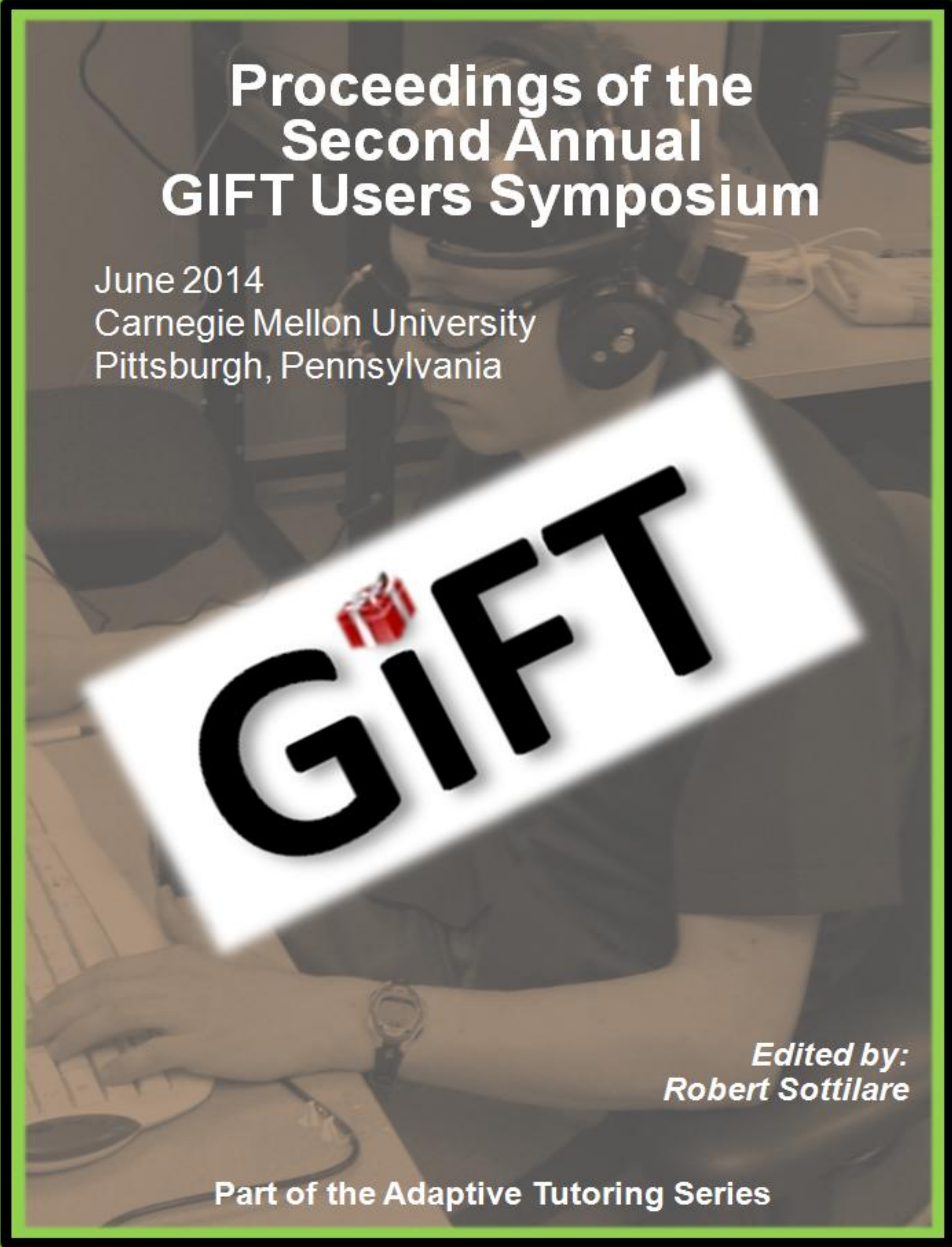
Proceedings of the Second Annual GIFT Users Symposium

June 2014
Carnegie Mellon University
Pittsburgh, Pennsylvania



Edited by:
Robert Sottilare

Part of the Adaptive Tutoring Series

Proceedings of the 2nd Annual Generalized Intelligent Framework for Tutoring (GIFT) Users Symposium (GIFTSym2)

Conducted 12–13 June 2014 at Carnegie Mellon University

Edited by:
Robert A. Sottilare

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

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

International Standard Book Number: 978-0-9893923-4-1

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

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

CONTENTS

Theme I: Authoring Intelligent Tutoring Systems

Examining Opportunities to Reduce the Time and Skill for Authoring Adaptive Intelligent Tutoring Systems

Robert A. Sottilare
U.S. Army Research Laboratory

Introduction

Intelligent tutoring systems (ITSs) as an instructional medium to provide one-to-one tutoring have been shown to be more effective learning tools than traditional classroom training (VanLehn, 2011; VanLehn, et al., 2005; Lesgold, Lajoie, Bunzo & Eggan, 1988) and more cost effective than providing one-to-one human tutoring in large organizations. It is estimated that it requires between 100 and 200 hours of a multidisciplinary team's time are required to author (develop) one hour of instruction for delivery by ITSs. Authoring teams can be generalized to include these highly skilled disciplines: computer scientists, instructional designers, human factors psychologist, learning specialists, and domain experts.

The time and skill required to author ITSs along with their associated high cost is a major barrier in their widespread adoption. This paper identifies authoring challenges and gaps in ITS research, and examines emerging and future opportunities to produce tools and methods (e.g., automation) that reduce the tedious process of authoring and make ITS authoring commonplace for the masses. Toward this purpose, our examination focuses primarily within two categories: methods to reduce authoring requirements and methods to automate authoring processes.

Recommendations are also provided for future development of authoring tools and standards within the Generalized Intelligent Framework for Tutoring (GIFT), an open-source tutoring architecture for authoring, automated management of instruction, and analysis of effect (Sottilare, Brawner, Goldberg & Holden, 2012; Sottilare, Holden, Goldberg & Brawner, 2013). GIFT development is being led by the U.S. Army Research Laboratory with contributions by leading academic, government, institutions in the ITS field. As of this writing, www.GIFTtutoring.org has over 400 users in 30 countries that provide a regular stream of guidance to improve GIFT's performance as an authoring, instructional, and analysis tool.

Authoring Goals for Intelligent Tutoring Systems

Our goal is to reduce the need for authoring where possible through interoperability standards and reuse. To augment this goal, we also want to reduce the knowledge and skill needed to author by developing job aids and other automation to support the authoring process. Based on these overarching goals, we pose the following authoring goals for ITSs, which have been adapted from Murray (1999), Murray (2003), Sottilare and Gilbert (2011), Sottilare, Goldberg, Brawner, and Holden (2012), and Sottilare (2013).

Decrease the effort of the author

It may be possible to decrease the effort to author ITSs by establishing and documenting standards for processes, tools, and reuse of components. Some existing technologies (e.g., GIFT, Cognitive Tutor

Authoring Tools) 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.

Another authoring goal is to establish standards to support rapid integration of external training or tutoring environments (e.g., serious games, virtual simulations, presentation content) to provide reuse of content and reduction of authoring effort. Standards for rapid integration of external environments will lower the need to create new content for practice and knowledge presentation, but will add the burden of matching appropriate content to course objectives. Care should be taken by authors to provide metadata to ease the search and location of appropriate content and scenarios by subsequent authors.

Decrease the skill threshold and help organize 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). Tools to aid the user in organizing their knowledge for quick recall and application can result in large authoring time savings.

Implement good design principles

This goal seeks to reduce the number of disciplines and the time required to author effective ITSs. An essential element of good design is the ability to support (i.e., structure, recommend, or enforce) good design principles. This will alleviate the author from the burden of knowing best pedagogical practices or optimal methodologies for user-system interaction.

Enable rapid assessment of prototypes

Enable rapid prototyping of adaptive tutoring systems to 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.

Challenges for Intelligent Tutoring Systems

Adaptive systems as opposed to adaptable systems change themselves to tailor interaction to meet user needs. Adaptive ITSs change instructional strategies (direction, support, feedback) in response to changes in learner states (cognitive, affective, and physical) to optimize their learning (knowledge and skill acquisition, retention).

Figure 1 shows how the ITS (tutoring agents and training environment) supports adaptive instruction of the learner. Per the zone of proximal development (ZPD) (Vygotsky, 1978), the tutor has options to increase/decrease challenge level in the training environment or increase/decrease the amount of support provided to bring the learner into a balance between the learner's competence and the difficulty of the problem space presented by the training environment (e.g., game, virtual simulation).

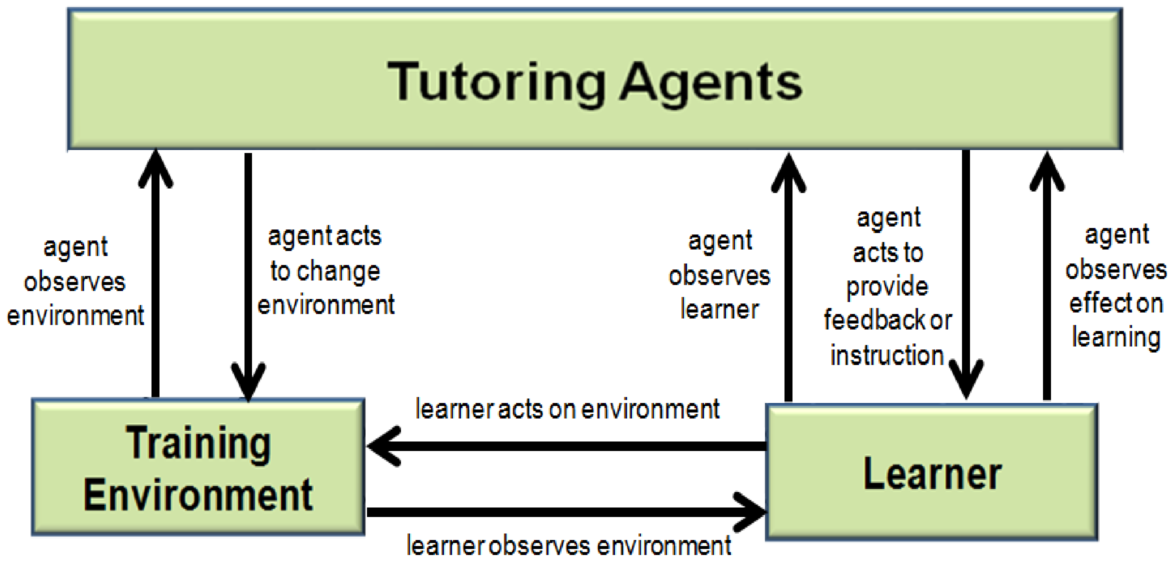

Tutoring Agents
agent observes environment
agent acts to change environment
agent observes learner
agent acts to provide feedback or instruction
agent observes effect on learning
Training Environment
Learner
learner acts on environment
learner observes environment

Figure 1. Interaction between the Learner and ITS (tutoring agents & training environment)

The ability of ITSs to adapt to the learner's needs comes at a cost. More adaptation requires additional domain content to support changes to the challenge level (e.g., increased complexity) in the training environment. New content may also come with requirements for authoring additional assessments. This authoring task is generally completed prior to runtime, but emerging technologies are evolving to automatically generate alternate scenario branches.

Another challenge for ITSs is the ability to shift from one tutoring domain to another. The complexity, definition (well or ill defined), and dynamics of domains that cover cognitive (e.g., decision making and problem solving), affective (e.g., moral judgment, emotional intelligence), and psychomotor tasks (e.g., marksmanship and golfing) vary widely so developing standards for domain models within ITSs is a major challenge.

Methods to Reduce Authoring Requirements

This section discusses methods that show promise in reducing ITS authoring requirements. As discussed earlier, a primary method of reducing authoring requirements is to promote reuse through interoperability standards and links to external tutoring and training environments (simulations, games, tools, and training content). Below are three methods demonstrated but not yet in widespread use.

Interoperability with serious games

The gateway module within GIFT provides an interaction standard for integrating external tutoring and training environments. The popularity of serious games which are virtual games used for training tactical tasks (e.g., land navigation) brings an element of high engagement to game-based tutoring. Game-based tutors use the immersive qualities of games along with the instructional best practices of tutors to provide adaptive one-to-one learning experiences. Serious game scenarios may be reused, copied, and modified rapidly in the game's scenario editor to support complex branching based on adaptation needs.

Figure 2 illustrates the type of data exchanged between the game interface layer and the tutor interface layer. Entity state, game state, and interaction data generated by the learner provide fodder for ITS instructional decisions while feedback and scenario change strategies recommended by the tutor are implemented as tactics in the game. Trials are ongoing to evaluate the effect of serious games with GIFT-based tutors and AutoTutor. To date, GIFT has been integrated with Virtual BattleSpace2 (VBS2), vMedic, and the Unity Game Engine. Standardization of data exchanged between the game and tutor interface layers is a near-term opportunity to increase interoperability and reduce authoring time to integrate unique tutor-game pairings. Automation of this process is another opportunity discussed in the "Methods to Automate Authoring Processes" section of this paper.

