248, 249, 250
 NAEP, 177, 241, 252
 Nardi, B.A., 92
 Nash, P., 180, 191
 Nathan, M.J., 90
 National Research Council, 107, 371
 Naylor, S., 252
 Nelson, B., 106, 189, 190, 191
 Nelson, I., 42
 Nemet, F., 252
 Nesbit, J.C., 120, 251, 275, 280
 Newell, A., 92, 280
 Newman, S., 34, 43, 168
 Newman, S.E., 43
 Ngampatipatpong, N., 252
 Nguifo, E.M., 61, 238
 Nicholson, D., 44
 Niculescu, R.S., 168
 Nielsen, J., 7, 29, 290
 Nielsen, R., 4, 6, 7, 10, 29, 250, 286, 288
 Niraula, N., 149, 150, 240
 Nixon, T., 143, 144, 273, 279
 Nkambou, R., xiv, 61, 238
 Norman, D., 29, 290, 298
 Noy, N.F., 136
 Nulty, A., 180, 191
 Nussbaum, E., 252
 Nuzzo-Jones, G., 90
 Nye, Benjamin, v, xiv, 6, 47, 48, 49, 51, 52, 58, 62, 63, 67, 123, 133, 136, 147, 148, 149, 150, 179,

- 191, 199, 210, 223, 225, 227, 239, 301, 316, 330,
 332, 334, 343, 366
 Nyulas, C., 136
 O'Connor, P.E., 162, 168
 O'Donnell-Johnson, T.M., xiii
 O'Neill, E., 167
 O'Reilly, T., 110, 121
 Oakes, C., 333, 344
 Ocumpaugh, J., 148, 149
 Oezbek, C., 190
 Ogan, A., 62
 Ohlsson, S., 92, 239, 280
 Ohmaye, E., 44
 Oja, M.K., 29
 Olde, Brent, 149, 161, 366
 Oliveri, M.E., 169, 178
 Olney, Andrew, iii, xiii, 28, 32, 33, 43, 44, 56, 57, 62,
 120, 133, 135, 152, 159, 181, 190, 191, 196, 197,
 209, 227, 228, 231, 232, 234, 238, 239, 240, 269,
 280, 281, 367
 Olsen, J.K., 29, 92, 280
 Olson, W., 168
 Oppezzo, M.A., 238
 Oranje, A., 178
 Osborn, J., 252
 Osborne, J., 242, 251, 252
 Oser, R.L., 43
 Osin, O., 63
 Ososky, Scott, 257, 258, 283, 345, 367
 Ostrow, K., 92
 Ourada, S., 90, 177, 278, 297
 Ozuru, Y., 110, 121
 Paas, F., 350, 354
 Pahl, C., 164, 167
 Pain, H., 28
 Palinscar, A.S., 239
 Pallant, A., 108
 Palmer, M., 245, 248, 250, 252
 Pane, J.F., 137, 275, 280
 Panzoli, D., 160
 Pappas, C., 136
 Paquette, L., 148, 149
 Pardos, Z., 92, 168, 280
 Parvarezki, J., 144
 Pashler, H., 140, 144, 163, 168
 Patel, A.M., 42
 Patil, A.S., xiv
 Patvarczki, J., 63, 92, 281, 299
 Pausch, R., 294, 298
 Paviotti, G., 44
 Pavlik, Philip, 32, 44, 51, 61, 62, 91, 93, 120, 144,
 163, 168, 196, 209, 227, 235, 239, 269, 279, 280,
 367
 pedagogical agent, 77, 147, 151, 152, 153, 154, 155,
 156, 157, 158, 159, 241
 pedagogical model, iii, x, 9, 48, 201, 264, 292, 318,
 322
 Pekker, A., 28
 Pellegrino, J., 279
 Pelletier, R., xiii, 90, 238, 278, 315
 Pellom, B., 252
 Pennebaker, J.W., 186, 191
 Penumatsa, P., 186, 190
 Pereira, F., 168
 Perfetti, C., 50, 62, 163, 167, 228, 238
 Perrotta, C., 169, 178
 Persky, H., 169, 177
 Person, N., xiii, xiv, 62, 178, 238, 239, 240
 Petre, M., 90
 Petridis, P., 151, 160
 Petrosino, A.J., 251
 Piaget, J., 29
 Picard, R., xiv, 168
 Piech, C., 232, 240
 Pietrocola, D., 63
 Pinkwart, N., 62, 120, 239
 Piwek, P., 152, 153, 157, 160
 Pizzini, Q.A., 44, 311, 316
 Plaisant, C., 288, 290
 Podestá, M.E., 107
 Poliquin, A., 252
 Pollack, M.E., xiii, 259
 Popescu, Octav, 90, 257, 261, 278, 367
 Popovic, Z., 144
 Pradhan, S., 248, 252
 Preece, J., 287, 290
 Prendinger, H., 152, 153, 160
 Presson, N., 62, 239
 Preuss, S., 152, 160
 Priest, H., 45
 Prothero, W., 251
 Protopsaltis, A., 160
 Psotka, J., xiv, 44, 45, 299, 316
 psychomotor domain, xiv, 258, 346, 348, 349, 352,
 354
 Punamaki, R.-L., 28
 Pynadath, D.V., 152, 160, 225
 Qiu, L., 334, 343
 Quellmalz, E.S., 169, 178
 Radecki, L., 211, 225
 Raes, A., 102, 107
 Rafferty, A.N., 99, 107
 Ragusa, Charles, iv, vi, xii, xiv, 68, 123, 136, 342,
 368
 Rahman, M.F., 62, 136, 150, 343
 Rai, D., 278
 Raizada, R., 315
 Ramachandran, Sowmya, 40, 44, 257, 333, 368
 Ramaswamy, N., 90, 298
 Ramírez, J., 163, 167
 Ranney, M., xiv, 354

- Rau, M., 92, 280
 Ray, F., 44, 135, 160, 240, 331
 Raykar, V.C., 232, 240
 Razzaq, L., 63, 92, 144, 281, 299
 Rebolledo Mendez, G., 62
 Reder, L.M., 180, 189
 Redfield, C.L., 91, 280, 316, 334, 343
 Reed, S.K., 92
 Reeve, R., 30
 Regev, J., 251
 Rehak, D.R., 233, 238
 Reiser, B., xiv, 251, 354
 Remington, R.W., 28
 Resnick, M., 43, 291, 294, 299
 Reyna, V.F., 45
 Rice, W., 281
 Richard, K., 191
 Richards, F.A., 28
 Rickel J., 44
 Riedl, M.O., 44
 Riesbeck, C., 39, 43, 44, 334, 343
 Riesbeck, C.K., 43, 44
 Ringenberg, M., 92, 159, 225, 280
 Ringnér, H., 28
 Ritter, Steven, 9, 29, 54, 63, 68, 82, 84, 90, 92, 137, 138, 139, 140, 143, 144, 170, 177, 234, 240, 271, 272, 275, 277, 278, 281, 291, 293, 296, 297, 299, 368
 Rittle-Johnson, B., 89, 91, 280
 Roberts, R.B., 334, 343
 Robinson, D., 177
 Robinson, L.J.B., 225
 Robinson, S., 163, 168, 178, 211
 Robson, E., 44
 Robson, R., 44, 135, 160, 240, 331
 Rodrigo, M.T., xiii
 Rody, F., 42
 Roediger, H.L., 163, 167, 229, 240
 Rogers, Y., 287, 290
 Rohrbach, S., 92
 Rohrer, D., 16, 17, 28, 144, 163, 168
 Rohrer-Murphy, L., 28
 Roll, I., xiii, 63, 92, 120, 278, 281
 Romine, W.L., 181, 191
 Roscoe, R.D., 50, 52, 56, 61, 63, 110, 111, 120, 121
 Rose, C., 43, 149, 152, 159, 191, 225, 269, 273, 279
 Rosé, C.P., 43, 149, 191, 251, 278, 281
 Rosenthal, D., 267, 279
 Rosenzweig, L., 43
 Rossi, P.G., 44
 Roth, W.-M., 252
 Rotherham, A.J., 182, 191
 Row, R., 224
 Rowe, Jonathan, 33, 44, 54, 63, 110, 121, 149, 151, 368
 Roy, M.E., 334, 343
 Ruis, Andrew, 147, 150, 179, 189, 369
 Ruitenberg, M.F.L., 348, 354
 Ruiz-Primo, M.A., 101, 107
 Rummel, N., 29, 92, 280
 Rupp, A.A., 187, 191
 Rus, V., xiv, 45, 150, 160, 210, 240, 281
 Rusk, N., 299
 Russell, D., 45
 Ryder, J., 42
 Ryoo, K.K., 99, 100, 107
 Sadler, T.D., 180, 181, 191
 Salas, E., 43, 63, 167
 Salden, R.J., 63
 Samaddar, A.B., 28
 Samaddar, S.G., 28
 Samei, B., 201, 210
 Sampson, V., 90, 189, 252, 278
 Sancho, P., 120
 Sandler Training, 211, 225
 Sanghvi, B., 177
 Sani, S., 45
 Santarelli, T., 43
 Sato, E., 108
 Savova, G., 250
 Scardamalia, M., 27, 30
 Scarpinatto, K.C., 264, 279
 Schank, R.C., 45
 Schatz, S., 333, 344
 Schuble, L., 27, 190, 251
 Schellens, T., 102, 107
 Schifter, C., 181, 191
 Schmorrow, Dylan, 44, 149, 161, 369
 Schneider, P., 136
 Schön, D.A., 180, 191, 291, 299
 Schrider, P., 224
 Schroeder, N.L., 151, 160
 Schulze, K., 30, 63, 259, 299, 316
 Schwartz, D.L., 238, 291, 297
 Schwartz, S., 252
 Schweingruber, H., 251
 Scott, B., 39, 44, 189, 283, 345, 367
 Scott, R., 44
 Segedy, J., 276, 281
 Seip, J., 43
 Sengupta, P., 179, 189
 serious games, ix, xiv, 4, 6, 44, 59, 92, 121, 178, 224, 256, 257, 259, 299, 369, 371
 Settlage, J., 96, 107
 Sewall, Jonathan, 9, 27, 40, 42, 54, 61, 74, 75, 76, 77, 78, 89, 90, 91, 92, 143, 169, 177, 179, 181, 182, 189, 229, 238, 255, 257, 259, 261, 262, 263, 264, 278, 280, 291, 297, 298, 303, 309, 315, 334, 343, 369
 Shadbolt, N.R., 305, 315
 Shaffer, David, 21, 29, 147, 148, 149, 150, 179, 180, 181, 189, 191, 369