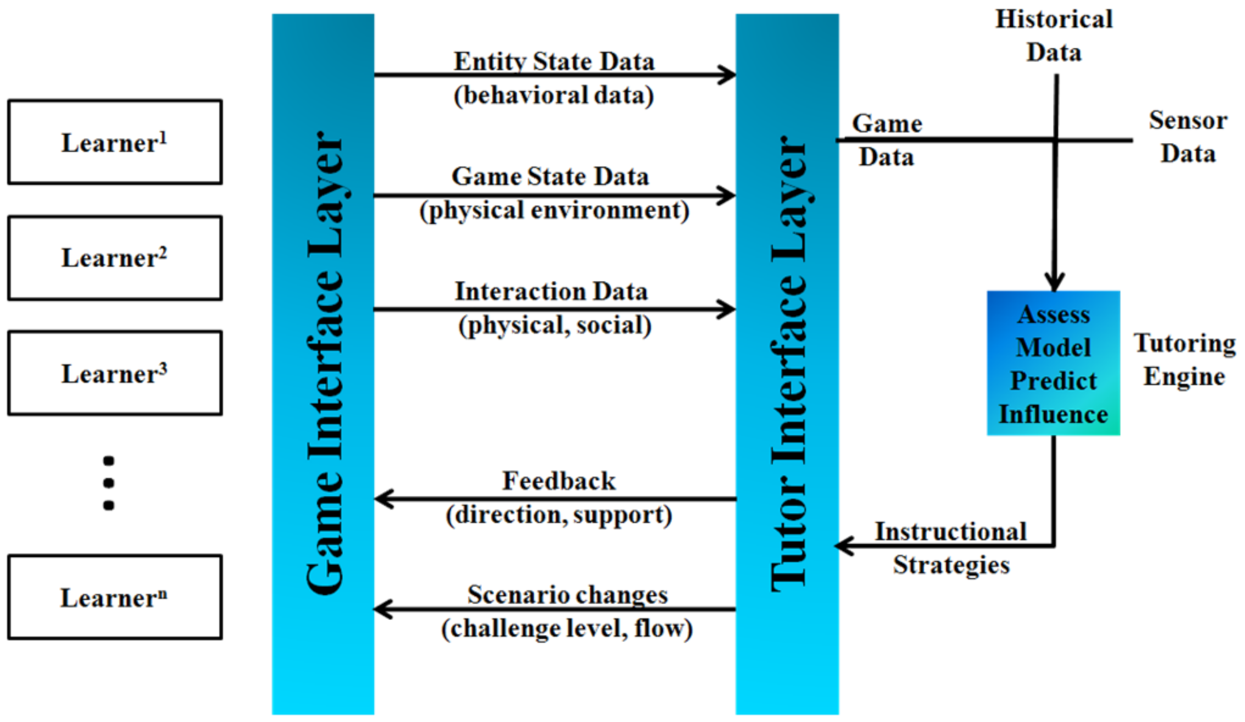

Learner1
Learner2
Learner3
Learnern
Game Interface Layer
Entity State Data (behavioral data)
Game State Data (physical environment)
Interaction Data (physical, social)
Feedback (direction, support)
Scenario changes (challenge level, flow)
Tutor Interface Layer
Historical Data
Game Data
Sensor Data
Assess Model Predict Influence
Tutoring Engine
Instructional Strategies

Figure 2. Notional Game-based Tutoring (Sottilare & Gilbert, 2011)

Interoperability with external web services

As part of recent releases of GIFT, ARL implemented calls to external AutoTutor webservices. Webservices available through GIFT support AutoTutor dialogue-based tutoring include: 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; and various other tutoring and delivery style mechanisms. Web service calls are data driven and therefore largely domain-independent. Interfaces have been standardized within GIFT to support interaction with commercial virtual humans via the Media Semantic character set. This addition to GIFT cuts out the need to program individual interactions between the tutor and the learner, and thereby reduces authoring load.

Interoperability with hardware and software-based data collection methods

Another feature of GIFT is the reusability of hardware interfaces. In some cases, it may be necessary to collect data about the learner to support the real-time tutoring decision process. These data might be collected through learner actions (e.g., speaking, typing) and a software-based capture method. ARL has also has created a surrogate software-based sensor, which can be used for sensitivity testing and experimentation in the absence of other sensors.

Data may also be captured by hardware-based sensors. Since its initial release, ARL has integrated several hardware-based sensors as part of its experimental development. These sensors currently include interfaces for the Emotive Epoch (commercial low-cost electroencephalograph (EEG)), the Affective Q-Sensor (commercial electro-dermal activity sensor), Microsoft Kinect (commercial sensor with software-based affect and physical state detection), and a host of physiological sensors.

Methods to Automate Authoring Processes

This section discusses methods that show promise in automating ITS authoring processes. By automating processes, we can lower the authoring load and knowledge required to author ITSs. For GIFT, the goal is to be able to provide tools suitable for non-computer scientists who are domain experts, the ability to author an ITS to support lessons in various domains (cognitive, affective, psychomotor, social, hybrid) in 50% of the time currently required for novice ITS authors. ARL envisions various degrees of automation from guided-authoring processes similar to the “TurboTax” experience to full automation. Below are two methods demonstrated but not yet in widespread use.

Automating the development of expert models

The concept automatically developing an expert model is commonly referred to as Tools for Rapid Automated Development of Expert Models (TRADEM) and it offers reduced time and skill, and thereby cost, to develop an essential part of the training domain without human knowledge of the domain. ARL is currently collaborating with Eduworks, a small business in Oregon, to develop tools and methods to automate the development of expert models for use by GIFT to produce adaptive tutors by datamining domain-specific instructional material using techniques to extract rules, principles, tasks, standards, conditions and hierarchical relationships from text in field manuals (Figure 3). Expert models, part of the GIFT domain module, are used to assess learner performance and the correctness of learner actions during tutoring sessions.

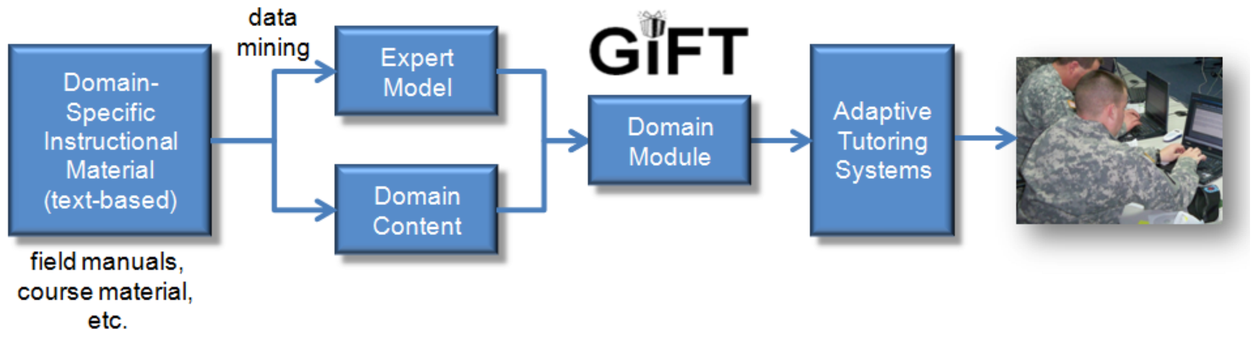

Domain-Specific Instructional Material (text-based)
field manuals, course material, etc.
data mining
Expert Model
Domain Content
GIFT
Domain Module
Adaptive Tutoring Systems

Figure 3. TRADEM process

Automating the development of middleware for integration of games and tutors

As mentioned in the "Methods to Reduce Authoring Requirements" section of this paper, the opportunity to automate the integration of games and tutors will combine higher levels of engagement found in serious games with the effective learning techniques found in tutors. ARL is collaborating with CHI Systems, a small business in Pennsylvania, to develop tools to automate the process of developing middleware to link serious games and ITSs. This middleware development tool is commonly known as Game-based Architecture for Mentor-Enhanced Training Environments (GAMETE) and this process will be integrated into GIFT in future versions.

Conclusions and Future Recommendations for Research

This paper identified opportunities for reducing authoring through standardization and interoperability to support increase reuse of ITS components and models, and through automation to reduce the time and skill required to author ITSs. Through GIFT and its user community, ARL is attempting to standardize components and interfaces to promote enhanced interoperability. ARL is also conducting research to generalize tools and methods to support reduced time/skill to produce ITSs which support cognitive, affective, psychomotor, social, and hybrid tutoring/training domains. GIFT also provides a serious of tools and templates to ease the authoring burden. Finally, GIFT is being architected to support coupling of generalized services, which can be made available to a variety of serious games and simulations to support self-regulated learning on demand. Additional research is needed to determine optimal methods for authoring:

- Techniques, strategies and tactics within GIFT
- Performance and competency assessments within GIFT
- Domain content and expert model development
- New content and instructional models from existing data sources

References

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.

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.

Sottilare, 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.

Sottilare, R., Brawner, K., Goldberg, B. & Holden, H. (2012). The Generalized Intelligent Framework for Tutoring (GIFT). US Army Research Laboratory.

Sottilare, 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.

Sottilare, R., Holden, H., Goldberg, B. & Brawner, K. (2013). The Generalized Intelligent Framework for Tutoring (GIFT). In Best, C., Galanis, G., Kerry, J. and Sottilare, R. (Eds.) *Fundamental Issues in Defence Simulation & Training*. Ashgate Publishing.

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

Vygotsky, L.S. (1978). Mind in Society: The development of higher psychology processes. Cambridge MA: Harvard University press.

Unwrapping GIFT: A Primer on Authoring Tools for the Generalized Intelligent Framework for Tutoring

Michael Hoffman and Charles Ragusa
Dignitas Technologies, LLC

Introduction

The Generalized Intelligent Framework for Tutoring (GIFT)1 is a framework and tool set for the creation of intelligent and adaptive tutoring systems (Brawner, 2012; Sottilare, 2012; Sottilare 2012, January). In its current form, GIFT is largely a research and development (R&D) tool designed to provide a flexible experimentation platform for researchers in the intelligent and adaptive tutoring field. However, as GIFT matures, it moves ever closer to becoming a production quality framework suitable for use in any fielded training systems.

In the past, tutors were generally built around a single domain (e.g., math) or training application/interface such as Bohemia Interactive's Virtual Battlespace 2 (VBS2). Since the tutors themselves were limited, their authoring tools were often simple, user friendly, and easy to learn. However, these attributes can heavily constrain the author when it comes to expanding the systems capabilities to other subjects. It is difficult to design an intuitive authoring tool that is flexible enough to support almost any domain. The authoring tools that claim to provide a way to author a course or assessment rules without programming often lack the ability to easily add logic associated with new domains, concept assessment rules, and a communication protocol that works with a new training application (TA) without programming. In addition, those tools often require a well-defined set of tasks/concepts (e.g., start with X, then do Y and finally Z) within a well-defined environment (e.g., interact with a form, a panel of interactive components or in a small 3D environment). Usually the simpler the authoring tool, in terms of authoring choices, the more constrained the user will be in creating a tutor that will deliver the desired content and instructional strategies. This can be an indication of the tutor's limited functionality across other domains and TAs.

Since GIFT is constantly evolving to support the needs and requirements of various organizations and domains, any authoring tools that are created need to be as accommodating. This paper provides an overview of the GIFT authoring tools including the initial design decisions and requirements, as well as the history and next generation of development to date with references to specific GIFT publically released files.

[1] GIFT users are encouraged to register on the GIFT portal at https://www.gifttutoring.org. The site provides access to the latest builds, source code, and documentation, and supports active forums for general discussion and trouble-shooting.

Design of Authoring Tools

During the early development stages of GIFT, we focused on developing the core architecture including communication and course management, as well as extensive event logging. In this phase, rudimentary courses were created to help drive Domain and Tutor module development in a domain-independent fashion. As we increased the functionality for each module, it was quickly discovered that all of the GIFT modules needed input files that would be used to not only alter a modules default settings but also load user-dependent (e.g., learning management system (LMS) history) and domain-dependent (e.g., course flow, performance assessment rules) configurations. Instead of trying to adapt one or more existing file formats often associated with tutoring (e.g., Sharable Content Object Reference Model (SCORM), Shareable Knowledge Object (SKO)), which may not have the necessary elements to describe every aspect of the GIFT modules, we decided to use the Extensible Markup Language (XML). XML provides a set of rules for encoding a document in a human and machine readable format and many application programming interfaces (APIs) have been developed to aid software developers. Moreover, it can be used across all of the GIFT modules by simply defining a different XML schema definition (XSD) for each. Each XSD could then be used to generate Java code (GIFT's native programming language) that makes it easy for a GIFT developer to use the defined elements found in an associated XML file. This tight integration between schema and source code greatly reduces errors and inconsistencies when a schema changes without changing the source code. Another benefit of the way GIFT utilizes XSDs is that it inherently defines a modules configuration definition. This definition can be used by third party applications to independently generate GIFT compatible XML files, thereby allowing users to still use authoring tools they are familiar with while expanding the suite of tools available to the GIFT community.