- Shapiro, J.A., 30, 63, 259, 299, 316
 Sharp, H., 286, 287, 290
 Shelby, R., 30, 63, 259, 299, 316
 Sheng, M., 93
 Sheridan, S., 224
 Sherwood, R.D., 42
 Shetty, S., 90, 298
 Shinkareva, S.V., 166, 168
 Shneiderman, B., 288, 290
 Shores, L.R., 63
 Shouse, A., 251
 Shute, V.J., 33, 45, 91, 112, 118, 121, 169, 178, 280, 292, 296, 299, 301, 313, 316, 333, 344
 Si, M., 160
 Silbergliitt, M.D., 178
 Silc, K., 103, 106
 Sille, R., 42
 Silverman, B.G., 62, 63
 SIMmersion, 218, 225
 Simmons, T.G., 211, 225
 Simon, H.A., 92, 189, 240, 280
 Simon, S., 75, 92, 180, 189, 229, 240, 252, 270
 Simperl, E.P.B., 163, 168
 Simpson, E., xiv, 354
 Sinatra, Anne, 144, 147, 149, 257, 283, 285, 290, 370
 Sinatra, G., 252
 Singh, S., 62
 Sitaram, S., 29
 Siyahhan, S., 189
 skills, iii, v, vii, xii, 3, 10, 12, 16, 19, 23, 28, 33, 34, 37, 42, 47, 48, 49, 50, 60, 63, 68, 77, 78, 79, 82, 86, 88, 97, 104, 112, 116, 121, 123, 128, 129, 137, 138, 141, 155, 162, 169, 170, 177, 178, 190, 191, 195, 196, 197, 211, 212, 213, 214, 215, 216, 218, 220, 222, 223, 224, 228, 229, 252, 256, 257, 262, 280, 292, 293, 295, 298, 301, 302, 304, 308, 310, 312, 331, 333, 334, 337, 339, 345, 346, 348, 349, 350, 351, 359, 362, 363, 365, 366, 368
 Skillsoft, 219, 225
 Skogsholm, A., 279
 Slack, K., 106
 Slamecka, N.J., 119, 121
 Sleeman D., xiv
 Sleeman, D.H., 43
 Slotta, J.D., 96, 99, 106, 108
 Small, M., 100, 118, 121, 347
 Smith, B., 43
 Snow, Erica, 68, 109, 110, 112, 118, 119, 120, 121, 370
 Snyder, L., 252
 So, Y., 20, 36, 37, 40, 49, 52, 77, 78, 169, 177, 197, 247, 265, 298, 340
 Soller, A., xiv
 Soloway, E., 29
 Song, Y., 169, 178
 Sorensen, B., 44
 Sotomayor, T.M., 133, 136
 Sotilare, Robert, i, iii, iv, v, vi, viii, x, xii, xiii, xiv, 1, 3, 7, 9, 29, 32, 35, 40, 41, 42, 43, 44, 45, 58, 59, 61, 63, 89, 92, 121, 123, 136, 142, 144, 148, 149, 150, 151, 159, 160, 177, 178, 179, 191, 209, 210, 222, 223, 225, 253, 255, 258, 259, 261, 278, 280, 281, 283, 290, 292, 296, 299, 302, 303, 316, 317, 332, 345, 346, 350, 354, 356
 Souders, V., 43
 Spain, R., 61, 332
 Sparks, J.R., 169, 178
 Specht, M., 29
 Spitulnik, M., 99, 106
 Squire, P., 167
 Stacy, W., 165, 167, 168
 Stagl, K.C., 63
 Stahl, G., 29
 Stamper, J.C., 143, 144, 163, 168, 231, 238, 240, 273, 279, 281, 334, 343
 Stampfer Wiese, E., 257, 259
 Stampfer, E., 92, 239
 Steenbergen-Hu, S., 281
 Stefanescu, D., 149, 150, 240
 Stein, S.A., 28
 Stensrud, B., 44
 Stevens, A., 28
 Stinson, L.L., 170, 178
 Stone, B.A., 125, 151, 159, 251
 Stuart, P.E., 181, 191, 222
 Stuart, S., 225
 Styler, W., 250
 Sulcer, B., 61, 281
 Suraweera, P., xiv, 7, 29, 30, 92, 93, 178, 191, 239, 280, 298, 316
 Surface, E.A., 224, 368
 Surmon, D.S., 44, 311, 316
 Susarla, S., 152, 160, 169, 178
 Sussman, G., 27
 Suthers, D.D., 271, 279
 Svihla, V., 100, 108
 Svirsky, E., 252
 Swaak, J., 343
 Swan, J., 225
 Swanson, H., 100, 108, 190
 Sweller, J., 350, 354
 Swiecki, Z., 189
 Taatgen, N., 63, 93, 281
 Tai, M., 278
 Tao, J., 177
 Tao, T., 136, 169
 Tarr, Ronald, 149, 234, 240, 258, 317, 318, 332, 370
 Taylor, L., 30, 63, 259, 299, 316
 Taylor, Robert, 149, 151, 370
 Tecuci, G., 307, 316
 Thille, C., 91, 144
 Thomson, D., 92

- Thomson, E., 92, 224
 Tian, Y., 232, 240
 Timms, M.J., 178
 Tinker, B., 108
 Tinker, R., 108
 Torreano, L.A., 313, 316
 Torrente, J., 120
 Towle, B., 143, 144
 Towne, D.M., 33, 44, 45, 316
 Towns, S.G., 151, 159
 Trafton, G., 62
 Trafton, J., xiv, 354
 training effectiveness, 136, 165, 314, 353, 354
 Traum, D., 224
 Trudeau, E.J., 28
 Tudorache, T., 136
 tutor model, iii, iv, 196, 247
 Tutoring Research Group, 7, 178, 209
 Underwood, J.D., 27, 143, 315
 Urban Brain Studios, 219, 225
 Urdan, T., xiii, 190
 usability, v, 4, 5, 6, 9, 10, 13, 14, 15, 16, 18, 20, 23,
 27, 29, 74, 96, 112, 130, 132, 135, 153, 175, 257,
 285, 286, 287, 288, 289, 290, 292, 296, 301, 305,
 314, 320, 331, 333, 342, 352, 360, 362
 user experience, 31, 32, 33, 132, 182, 183, 184, 185,
 257, 285, 286, 288, 331, 367
 user interface, iii, v, 5, 6, 9, 16, 35, 72, 89, 90, 125,
 128, 153, 183, 220, 223, 239, 258, 284, 288, 292,
 302, 304, 307, 334, 335, 342, 345, 352
 Valadez, G.H., 240
 Valente, A., xiii, 28, 43, 159, 224
 Van der Lubbe, R.H.J., 348, 354
 Van Eck, R., 152, 160, 169, 178
 van Joolingen, W.R., 44, 45, 343
 Van Lehn, K., 242
 Van Merriënboer, J., 350, 354
 van Merriënboer, J., 90
 Van Nice, J., 216, 225
 van Velsen, Martin, 90, 257, 261, 371
 van Vuuren, S., 252
 VanLehn, K., xiv, 30, 42, 43, 45, 62, 63, 93, 144,
 149, 160, 191, 240, 251, 252, 259, 281, 298, 299,
 316, 332
 Varma, K., 96, 104, 106
 Vartak, M., 55, 63, 92, 144, 281, 299
 Vattam, S.S., 291, 299
 Veden, A., 44, 240
 Veermans, K., 33, 45, 343
 Velsen, M.V., 89, 91, 280
 Ventura, M., 43, 120, 121, 178, 190, 209, 299
 Verwey, W.B., 348, 354
 visibility, 4, 5, 41, 132, 287, 288, 341, 349
 Vitale, J., 99, 108
 Vitányi, P.M., 28
 Voerman, J.L., 151, 159
 Von Ahn, L., 240
 von Davier, A., 178
 Voogt, J., 106
 Voss, J., 252
 Vuong, A., 141, 143
 Vye, N., 291, 297
 Vygotsky, L.S., 104, 108, 242, 252
 Waalkens, M., 63, 93, 281
 Wagner, A., 90, 279
 Wainess, R.A., 164, 168
 Walker, E., 62
 Walker, J., 49, 62, 252, 311, 316
 Walkington, C.A., 275, 281
 Wallace, P., 209, 210
 Waller, A., 28
 Walles, R., 151, 159
 Wang, T., 190
 Wang, W., 168
 Wang, X., 168, 169, 177, 234, 317, 318, 319
 Wang-Costello, J., 240, 332
 Ward, Wayne, 196, 197, 241, 243, 244, 250, 252,
 359, 371
 Warner, C., 250
 Warrant, S., 189
 Watson, A.M., 224
 Weatherly, R., 62
 Weaver, R., 63
 Weil, A.M., 45
 Weiss, A., 169, 177
 Weitz, R., 63, 279
 Weld, D.S., 91
 Wenger, E., 104, 180, 190
 Westerfield, G., 93
 Weston, T., 252
 Weyer, N., 63
 Wheeler, L., 299
 Wheeler, T.A., 168, 293
 Whitaker, E.T., 45
 White, B., 43
 Whitehead, E.J., 136
 Whitman, N., 224
 Wichmann, A., 100, 101, 108
 Widmer, C.L., 45
 Wiek, A., 62
 Wiemer-Hastings, K., 43, 209
 Wiemer-Hastings, P., 7, 43, 209, 210
 Wiggins, M., 136
 Wilcox, A., 211, 225
 Wilensky, U., 102, 104, 106, 108
 Williams, B., 27, 143, 315
 Williams, C., 62, 238, 239
 Williams, S.M., 42
 Williamson, C., 92
 Willingham, D., 182, 191
 Wing, J.M., 299
 Wing, R.M., 144, 295