XML editor

Having a suite of schemas in place is useful as a developer; however, as GIFT evolved from an abstract design to a concrete implementation, it was apparent that we needed to shift a portion of our focus toward the usability of the GIFT authoring process. The initial goals of this endeavor were aimed at providing a graphical user interface for authoring that also included improved element validation, tips, and hints for complex elements, and most importantly, an interface that would require minimal software development as the schemas were continually being improved upon. User interface design can consume a large quantity of software development time. This is compounded when the back end logic is constantly changing and the application's requirements are ill defined. Having a dynamic authoring user interface allows more time to be spent on increasing the functionality of a system rather than the dealing with maintaining the layout and actions of graphical components.

After researching free XML editors, we decided on the Java application called XAmple XML Editor. XAmple provided the ability to dynamically analyze a given schema and then generate a document specific graphical user interface. It was more than a simple XML editor in that it provided additional tools such as invalid element identification, displaying of messages about selected nodes and the ability to associate custom dialogs on a per element basis. Within two weeks, we built an infrastructure in GIFT where, given a schema file, a new Java based authoring tool could be created in under five minutes. The tools inherently could load, create, edit, validate, and view an XML file associated with a different schema. From there the suite of XML based GIFT authoring tools was built. Each with the ability to dynamically update the authoring user interface based on schema changes without needing to recompile GIFT or having to restart the authoring tool. Over time additional improvements were made to the tools such as creating custom dialogs that could search for entries to select from (e.g., survey elements, concepts of a lesson, learner state attributes, performance assessments, auto generate IDs, file browsers), generalized custom inputs (e.g., Name/Value pairs) to limit software development and convert content from a previous version of GIFT to the latest version. The tools were evolving into more than mere XML editors.

Web-based authoring

Around the same time the GIFT XML authoring tools were being created, we had a requirement to also produce a survey authoring system (SAS). This application would not only replicate the features present in available survey authoring tools like "SurveyMonkey,"[2] but allow us to introduce tutoring-related components with the various survey elements. Since most, if not all, of the most popular survey authoring and delivery applications are web based, we decided to make the GIFT SAS a web application. There were several reasons for this decision. One of which was based on having a well-defined set of requirements for the application. This meant that the interface would remain relatively the same once built (when compared to the other GIFT authoring tools). Furthermore the GIFT development team could reuse some of the infrastructure in place for the web-based Tutor User Interface (TUI). This infrastructure is based on Google Web Toolkit (GWT),[3] which is a free, open source development toolkit for building and optimizing complex browser-based applications. With the goal of providing a productive development environment without the developer having to be an expert in browser quirks (e.g., how each browser type interprets different web based specifications), XMLHttpRequest, and JavaScript, GWT allowed us to write client-side applications in Java and deploy them with minimal effort. The development of the SAS identified several issues:

- Paging in survey elements when there may be hundreds if not thousands to choose from (e.g., questions in the question bank)
- Providing the ability to backup/restore the survey database
- Sharing survey elements across surveys to minimize the effort need to distribute minor changes (e.g., spelling/grammar mistakes).
- In addition, with the delivery of a web based application, we built upon GIFTs ability to handle client-server interactions.

Current Authoring Tools

The current suite of authoring tools has provided a plethora of configuration options across all GIFT modules, satisfying several research data collection and experiment efforts. There are three categories of

[2] For more information on SurveyMonkey refer to www.SurveyMonkey.com.
[3] For more information on GWT refer to www.gwtproject.org.

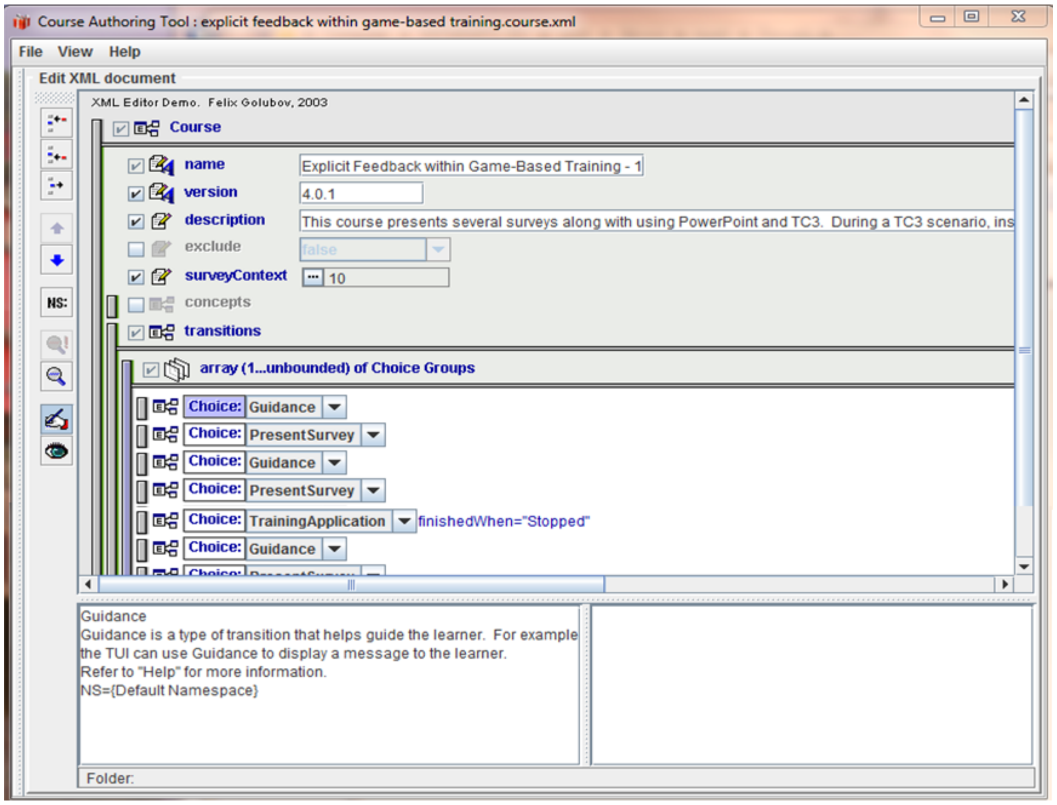
authoring tools in GIFT: XML based, web based and third party. Each category contains one or more applications that produce specific compositions for a GIFT session.

XML based

Course Authoring Tool (CAT)

The XML-based Course Authoring Tool (CAT) is used to author GIFT course XML files. A GIFT course consists of one or more course transitions (e.g., Guidance, Survey, Lesson Material, Training Application, Branch Point, and After Action Review (AAR)) along with some course metadata such as the unique course name and a description about the course. For more information on authoring a course, please refer to the previous paper in the "Unwrapping GIFT" series called "Unwrapping GIFT: A Primer on Developing with the Generalized Intelligent Framework for Tutoring." One way to start the CAT is by executing the GIFT/scripts/tools/launchCAT.bat script. With the interface that is presented, the user can choose to create a new course or open and then edit an existing course.xml file (for general information on the XML Editor dialog, please refer to GIFT/docs/content/GIFTXMLAuthoringTools.htm). Not only does the CAT provide a means to author a schema validated course.xml file but it also queries for the appropriate survey elements and displays Domain Knowledge Files (DKF) as well as HTML files for one to choose from when necessary. Without those types of customization, users would be left to read documentation and source code or merely guess at the appropriate values to provide for XML elements in the schema.

In **Error! Reference source not found.** the CAT is shown populated with an existing GIFT course. Here he flat, ordered list of transitions, shown in the large white section of the dialog, defines the user's experience through a course. In some instances, such as during experiments using GIFT, this fixed flow is often necessary. With the branch point transition; however, a course author can introduce the chance for a different course of action to be executed based on individualized learner states. Future versions of GIFT will expand on this component, making it concept based with a dynamic survey which tests the learner's comprehension along with the introduction of remediation to the GIFT system.


Course Authoring Tool : explicit feedback within game-based training.course.xml

Edit XML document

XML Editor Demo. Felix Golubov, 2003

Course
- name: Explicit Feedback within Game-Based Training - 1
- version: 4.0.1
- description: This course presents several surveys along with using PowerPoint and TC3. During a TC3 scenario, ins
- exclude: false
- surveyContext: 10
- concepts
- transitions

array (1...unbounded) of Choice Groups
- Choice: Guidance
- Choice: PresentSurvey
- Choice: Guidance
- Choice: PresentSurvey
- Choice: TrainingApplication finishedWhen="Stopped"
- Choice: Guidance

Guidance
Guidance is a type of transition that helps guide the learner. For example the TUI can use Guidance to display a message to the learner.
Refer to "Help" for more information.
NS={Default Namespace}

Folder:

Figure 4. The CAT showing the "Explicit Feedback within Game-Based Training – 1" course with the course metadata and the list of transitions exposed.

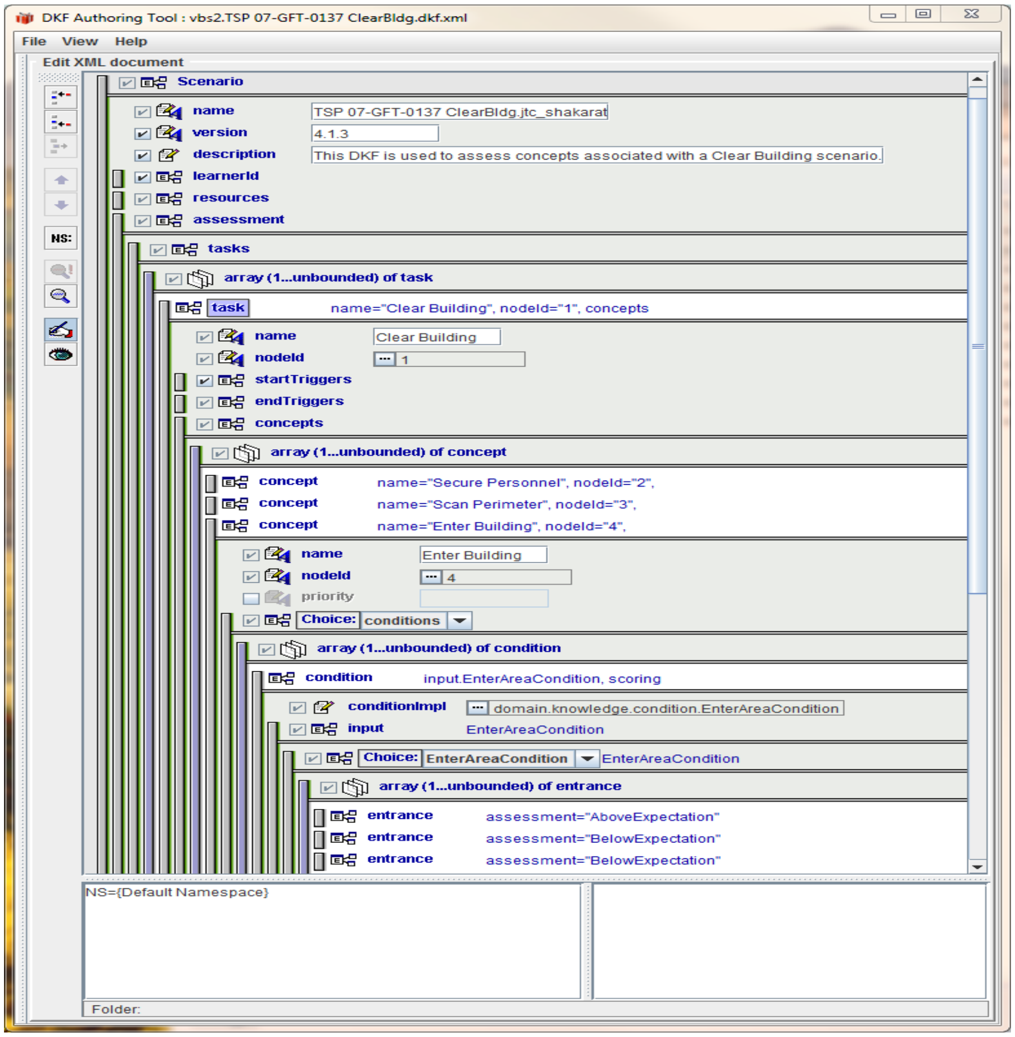
After authoring, validating and then saving the newly created course, the author is able to see the name of the course appear in the course selection webpage of the TUI (*helpful hint:* if the author is viewing the course selection webpage when first saving the course, refresh the webpage to force the list to refresh). One of the main issues course designers have with this tool is that it presents course transitions from the view of a GIFT software developer. Ideally, it should guide the author to layout material according to IMI levels, individual differences, desired cognitive/affective metrics and overarching course objectives.