- Winne, P.H., 27, 120
Wintersgill, M., 30, 63, 316
Wise, B., 252
Wobbrock, J.O., 27
Wogulis, J.L., 44, 316
Wolfe, C.R., 45
Wood, D., 190, 343
Wood, D.J., 143
Woods, A., 45
Woolf, B., xiv, 29, 30, 44, 63, 120, 159, 160, 191, 278, 316
Wray, R.E., 45
Wright, M., 11, 169, 178
Wu, S., 62, 239
Wylie, R., 93, 240
Xhakaj, F., 278
Xie, C., 102, 108
Yacef, K., 61, 91, 93, 120, 143, 144, 156, 159, 279
Yan, Z., 28
Yang, M., 62, 136, 150
Yaron, D., 264, 278
Yarzebinski, E., 315
Yates, K., 42, 90
Yng, M., 343
Young, R.M., 44
Yu, S., 240
Yudelson, M.V., 140, 143, 144
Zakharov, K., xiv, 7, 29, 93, 178, 191, 239, 280, 298, 316
Zap, N., 169, 177
Zapata, D., 112, 121, 149, 150, 169, 173, 177, 178, 296, 299, 371
Zapata-Rivera, Diego, 112, 121, 149, 150, 169, 173, 178, 296, 299, 371
Zarka, D., 44
Zhang, J., 30
Zhang, Z.H., 15, 18, 30, 99, 108
Zhao, L.H., 240
Zheng, J., 252
Zhu, J., 229, 232, 240
Zhu, X., 240
Zirkel, J., 42
Zoellick, C., 332
Zohar, A., 252
Zuiker, S., 189
Zwaan, R.A., xiv