DKF Authoring Tool (DAT)

The DKF authoring tool (DAT) is used to author DKFs in GIFT. One way to start the DAT is by executing the GIFT/scripts/tools/launchDAT.bat script. A DKF consists of the performance assessment rules for a lesson's concept map as well as the learner state transitions of interest and their associated instructional strategies. A DKF is used during a TA course transition (lesson). It configures one or more assessment engines that can run concurrently in the Domain module each time a TA transitions is encountered.

Within the concept hierarchy/map of the DKF structure are a set of conditions that describe how a concept can be assessed. The assessment rules are defined in a condition implementation Java class, which registers for game state messages of interest (e.g., Entity State) and provides an assessment result (Above, At, Below Expectation) that is then aggregated up the concept hierarchy. During the execution of a lesson in a course, the learner's state (cognitive/affective/performance) is expected to change based on a suite of inputs provided to GIFT. Based on the metrics the user can collect, be it sensor or training application based, a DKF author would then need to identify the learner state transitions that should result in the delivery of an instructional strategy (e.g., if the learner is bored the user may want to make the scenario more exciting as a form of micro-adaptation). In some instances, a defined state transition may happen more than once, in which case, the DKF allows the author to provide more than one instructional strategy to choose from. Furthermore, each strategy then has one or more tactics (i.e., a strategy implementation) to choose from. This flexibility allows GIFT to move away from a fixed state diagram and introduce a stochastic system with algorithms such as the Markov decision process.

One of the main shortcomings with this tool is in how it presents the heavily nested XML tree to the user. As per Figure 5, the XML authoring tool exposes the hierarchy of information in a tree-like structure synonymous with XML. However, this is not an ideal layout for many of the experts who need to author a DKF. Instead, a tool that intuitively organizes all of the necessary information (concepts, assessment rules, transitions between states, intervention tactics) in an author's field of view is profoundly coveted.

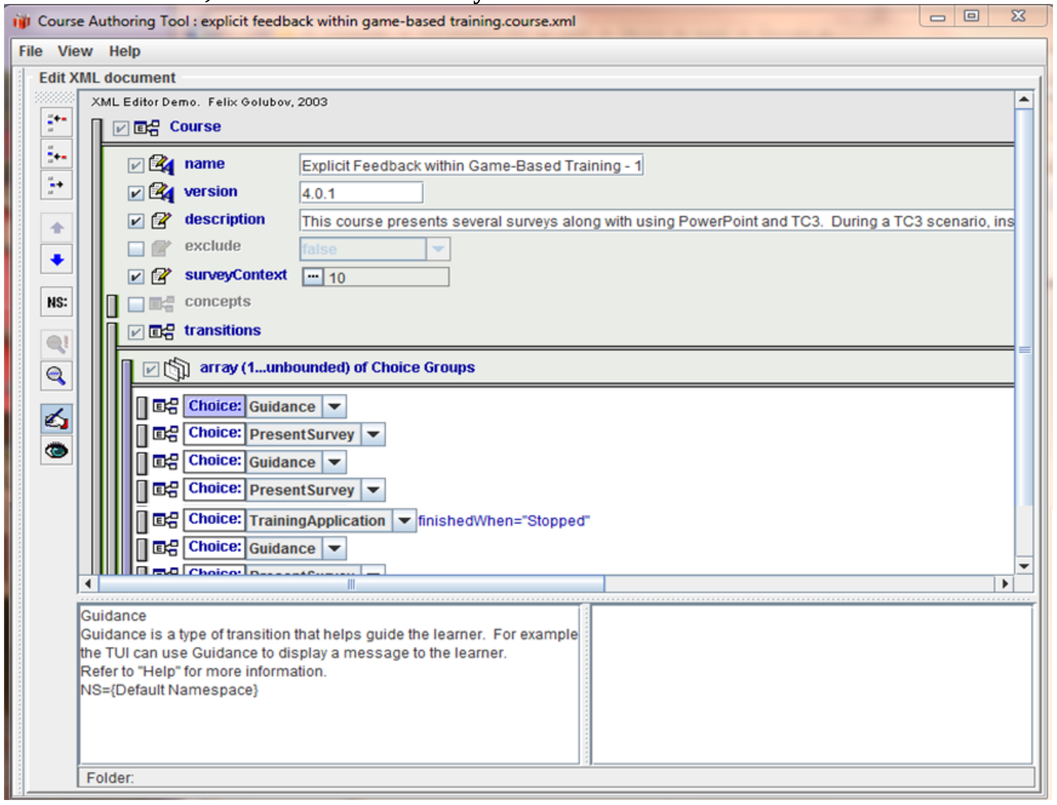


```
Scenario
  name            TSP 07-GFT-0137 ClearBldg.jtc_shakarat
  version         4.1.3
  description     This DKF is used to assess concepts associated with a Clear Building scenario.
  learnerId
  resources
  assessment
    tasks
      array (1...unbounded) of task
        task      name="Clear Building", nodeId="1", concepts
          name            Clear Building
          nodeId          1
          startTriggers
          endTriggers
          concepts
            array (1...unbounded) of concept
              concept     name="Secure Personnel", nodeId="2",
              concept     name="Scan Perimeter", nodeId="3",
              concept     name="Enter Building", nodeId="4",
                name        Enter Building
                nodeId      4
                priority
                Choice: conditions
                  array (1...unbounded) of condition
                    condition     input.EnterAreaCondition, scoring
                      conditionImpl   domain.knowledge.condition.EnterAreaCondition
                      input           EnterAreaCondition
                        Choice: EnterAreaCondition   EnterAreaCondition
                          array (1...unbounded) of entrance
                            entrance    assessment="AboveExpectation"
                            entrance    assessment="BelowExpectation"
                            entrance    assessment="BelowExpectation"
NS={Default Namespace}
Folder:
```

Figure 5. The DAT showing the Clear Building DKF contents. The XML tree is expanded to show the complicated nesting of information that is likely to happen.

Others

There is a suite of other XML authoring tools in GIFT that were quickly developed using the existing XML editor, each of which look very similar to the CAT and DAT show in



Course Authoring Tool : explicit feedback within game-based training.course.xml

File View Help

Edit XML document

XML Editor Demo. Felix Golubov, 2003

Course

name	Explicit Feedback within Game-Based Training - 1
version	4.0.1
description	This course presents several surveys along with using PowerPoint and TC3. During a TC3 scenario, ins
exclude	false
surveyContext	10
concepts	
transitions	

array (1...unbounded) of Choice Groups

Choice: Guidance
Choice: PresentSurvey
Choice: Guidance
Choice: PresentSurvey
Choice: TrainingApplication finishedWhen="Stopped"
Choice: Guidance

Guidance
Guidance is a type of transition that helps guide the learner. For example the TUI can use Guidance to display a message to the learner.
Refer to "Help" for more information.
NS={Default Namespace}

Folder:

Figure 4 and Figure 5, respectively. The Learner Configuration Authoring Tool (LCAT) is currently used to author learner configuration XML files (GIFT\config\learner\LearnerConfiguration.xml). That file describes the pipeline of how sensor data from the Sensor module is translated, classified, and then predicted upon (into current/future/predicted temporal categories) in order to produce learner state attribute values. As further learner state modeling research is implemented in GIFT, we expect this configuration file to undergo many improvements.

The Sensor Configuration Authoring Tool (SCAT) is used to author a sensor configuration XML file (examples in GIFT\config\sensor\), which configures the sensors that will be used in a Sensor module instance. Eventually, the Sensor module will be able to reconfigure itself on a per-user, per-course basis in order to collect the cognitive and affective metrics of interest.

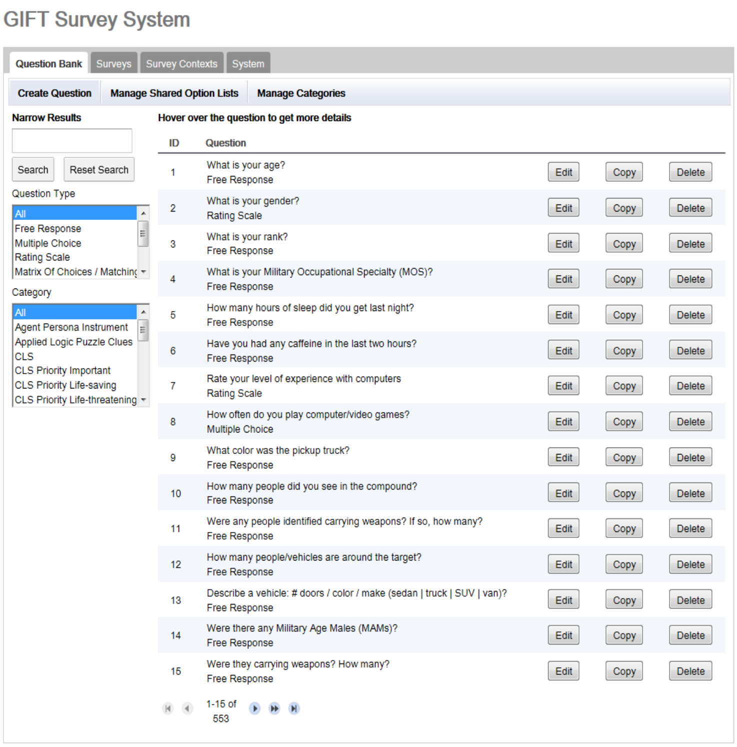
The Pedagogy Configuration Authoring Tool (PCAT) provides the ability to author a configuration file for the Engine for Macro and Micro Adaptive Pedagogy (eM2AP). This configuration describes the ideal metadata attributes of content to present in the various quadrants on Merrill's Component Display Theory (CDT).

The Metadata Authoring Tool (MAT) can create a metadata XML file (examples in Domain\Simple Branching Example\) that describes a single domain resource file (e.g., PowerPoint show) in a content dependent and learner state independent fashion. The metadata files are used by course branching logic and eM2AP to select the most appropriate and available content to present to the user based on the current learner state and empirical research.

Web based

Survey Authoring System (SAS)

The SAS was the first web-based authoring tool created for GIFT. It provides a means to author and compose surveys to present during the execution of a GIFT course. Surveys are built by selecting one or more questions from the shared question bank seen in. Figure 6. The question bank allows authors to reuse pre-existing and previously authored questions in various ways defined by a particular survey. After the survey copies portions of the question's elements, customizations to those elements "original values can be made such as altering the scoring rules". Unlike the XML-based authoring tools, the SAS provides a way to preview and test the author's progress. When a survey is previewed, several presentation details are ignored such as requiring questions to be answered, warning if questions are not answered, and question scoring. To execute those features, the user will want to test the survey instead. After authoring one or more surveys to use in a GIFT course, the next step is to create a survey context. A survey context is a grouping of surveys that can then be selected during course or DKF authoring via the CAT and DAT, respectively. Think of it as a way to not only filter the list of Surveys existing in your GIFT instance but also collate the survey question responses made under a given situation such as the execution of a particular course. For post analysis of a data collection or experiment event, this organization is beneficial in that it allows you to naturally identify the appropriate survey results to export.



ID	Question			
1	What is your age? Free Response	Edit	Copy	Delete
2	What is your gender? Rating Scale	Edit	Copy	Delete
3	What is your rank? Free Response	Edit	Copy	Delete
4	What is your Military Occupational Specialty (MOS)? Free Response	Edit	Copy	Delete
5	How many hours of sleep did you get last night? Free Response	Edit	Copy	Delete
6	Have you had any caffeine in the last two hours? Free Response	Edit	Copy	Delete
7	Rate your level of experience with computers Rating Scale	Edit	Copy	Delete
8	How often do you play computer/video games? Multiple Choice	Edit	Copy	Delete
9	What color was the pickup truck? Free Response	Edit	Copy	Delete
10	How many people did you see in the compound? Free Response	Edit	Copy	Delete
11	Were any people identified carrying weapons? If so, how many? Free Response	Edit	Copy	Delete
12	How many people/vehicles are around the target? Free Response	Edit	Copy	Delete
13	Describe a vehicle: # doors / color / make (sedan | truck | SUV | van)? Free Response	Edit	Copy	Delete
14	Were there any Military Age Males (MAMs)? Free Response	Edit	Copy	Delete
15	Were they carrying weapons? How many? Free Response	Edit	Copy	Delete

1-15 of 553

Figure 6. The SAS landing page that shows the question bank full of GIFT provided survey questions.

The SAS web-based interface provides a more intuitive and user friendly approach to authoring in GIFT. With the layout implemented, authors can easily flow through the survey authoring process and find the desired information in a timely fashion.

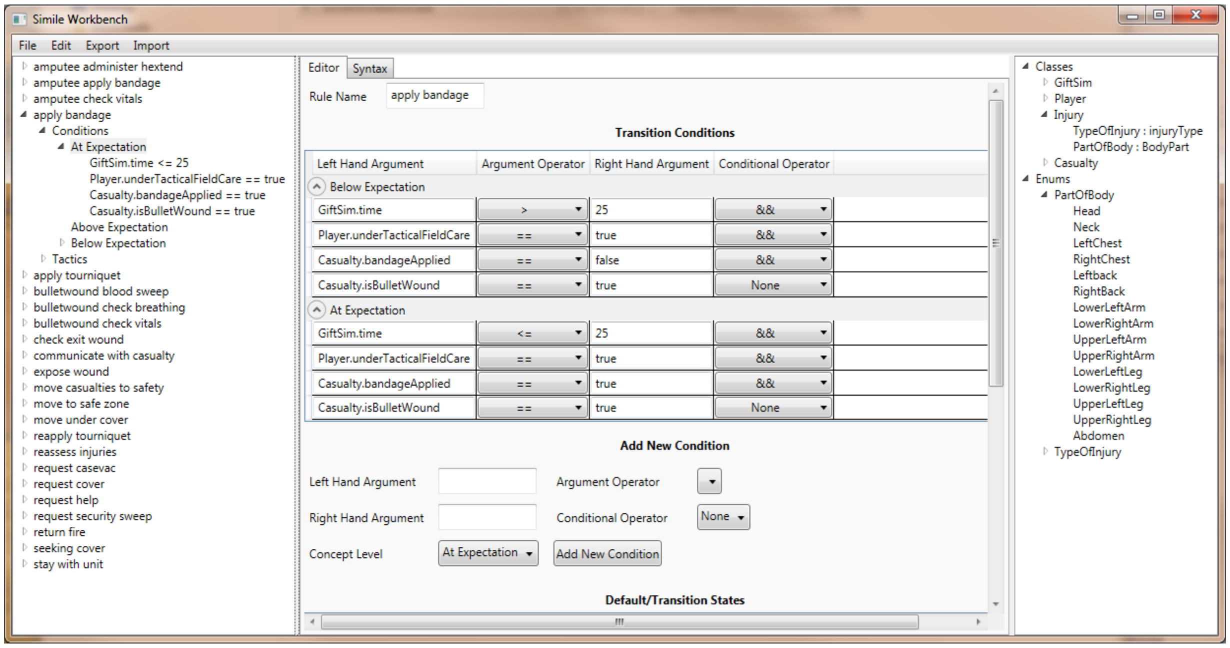
Third party

Beyond the core XML- and Web-based authoring tools, there are a few additional third-party authoring tools have been integrated and delivered with recent versions GIFT. We are always exploring other applications and hope the GIFT community will deliver additional tools as a means to improve upon the framework.

SIMILE Workbench

During the development of GIFT version 3.0, we integrated the SIMILE[4] assessment engine with the Domain module. This added a second assessment engine to what we now call the "Default Assessment Engine" in the Domain module. In addition, that task lead to identifying how future external performance assessment engines might integrate with GIFT. For SIMILE to execute, it requires a specific configuration file which contains the assessment rules for a lesson in a GIFT course. This file is authored using the SIMILE workbench shown in Figure 7. The workbench is a desktop application that provides an interface for users to easily create "assessment models" that account for the different relationships between the student and the simulation. It also allows training developers to tailor the way a student is evaluated without having to write code. As part of the first integration effort, the SIMILE integration with GIFT could only assess Tactical Combat Casualty Care[5] (TC3, aka vMedic) game state messages. In the near future, this coupling will evolve to allow users to assess any GIFT game state message without needing to alter the SIMILE engine or workbench. Furthermore, usability evaluations will identify aspects of the tool that need to be altered to lower the learning curve associated with authoring using the workbench.

[4] Visit http://www.ecsorl.com/solutions/simile for more information on SIMILE or use the forums on the GIFT portal at https://www.gifttutoring.org.
[5] Visit http://www.ecsorl.com/solutions/tactical-combat-casualty-care-simulation for more information on TC3 (vMedic).



- amputee administer hextend
- amputee apply bandage
- amputee check vitals
- apply bandage
  - Conditions
    - At Expectation
      - GiftSim.time <= 25
      - Player.underTacticalFieldCare == true
      - Casualty.bandageApplied == true
      - Casualty.isBulletWound == true
    - Above Expectation
    - Below Expectation
  - Tactics
- apply tourniquet
- bulletwound blood sweep
- bulletwound check breathing
- bulletwound check vitals
- check exit wound
- communicate with casualty
- expose wound
- move casualties to safety
- move to safe zone
- move under cover
- reapply tourniquet
- reassess injuries
- request casevac
- request cover
- request help
- request security sweep
- return fire
- seeking cover
- stay with unit

Rule Name: apply bandage

Transition Conditions

Left Hand Argument	Argument Operator	Right Hand Argument	Conditional Operator
Below Expectation			
GiftSim.time	>	25	&&
Player.underTacticalFieldCare	==	true	&&
Casualty.bandageApplied	==	false	&&
Casualty.isBulletWound	==	true	None
At Expectation			
GiftSim.time	<=	25	&&
Player.underTacticalFieldCare	==	true	&&
Casualty.bandageApplied	==	true	&&
Casualty.isBulletWound	==	true	None

Add New Condition

Left Hand Argument: | Argument Operator:
Right Hand Argument: | Conditional Operator: None
Concept Level: At Expectation | Add New Condition

- Classes
  - GiftSim
  - Player
  - Injury
    - TypeOfInjury : injuryType
    - PartOfBody : BodyPart
  - Casualty
- Enums
  - PartOfBody
    - Head
    - Neck
    - LeftChest
    - RightChest
    - Leftback
    - RightBack
    - LowerLeftArm
    - LowerRightArm
    - UpperLeftArm
    - UpperRightArm
    - LowerLeftLeg
    - LowerRightLeg
    - UpperLeftLeg
    - UpperRightLeg
    - Abdomen
  - TypeOfInjury

Figure 7. The SIMILE Workbench populated with the assessment rules for the "Explicit Feedback within Game-Based Training – 1" GIFT course.

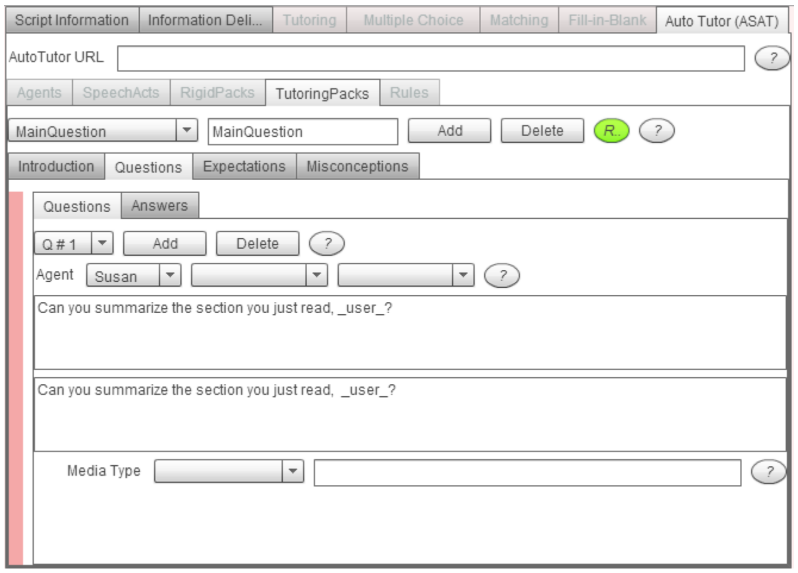
AutoTutor Script Authoring Tool (ASAT)

Around the same time that SIMILE was integrated with GIFT, we also integrated the AutoTutor Web Service (ATWS), which provides an interface to AutoTutor[6] (AT). AutoTutor is an intelligent conversational engine that allows a user to interact with an avatar to conduct a dialog that usually results in delivery and elicitation of information. In GIFT, an AutoTutor component is represented as a top-level course transition or a DKF condition[7]. The ATWS requires a conversation script in the format of a SKO to execute on. That script is authored using the AutoTutor Script Authoring Tool (ASAT). ASAT consists of content editing tools, templates for production rules, and a variety of additional functions that can be used to fill in conversation content based on existing rule templates, create new rules or integrate conversations with various forms of media. Each script usually centers on a specific topic such as "*An airplane flying horizontally drops a package when it is right above the target. Does the package hit the target?*" and terminates when the defined goal(s) of the conversation are reached.

During the first integration effort, only a web-based ASAT (WASAT), seen in Figure 8, was available to authoring SKOs. Since then a desktop version has been added to the GIFT release and a detailed set of authoring instructions has been created. One of the challenges in using the ASAT is when an author wants to create a self-reflection assessment. This happens to be the critical piece to author when using AT with GIFT. One of the usability challenges with this tool is the information it requires to author an AT SKO. Not only does it need a question to elicit an assessment but a semantic answer based on the user's semantic engine, tutoring feedback that considers number of "turns", feedback triggers for each turn and

[6] For more information on AutoTutor, refer to http://www.autotutor.org/ or use the forums on the GIFT portal at https://www.gifttutoring.org.
[7] See Domain/AutoTutorSession.example.dkf.xml in a the GIFT release for an example AT DKF configuration.

the relationship of the user's response corresponding to a percentage of coverage to relevant new, irrelevant new, relevant old, and total coverage of the desired answer.



MainQuestion

MainQuestion

Q # 1

Agent: Susan

Can you summarize the section you just read, _user_?

Can you summarize the section you just read, _user_?

Figure 8. The WASAT populated show a Question from a Tutoring Pack.

Next-Generation Authoring Tools

The future of GIFT authoring tools will be driven by the lessons learned and upcoming GIFT development, as well as through input from the community of users. As described previously, the current XML authoring tools are highly agile. Thus, when new GIFT requirements force changes to a schema, updating the corresponding authoring tool is relatively simple, and often entirely automatic and transparent. This agility is extremely powerful. By authoring just above the XML level, users of the current authoring tools also have complete flexibility to author across the entire spectrum of capabilities supported by GIFT as there are no artificial limitations imposed by the authoring tools. As such, these tools will necessarily remain part of the GIFT baseline for years to come.

However, despite the agility, flexibility, and engineering usefulness of these tools, they are still rather weak in the areas of usability and user-friendliness, especially when considered from the viewpoint of the most important end users: researchers, educators, course designers, and subject matter experts and not engineers or computer programmers. An abbreviated list of strategies for making the tools intuitive and easy-to-use is as follows:

- Hide unnecessary complexity
- Provide a logical presentation that makes sense to the user based on their role.
- Make reasonable assumptions about what user wants to do, providing most likely defaults where appropriate.

- Present related elements within the same view, and where possible avoid use of cryptic ID numbers and the like.
- Allow "Preview" functions where appropriate.
- Provide context sensitive help and input validation.
- Where appropriate, provide optional "Wizards" to guide users through difficult authoring steps.
- Engage target users to get feedback on design.

In the current suite of tools, individual tools are related, but not integrated. Thus, for example, a course author wanting to include a survey in their course, must first author the survey using the survey authoring tool, then from the (separate) course authoring tool, select that survey from an set of existing surveys.

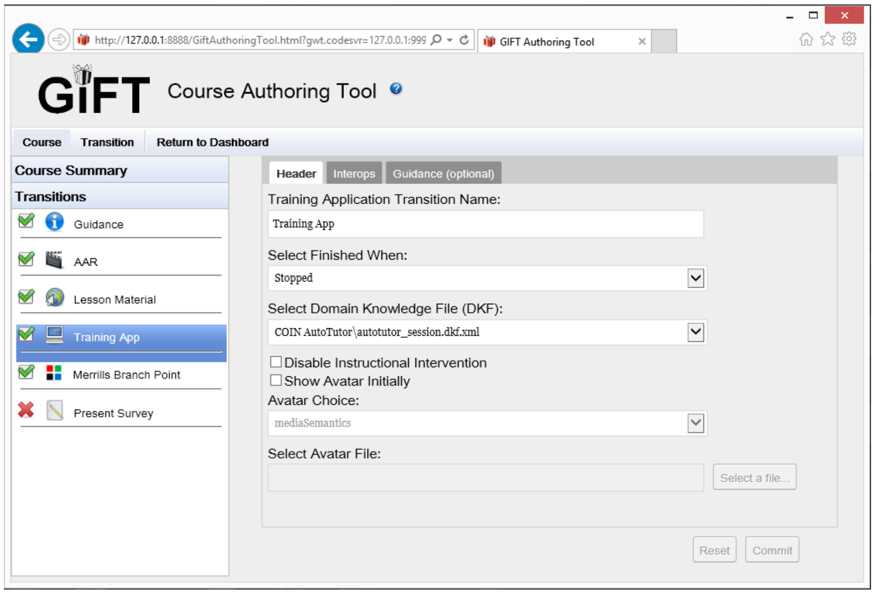
The next generation of authoring tools, seeks to address these key issues by creating a suite of web-based tools. Making the GIFT tools web based supports GIFT's near term objective of web-enabling GIFT as whole. In this scenario, users/authors will log into a GIFT web site and access tools through their browser, rather than launching desktop applications on their workstations. This approach has several benefits:

- Multiple users can share a single GIFT installation, either a site- or cloud-based, thus relieving individual users from having to setup and maintain their own GIFT installation.
- Multiple users can collaborate on authoring a single course, allowing each authoring participant to contribute according to their individual expertise
- By implementing user roles, we can tailor the user interfaces to expose or hide functionality appropriate to current user's role

The preferred approach and the one we are seeking in the next generation of authoring tools, is to have the entire tool suite presented as a single "GIFT Authoring Tool" (GAT) that allows user to move among the various tools seamlessly and effortlessly.