Design Recommendations for Intelligent Tutoring Systems

Volume 3 Authoring Tools & Expert Modeling Techniques

Design Recommendations for Intelligent Tutoring Systems (ITSs) explores the impact of intelligent tutoring system design on education and training. Specifically, this volume examines "Authoring Tools and Expert Modeling Techniques". The "Design Recommendations book series examines tools and methods to reduce the time and skill required to develop Intelligent Tutoring Systems with the goal of improving the Generalized Intelligent Framework for Tutoring (GIFT). GIFT is a modular, service-oriented architecture developed to capture simplified authoring techniques, promote reuse and standardization of ITSs along with automated instructional techniques and effectiveness evaluation capabilities for adaptive tutoring tools and methods.



About the Editors:

- **Dr. Robert Sottilare** leads adaptive training research at the Army Research Laboratory and is a co-creator of the Generalized Intelligent Framework for Tutoring (GIFT). His research interests include team modeling, and adaptive architectures.
- **Dr. Arthur Graesser** is a professor in the Department of Psychology and the Institute of Intelligent Systems at the University of Memphis and is a Senior Research Fellow in the Department of Education at the University of Oxford.
- **Dr. Xiangen Hu** is a professor in the Department of Psychology at The University of Memphis and visiting professor at Central China Normal University. Dr. Hu received his Master (applied mathematics) from Huazhong University of Science & Technology, Master (Social Sciences) and Ph.D. (Cognitive Sciences) from the University of California, Irvine
- **Dr. Keith Brawner** is an adaptive training scientist at the U.S. Army Research Laboratory's SFC Paul Ray Smith Simulation & Training Technology Center. The focus of his research is authoring tools, learner modeling and adaptive architectures. Dr. Brawner is a co-creator of the Generalized Intelligent Framework for Tutoring (GIFT).

A Volume in the Adaptive Tutoring Series