GIFT Authoring Tool (GAT)

The initial version of the web-enabled CAT included in GIFT 2014-1X and seen in Figure 9, is a fully functional browser-based course authoring tool, suitable for immediate use by GIFT users. The new CAT meets several (but not all) of the desired objectives and we believe it to be substantially easier to use than the legacy CAT.



Training Application Transition Name: Training App

Select Finished When: Stopped

Select Domain Knowledge File (DKF): COIN AutoTutor\autotutor_session.dkf.xml

☐ Disable Instructional Intervention

☐ Show Avatar Initially

Avatar Choice: mediaSemantics

Select Avatar File:

Figure 9. The web-based CAT showing an instance of a TA course transition. Notice how this interface is more user-friendly and graphically oriented versus the XML based authoring tools used by GIFT developers.

Details on the use of the new CAT are beyond the scope of this document; however, we will call out a few of the key features. Top-level operations are driven by the menu-bar at the top of the work area. The file menu allows for creating and saving courses. The transition menu allows for adding, deleting, and reordering transitions. At any time the user is working with transitions, a list of all transitions appears in a scrollable/selectable list along the left side of the view. Selecting a transition loads the transition into an editor panel that is presented in the main/center panel. For consistency and convenience, context sensitive help appears just above the editor pane for any element selected with the mouse.

Though eminently useable, at this time the new CAT still falls short of the long term goals. Currently (despite the fact that it runs in the browser), the CAT is not ready to be deployed to the web. Furthermore it does not support login, roles, or multiple concurrent users. Ideally we hope to at least come close the agility offered by the legacy authoring tools, so as to minimize any delay between new GIFT functionality and corresponding authoring tool support. Authoring tools are a high priority but not the only priority as we're also expanding the functionality of GIFT. Nevertheless, by the end of 2014 we plan to have a user-friendly, browser based tool for DKF authoring. The next step beyond that would be to seamlessly integrating the web based interfaces with the other existing tools such as the SAS. Look for these improvements as well as implementations that mimic the other existing XML authoring tools in upcoming releases of GIFT.

Conclusion

Although user friendly interfaces are desired, there will still be an active and ever present need for agile XML authoring tools in GIFT. Once considered a step above creating XML files in a text editor, these tools will maintain a direct relationship with the latest research and development. This is mainly because user friendly tools are often incapable of adapting easily to new and innovative features. Ideally, a web based approach to authoring is preferred as it appears to offer an easier way to integrate with other third party training applications in addition to enabling mobile authoring through a webpage. With GIFT, any number of tailored user interfaces can be built without having to duplicate the underlying data and rules mostly defined by the set of XML schemas. The features that must be provided (tools to help author a course) can be presented in many different variations having been built on a common model. Although, before committing to a complex model, be sure there is a reasonable path to an authoring tool. After all, the application may be a good idea on paper, but if the community is not able to use it, what good is it?

References

Brawner, K., Holden, H., Goldberg, B. & Sottilare, R. (2012, January). Recommendations for Modern Tools to Author Tutoring Systems. In The Interservice/Industry Training, Simulation & Education Conference (I/ITSEC)(Vol. 2012, No. 1). National Training Systems Association.

Ragusa, Charles, Hoffman, Michael, Leonard, Jon (2013). Unwrapping GIFT: A Primer on Developing with the Generalized Intelligent Framework for Tutoring. In AIED 2013 Workshops Proceedings Volume 7 (http://ceur-ws.org/Vol-1009/aied2013ws_volume7.pdf)

Sottilare, R. A., Brawner, K. W., Goldberg, B. S. & Holden, H. K. (2012). The Generalized Intelligent Framework for Tutoring (GIFT).

Sottilare, R. A., Goldberg, B. S., Brawner, K. W. & Holden, H. K. (2012, January). A Modular Framework to Support the Authoring and Assessment of Adaptive Computer-Based Tutoring Systems (CBTS). In The Interservice/Industry Training, Simulation & Education Conference (I/ITSEC)(Vol. 2012, No. 1). National Training Systems Association.

SimStudent: Authoring Expert Models by Tutoring

Christopher J. MacLellan, Eliane Stampfer Wiese,
Noboru Matsuda, and Kenne R. Koedinger
Carnegie Mellon University

Introduction

Intelligent tutoring systems are effective at improving learning (Koedinger & Anderson, 1997; Pane, Griffin, McCaffrey & Karam, 2013; Ritter, Anderson, Koedinger & Corbett, 2007; Vanlehn et al., 2005), but development costs remain a formidable obstacle to their general adoption (Sottilare & Holden, 2013). As an example, despite widespread use of math Cognitive Tutors (more than 500k students per year complete a Carnegie Learning tutor course), they have not been widely used more broadly (e.g., in online education platforms such as Khan Academy, Coursera, etc.), perhaps because their learning benefits are not thought to outweigh the costs of their development. Authoring costs are particularly pronounced for massive online education platforms, which have large quantities of content that vary widely across domains (Khan academy has about 500 hours of videos spread over 40 units in Math, Science, Economics, and Humanities). Similar to the goals that motivate the Generalized Intelligent Framework for Tutoring (GIFT), our work aims to increase the value of intelligent tutoring systems (ITSs) by improving both sides of the cost-benefit equation to build higher quality tutors that lead to more robust learning while also decreasing authoring time.

SimStudent is an outgrowth of the Cognitive Tutor Authoring Tools (CTAT) (Aleven, McLaren, Sewall & Koedinger, 2006; 2009). CTAT provides tools for constructing drag-and-drop tutor interfaces and authoring Example-Tracing Tutors, and creating an expert model for Model-Tracing Tutors. The Model-Tracing Tutor is more general, but more costly to produce. Authoring in this paradigm consists of manually constructing production rules that define which actions are appropriate given the current problem-solving state, e.g., if there is a constant on both sides of the equation, then subtract one of those constants from both sides. These production rules can generalize to a wide range of problems, as long as the if" part of the production rule is applicable. Authoring an Example-Tracing Tutor consists of demonstrating every possible action for every state directly in the tutor interface (e.g., for the equation 4 + x = 2x - 5, the author would demonstrate subtracting 4 from both sides). These demonstrations compose a behavior graph, which specifies which actions are legal in each state. While authoring these tutors is generally much easier (students can learn to build example-tracing tutors in an afternoon (Aleven et al., 2009)), they are much more specific than Model-Tracing Tutors. Demonstrations with Example-Tracing tutors can be generalized to new problems that share the same underlying structure (e.g., demonstrating 4 + x = 2x-5 could be generalized to 10 + x = 3x - 6 but not to 4 + x = 5) using a technique called “mass production,” but problems with different structures require additional demonstrations. These types of tutors are two ends of a spectrum: Model-Tracing Tutors are difficult to produce, but they are quite general; and Example-Tracing tutors are easy to produce, but are quite specific. Our goal is to combine the best of both worlds in an authoring tool that makes general tutors easy to build.

SimStudent, our authoring system (Matsuda, Cohen & Koedinger, n.d.), uses machine-learning techniques to try and bridge the gap between Example-Tracing Tutors and Model-Tracing Tutors. It does this by learning general production rule models from demonstrations and feedback. In this paper, we summarize how this system works, give a step-by-step example of how a tutor might be authored with SimStudent, and discuss the different lines of research we are currently pursuing with SimStudent.

The SimStudent Architecture

SimStudent[8] was created for three purposes: 1) to advance theories of human learning, 2) to explore the learning-by-teaching phenomenon, and 3) to improve the authoring of intelligent tutors. We briefly review the SimStudent architecture, discuss prior findings, and then describe how SimStudent can be used to author tutors.

SimStudent learns from four sources of knowledge (Matsuda et al., n.d.) (Figure 1). Feature predicates and primitive functions are built before SimStudent starts learning, and User Feedback and User Demonstrations come from SimStudent's learning environment. First, SimStudent needs to recognize relevant features of the tutor interface (e.g., numbers, operators, the equals sign). These "feature predicates" are constructed by writing small Java functions, the equivalent of writing regular expressions, to identify key features in the interface. Second, SimStudent starts with a certain level of prior knowledge (e.g., SimStudent for algebra can add two numbers at the beginning); these "primitive functions" are also small Java functions, similar to basic Excel formulas, for performing mental and interface actions. Third, within the learning environment, SimStudent is provided with "user demonstrations." These consist of the author solving sample problem steps. Fourth, SimStudent learns from "user feedback", which is yes/no correctness feedback when it attempts steps in a problem. After tutor problems have been demonstrated, SimStudent will learn new rules, attempt to apply them to new problems, and ask the author for verification that the rules were applied correctly. Based on this author feedback, the condition statements of these rules are refined.

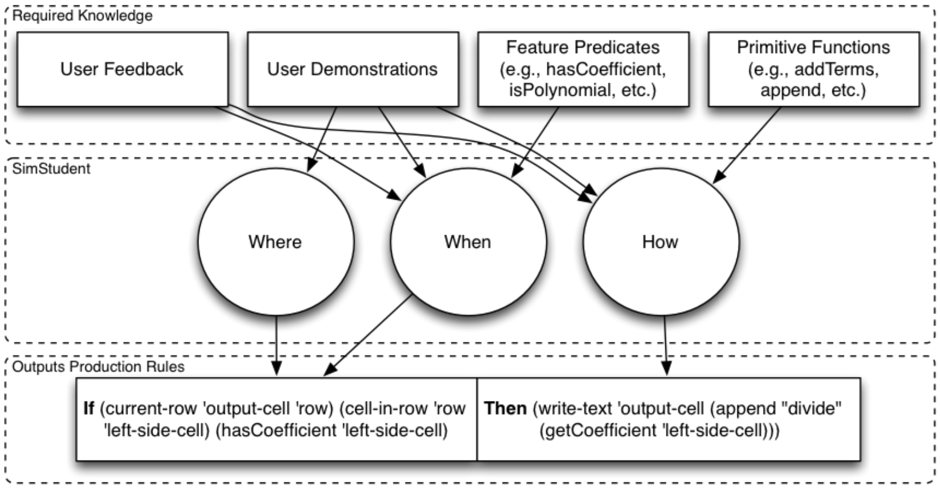

Required Knowledge
User Feedback
User Demonstrations
Feature Predicates (e.g., hasCoefficient, isPolynomial, etc.)
Primitive Functions (e.g., addTerms, append, etc.)
SimStudent
Where
When
How
Outputs Production Rules
If (current-row 'output-cell 'row) (cell-in-row 'row 'left-side-cell) (hasCoefficient 'left-side-cell)
Then (write-text 'output-cell (append "divide" (getCoefficient 'left-side-cell)))

Figure 10. The knowledge (squares) and learning processes (circles) used by the SimStudent system.

[8] For more details on SimStudent see http://www.simstudent.org/

Given these four sources of knowledge, SimStudent employs three learning mechanisms to produce general production rules. These three types of learning are called “How” learning, “Where” learning, and “When” learning. How learning identifies sequences of primitive function operators that would have plausibly produced the user demonstrations (e.g., going from 4+4x = 5 to 4x = 1 could be caused by subtracting the constant “4” from both sides or by subtracting the coefficient of “x” from both sides). How learning generates the “then” part of the production rules. Where learning identifies which interface elements are relevant to each demonstration, (e.g., learning that all the interface elements in the last used row are relevant). Lastly, When learning identifies the conditions under which a given sequence of operators is applicable. The Where and When learning jointly produce the “if” part of a production rule. As the author demonstrates problem steps, the three mechanisms learn new production rules. Once production rules are learned, SimStudent attempts to use those rules to solve practice problems. The rules are refined when the author provides correctness feedback on each step of the problem.

SimStudent enables us to test if the How, Where, and When mechanisms are reasonable approximations of how human students learn from demonstration and feedback. Indeed, empirical work indicates that models generated by SimStudent better fit student tutor data than models hand authored by domain experts (Li, Stampfer, Cohen & Koedinger, 2013). These results were replicated across three different domains (algebra, stoichiometry, and fraction addition). SimStudent may produce better results because it is less susceptible to “expert blind spots” (Nathan, Koedinger & Alibali, 2001) than domain experts. These blind spots refer to knowledge that an expert doesn’t realize they know. For example, a domain expert might view -x = 4 and -1x = 4 as equivalent, but the SimStudent model recognizes that additional knowledge is needed in the first case because the -1 coefficient is implicit. Improved student models are likely to result in better student learning (Koedinger, Stamper, McLaughlin & Nixon, 2013) because they guide interface design, problem selection, and assessment of student knowledge. Continuing the example above, the original model for students’ extraction of a negative coefficient lumped -x together with -3x, -5x, etc. That model assumes that practice on any of those examples would lead to improved performance on other examples within the group. In contrast, the SimStudent model would provide additional practice for -x and would not assume automatic transfer from -3x to -x. These findings, that SimStudent can create better models and that better models result in better student learning, show promise for leveraging SimStudent to create more effective tutors.

In addition to theory building, SimStudent has been used as a teachable agent. Instead of asking students to learn directly from the tutor, students are tasked with teaching SimStudent so that it can pass a quiz on the domain content. The learning-by-teaching paradigms aim to take advantage of the “protégé effect,” so called because students have been found to be more motivated to learn on behalf of a teachable agent than to learn for themselves (Chase, Chin, Oppezzo & Schwartz, 2009). Results (Matsuda et al., 2010) suggest that learning-by-teaching is as effective as a Cognitive Tutor for students who have reached a basic level of competency. This work seems to imply that we don’t need an expert model to teach students since they can learn simply by teaching the SimStudent agent. However, the students are still receiving feedback on how the SimStudent agent does on each quiz, and grading the quizzes is done using an expert model. Therefore, it is still necessary to author good expert models, even in a learning-by-teaching paradigm.

A third line of SimStudent research investigates the authoring of expert models for use in both tutoring systems and teachable agents. One study found that higher quality models are produced by providing SimStudent with both demonstrations and feedback, compared with only giving it demonstrations (Matsuda et al., n.d.). A follow up study showed that authoring an Algebra tutor with SimStudent is more efficient than authoring an equivalent tutor using Example Tracing and that the model learned by SimStudent is more general (MacLellan, Koedinger & Matsuda, 2014), when the background knowledge had already been authored. Overall, research on the SimStudent system suggests that it might be a viable tool for efficiently authoring tutoring content that is general and of high quality.

An Example of Authoring with SimStudent

Authoring with SimStudent is similar to authoring with CTAT. Details of authoring a tutor with CTAT are written up elsewhere (http://ctat.pact.cs.cmu.edu/), so we focus on the aspects of authoring that are unique to SimStudent: authoring background knowledge and tutoring the SimStudent system interactively. This example shows how to construct a simple algebra tutor using SimStudent.

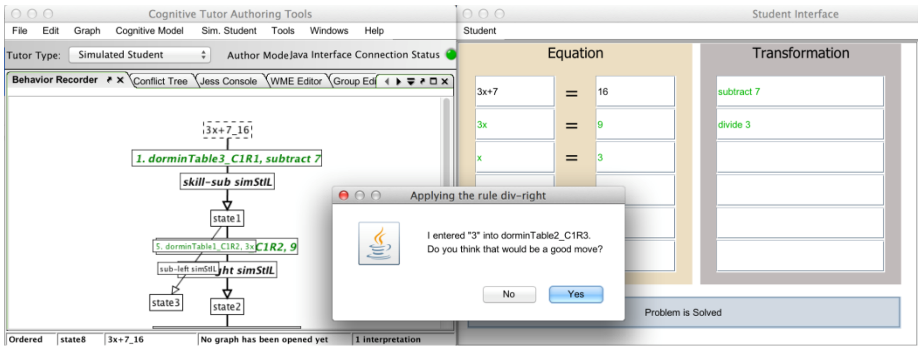
Authoring background knowledge

The SimStudent system separates the authoring of background knowledge from the construction of the expert model. Constructing the background knowledge requires basic programming skills; since the expert model is created through interactive tutoring, it requires no programming at all. The first class of background knowledge, feature predicates, are small Java functions that return True if a feature is present in an interface element and False otherwise. One example might be the “HasCoefficient” feature, which would be True for 3x but False for x + 1. SimStudent uses feature predicates to recognize important features in the tutor interface. For the algebra domain, we have authored 16 feature predicates. These predicates tend to be relatively general, so they can be reused from one tutor to the next.

The second class of knowledge, primitive function operators, are similar to the feature predicates, in that they are small java functions, but they take two inputs (taken either from interface elements or from the outputs of other primitive function operators) and return a single value. One example of a primitive function operator is “AddTerm”: when given two numbers it returns their sum. These operators enable SimStudent to explain demonstrations and take actions in the tutor interface. For the algebra domain, we have authored 28 primitive function operators. Similar to feature predicates, primitive functions tend to be reusable across tutors.

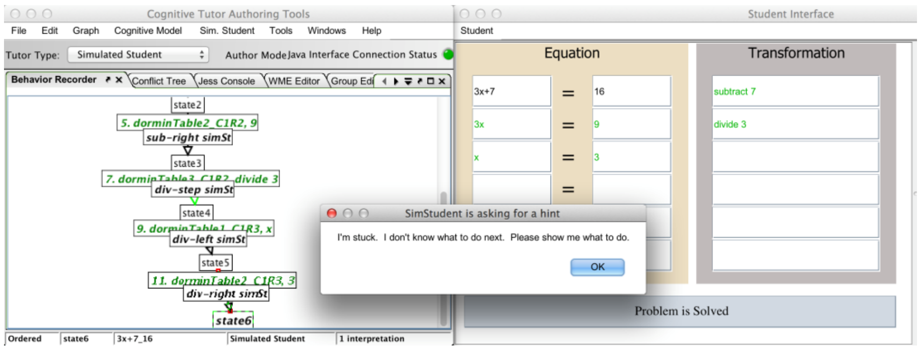
Tutoring SimStudent interactively

After constructing background knowledge, authoring is done in the tutor interface using CTAT and running SimStudent’s interactive learning module. SimStudent tries to solve the problem loaded into the interface by firing an applicable production rule and taking the step determined by the rule. After each step it asks for feedback on the correctness of that action (Figure 2). If the author provides positive feedback to SimStudent, then it will continue solving the problem. If the feedback is negative, SimStudent will try other applicable production rules. When it runs out of production rules that apply to the current step, it will ask the user to do that step and then use its learning mechanisms to learn a new production rule from that demonstration (Figure 3). After tutoring, SimStudent produces a behavior graph (shown on the left side in Figures 2 and 3) and a production rule file. The behavior graph can power an Example-Tracing tutor and the production rule file can run a Cognitive Tutor.


Cognitive Tutor Authoring Tools
File Edit Graph Cognitive Model Sim. Student Tools Windows Help
Tutor Type: Simulated Student
Author Mode Java Interface Connection Status
Behavior Recorder
Conflict Tree
Jess Console
WME Editor
Group Ed
3x+7_16
1. dorminTable3_C1R1, subtract 7
skill-sub simStIL
state1
C1R2, 9
state3
state2
Ordered
state8
3x+7_16
No graph has been opened yet
1 interpretation
Student Interface
Student
Equation
Transformation
3x+7
=
16
subtract 7
3x
9
divide 3
x
3
Applying the rule div-right
No
Yes
Problem is Solved

I entered "3" into dorminTable2_C1R3.
Do you think that would be a good move?

Figure 11. SimStudent asking for correctness feedback.



I'm stuck. I don't know what to do next. Please show me what to do.

Figure 12. SimStudent asking for a demonstration.

Future Work

The SimStudent architecture shows promise as a tool for simultaneously increasing authoring efficiency and model quality (MacLellan et al., 2014; Matsuda et al., n.d.), but more research still needs to be done. In terms of efficiency, few studies have directly compared the efficiency of different authoring approaches. We are exploring different usability and interaction models as a method for evaluating different approaches (e.g., the goals, operators, methods, and selections rules models). In terms of model quality, we are working to identify key performance metrics for general models. For example, in addition to evaluating accuracy and recall of a model for correct behavior, we are also looking at accuracy and recall of incorrect behavior. SimStudent can learn incorrect productions from correct instruction by making incorrect induction due to suboptimal background knowledge (Matsuda, Lee, Cohen & Koedinger, 2009). These plausible, but incorrect, inductions can be used to identify bug rules that might be missed by experts. As an example, when SimStudent is taught to divide both sides by 3 for the problem 3x = 6, it might incorrectly learn a rule for dividing both sides by any number on the left side of the equation (an error students commonly make). When SimStudent incorrectly solves a subsequent problem using this bug rule, it will receive negative correctness feedback and refine its rule to only apply to dividing by the coefficient. SimStudent could be modified to request a hint message for this misconception during authoring, such as text pointing out that the both sides of the equation are divided by the coefficient of x, not just any number. After this modification, it would be worth evaluating how authoring with SimStudent compares to Example-tracing or hand authoring in terms of number of bug rules identified.

In addition to evaluating efficiency and quality, we are also interested in exploring how to increase SimStudent's generality. To accomplish this, we have been exploring approaches for automatically learning the background feature predicates from tutoring (Li, Schreiber, Cohen & Koedinger, 2012). By reducing or eliminating the need to author this predicate knowledge, we will make it easier to apply SimStudent to new domains. Additionally, we have been exploring how this feature predicate learning can be used to apply SimStudent to learning models for open-ended tasks, such as educational games (Harpstead et al., 2013).

Using these new improvements, we are exploring the effectiveness of the SimStudent architecture for authoring content for a MOOC platform, such as Khan Academy. We are planning to recreate some of the MOOC instruction using SimStudent and to produce evidence that the cost-benefit of creating intelligent tutors for these platforms is worth it. There is a great potential for intelligent tutors to have a broader impact (through MOOCs and other avenues), if we can demonstrate that authoring tools can lower the cost to tutor authoring while jointly improving tutor quality and student learning. It is our hope that SimStudent, and other general tutor authoring platforms, can help achieve this goal.

Recommendations for GIFT

Based on our research with the SimStudent system, we have three recommendations for GIFT. First, as with many authoring frameworks, authoring expert models in GIFT is a challenging problem. As such, it may benefit from a tool like SimStudent to aid in this authoring process. SimStudent's automatically constructed expert models perform better than hand-authored models for multiple domains because they are not susceptible to expert blind spots. At the same time, in the process of generating these expert models, SimStudent makes errors that are often helpful in predicting human students' mistakes. These errors could form the basis of a misconceptions library, before any data are gathered from real students. Exploring how SimStudent's expert models and misconceptions could be used by GIFT may be a worthwhile direction for future work. This integration could take one of two forms: A SimStudent-like module could be constructed for GIFT that would allow authors to construct the domain knowledge by tutoring GIFT directly in the tutoring application or SimStudent could be configured to work with the tutoring application and then the production rule file generated by SimStudent could be converted into one of the domain knowledge formats acceptable to GIFT.

Second, we recommend that GIFT separate authoring of knowledge that is domain specific from the authoring of knowledge that is tutor specific. Domain general knowledge is already separated from domain specific knowledge in GIFT, but our research has found that domain specific knowledge is often reusable across tutoring applications. In the SimStudent system, we separated the construction of background domain knowledge (algebra features and operators), which tends to be reusable across tutors for the same domain (algebra), from the construction of an expert model for a specific tutor (how to solve particular algebra problems in the tutor interface). This was particularly useful because domain-specific background knowledge requires some Java programming abilities, whereas tutor-specific knowledge only requires the ability to demonstrate solutions in the tutor. This separation is useful because it allows domain experts, who may not know how to program, to construct the expert model for the tutor, if adequate domain knowledge already exists. Furthermore, our work with SimStudent has shown that domain-specific background knowledge tends to transfer across different tutors in the same domain and sometimes even across domains. For example, the feature predicates for extracting numbers and words from problem descriptions work in fraction addition tutors, algebra tutors, and chemistry tutors.

Finally, the modularity of GIFT makes it ideal for measuring the usability and efficiency of different combinations of authoring approaches and tools. We have used the GOMS model to evaluate the efficiency of different expert model authoring approaches (SimStudent and Example Tracing) in the context of CTAT. GIFT would benefit from similar analyses. Future GIFT research might explore how similar usability models can be employed for measuring the efficiency of different aspects of tutor authoring in a way that is comparable to other systems.

Acknowledgements

This work was supported in part by a Graduate Training Grant awarded to Carnegie Mellon University by the Department of Education (#R305B090023) and by the Pittsburgh Science of Learning Center, which is funded by the NSF (#SBE-0836012). This work was also supported in part by National Science Foundation Awards (#DRL-0910176 and #DRL-1252440) and the Institute of Education Sciences, U.S. Department of Education (#R305A090519). All opinions expressed in this article are those of the authors and do not necessarily reflect the position of the sponsoring agency.

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 & C. Tak-Wai, (pp. 61–70). Presented at the International Conference on Intelligent Tutoring Systems.

Aleven, V., McLaren, B. M., Sewall, J. & Koedinger, K. R. (2009). Example-Tracing Tutors: A New Paradigm for Intelligent Tutoring Systems. *International Journal of Artificial Intelligence in Education*, *19*(2), 105–154.

Chase, C., Chin, D. B., Oppezzo, M. & Schwartz, D. L. (2009). Teachable Agents and the Protégé Effect. *Journal of Science Education and Technology*, *18*, 334–352.

Harpstead, E., MacLellan, C., Koedinger, K. R., Aleven, V., Dow, S. P. & Myers, B. (2013). Investigating the Solution Space of an Open-Ended Educational Game Using Conceptual Feature Extraction. Presented at the International Conference on Educational Data Mining.

Koedinger, K. R. & Anderson, J. R. (1997). Intelligent Tutoring Goes To School in the Big City. *International Journal of Artificial Intelligence in Education*, *8*, 1–14.

Koedinger, K. R., Stamper, J., McLaughlin, E. & 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, (pp. 421–430). Presented at the Artificial Intelligence in Engineering, Memphis, TN.

Li, N., Schreiber, A. J., Cohen, W. W. & Koedinger, K. R. (2012). Efficient Complex Skill Acquisition Through Representation Learning. *Advances in Cognitive Systems*, *2*, 149–166.

Li, N., Stampfer, E., Cohen, W. W. & Koedinger, K. R. (2013). General and Efficient Cognitive Model Discovery Using a Simulated Student. In M. Knauff, M. Paulen, N. Sebanz & I. Wachsmuth. Presented at the Proceedings of the 35th Annual Meeting of the Cognitive Science Society, Austin: TX.

MacLellan, C. J., Koedinger, K. R. & Matsuda, N. (2014). Authoring Tutors with SimStudent: An Evaluation of Efficiency and Model Quality. In S. Trausen-Matu & K. Boyer. Presented at the International Conference on Intelligent Tutoring Systems.

Matsuda, N., Cohen, W. W. & Koedinger, K. R. (n.d.). Teaching the Teacher: Tutoring SimStudent Leads to More Effective Cognitive Tutor Authoring. *International Journal of Artificial Intelligence in Education*.

Matsuda, N., Keiser, V., Raizada, R., Tu, A., Stylianides, G., Cohen, W. W. & Koedinger, K. R. (2010). Learning by Teaching SimStudent: Technical Accomplishments and an Initial Use with Students. In V. Aleven, J. kay & J. Mostow, (pp. 317–326). Presented at the International Conference on Intelligent Tutoring Systems.

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, (pp. 1288–1293). Presented at the Annual Conference of the Cognitive Science Society, Austin, TX.

Nathan, M., Koedinger, K. R. & Alibali, M. (2001). Expert Blind Spot: When Content Knowledge Eclipses Pedagogical Content Knowledge (pp. 644–648). Presented at the Third International Conference on Cognitive Science.

Pane, J. F., Griffin, B. A., McCaffrey, D. F. & Karam, R. (2013). *Effectiveness of Cognitive Tutor Algebra I at Scale* (pp. 1–36). Santa Monica, CA: RAND Corporation.

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

Sottilare, R. A. & Holden, H. K. (2013). Motivations for a Generalized Intelligent Framework for Tutoring (GIFT) for Authoring, Instruction, and Analysis. In R. A. Sottilare & H. K. Holden, (pp. 1–150). Presented at the AIED 2013 Workshop on Recommendations for Authoring, Instructional Strategies and Analysis for Intelligent Tutoring Systems (ITS): Towards the Development of a Generalized Intelligent Framework for Tutoring (GIFT).

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. In G. McCalla, C. K. Looi, B. Bredeweg & J. Breuker, (pp. 678–685). Presented at the Artificial Intelligence in Engineering.

SIMILE: An Authoring and Reasoning System for GIFT

Howard Mall[1] and Benjamin Goldberg[2]
[1] Engineering Computer Simulations, Inc.; [2]U.S. Army Research Laboratory

Background

The Student Information Models for Intelligent Learning Environments (SIMILE) is composed of two components: an authoring system called the SIMILE Workbench and a run-time, reasoning component called the SIMILE Engine. Originally designed as an assessment middleware for simulations and serious games, it has broad applicability for doing learner assessments for any kind of learning experience that can produce data. We describe the work that was done to integrate the SIMILE Engine into the Generalized Intelligent Framework for Tutoring's (GIFT) Domain module. We also describe how the workbench can be used to author and export a Domain Knowledge File (DKF) as well as assessment rules for evaluating a learner's performance state with regard to defined concepts. We demonstrate how it was used to integrate the Tactical Combat Casualty Care Simulation (TC3sim) and also talk about the work that is currently being done to make it easier to bring in new systems with assessments into GIFT.

SIMILE started life as an Advanced Distance Learning Co-laboratory (ADL Colab) initiative. It was used to experiment with connecting serious games and simulations to learning management systems (LMSs). Part of this problem was that pre-existing simulations often did not report the same kind of assessments that one would track within a course of instruction inside of an LMS. Simulations typically stood (and stand) alone as a complete training system and did not interoperate within the ecosystem of learning systems that were geared towards courseware. The SIMILE Solution was to create a middleware that would provide the bridge between simulations and different kinds of LMSs.

Another research area of interest that SIMILE addressed was the authoring of assessments by learning professionals rather than programmers. Because of the close integration between a simulation and its assessment capability, software developers would often need to be called upon to author new assessments even if the simulated scenarios did not change. Because of the nature of software, the people creating the training would re-hire the original developers to make these changes. This made the simulations less flexible and often expensive to change.

As a side benefit, the SIMILE authoring capability could be used as a way to create assessments a priori to the development of a simulation or serious game scenario. An instructional professional could design the assessment and the domain data. The resulting models could then be delivered to the simulation developers as a specification for what data need to be exposed and how the simulation is intended to achieve its learning goals. The learning drives the simulation rather than the mechanics of the virtual environment driving the learning.

Standardizing the results of an assessment also helps to incorporate a number of simulations into a program of instruction. The simulations can act as replacements for each other without changing the assessment (as long as the simulations supply the same data.) Alternatively, multiple different simulations can be used for a full learning experience and with learner performance being assessed across all of them.

SIMILE has been used as a testbed for a number of different ideas about assessment middleware and disparate learning system interoperability. To this end, it was integrated into a number of simulations and virtual worlds. TC3sim, funded by what is now the U.S. Army Research Laboratory - Simulation Technology Training Center (ARL-STTC), was one of the first to incorporate SIMILE assessment. In the interest of full disclosure, this was ostensibly due to the fact that Engineering Computer Simulations, Inc., was the prime developer on both projects. TC3sim is a serious game for training U.S. Army combat medics. TC3sim uses SIMILE rules to assess a player's performance and generate an after action review (AAR) presented at the completion of a scenario.

SIMILE has been modified and updated to be used as both an authoring capability (i.e., Workbench) and a reasoning system to be incorporated into their GIFT architecture. The workbench has been modified to support GIFT assessment authoring and export concomitant GIFT Domain Knowledge Files (DKFs) that work in conjunction with the SIMILE rules. SIMILE's runtime process was incorporated into GIFT's Domain module for reasoning about a learner's performance within a serious game. TC3sim is used as a testbed for the integration of SIMILE and GIFT.

SIMILE Architecture

SIMILE is composed of two parts: the runtime Engine and the Workbench authoring tool (Figure 13.) The Engine ingests a configuration from the Workbench and then reasons about patterns of data that arrive from an external source (such as a simulation.) The Workbench is a graphical user interface (GUI) that allows people to author assessment rules and the data model that those rules reason about.

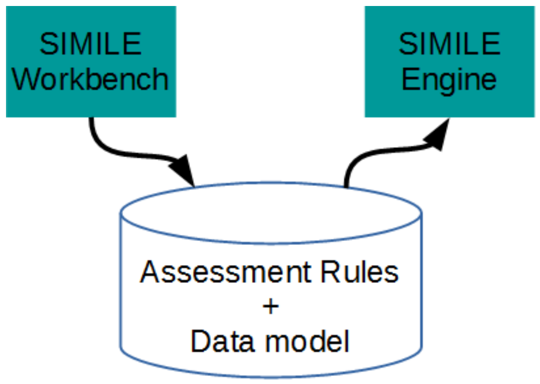


Figure 13. SIMILE Workbench authors a specification to be used by SIMILE Engine

Engine

The SIMILE Engine is a runtime component that generates assessment results. The engine ingests an Intelligent eXpert Syntax (IXS) file that specifies the assessment rules and the data model over which the rules reason. The IXS file can be hand coded or generated by an authoring system such as the SIMILE Workbench. The IXS file is loaded at the time the engine is run.

During a learning session, the SIMILE engine receives inputs from the system with which the learner is interacting (Figure 14.) In most examples, this is a simulation or serious game. The data model specification in the IXS file defines the schema of the events and objects that constitute the information that the SIMILE rules are interested in from the simulation. The plugin (i.e., Dynamic-Link Library (DLL)) defines how data are received from the external system and asserted within the SIMILE Engine for processing.

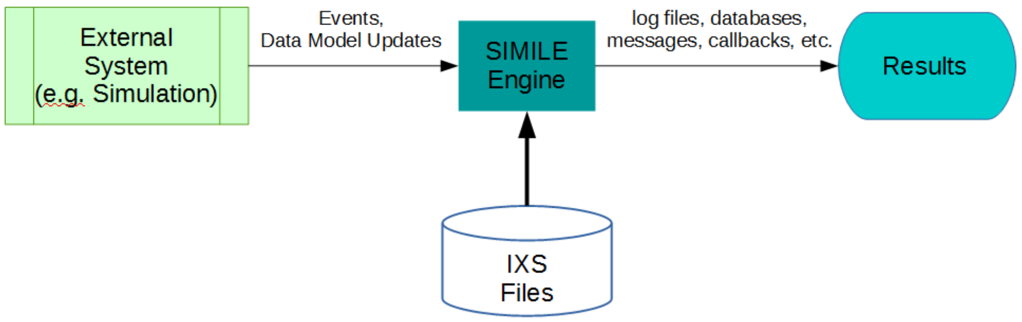

External System (e.g. Simulation)
Events, Data Model Updates
SIMILE Engine
log files, databases, messages, callbacks, etc.
Results
IXS Files

Figure 14. SIMILE Engine receives input from an external training system and publishes results of its reasoning.

The engine uses the rules to look for patterns within the instantiations of the data model and generates results based on those patterns. Basically, it is like medical diagnoses where the disease is inferred from the reported symptoms. In this case, the learner's state is inferred from the reported activity of their interaction with a training system, serving as a form of stealth assessment built around monitoring in-game behaviors (Shute, Ventura, Small & Goldberg, 2013).

The results generated by the engine can take a variety of forms such as writing information to a log file, calling functions, or sending messages using a specific protocol. The results are based on a component (i.e., DLL) that can be plugged into the engine. Rules can also assert information into the data model to be used by other rules. Chaining rules like this can be used to perform induction (Gonzalez, 1993): a diagnostic process.

The SIMILE Engine can be run as its own process and can speak to external systems through an inter-process communication protocol. It can also be incorporated into another runtime system to allow another process to control its behavior. This is how it is integrated into GIFT's architecture.

Workbench

The Workbench is the GUI that allows a user to create an IXS file to be used by the SIMILE Engine. The intent is to make it easier for a non-programmer to develop a data model of a learning environment and define how the learner will be assessed without knowledge of software programming. Additionally, the assessment model can be composed before a simulation is written where it can be used as a specification to the simulation programmer; or (as middleware) it allows assessments to be changed without altering the underlying simulation.

SIMILE's data model is composed of events and objects. Events are transient, one-time descriptions of data. Events are processed by SIMILE rules right when they occur. Objects are persistent entities within a learning system that have attributes (i.e., name-value pairs). Objects are created and persist for a certain amount of time within a learning session. The objects attributes can change over time and are updated during the training session. SIMILE rules look for patterns or thresholds of change within objects.

Rules are simple if-then statements. The “if” part of the rule defines the conditions for activating that rule. Matching a pattern within the defined data model is said to activate that rule. A rule, for example, can be activated by the occurrence of an event or the value of an object’s attribute going above or below a certain threshold (e.g., from TC3sim a virtual patient’s blood volume falling below 4000 mL.) The SIMILE Workbench provides an authoring environment for defining these conditions based on the defined data model. The user can select the object, the attribute, the operator, and the operand of the condition from drop downs within the authoring environment. Several conditions can be defined and related by “and” and “or” operators.

When a rule is activated, the “then” part of the rule is read. The “then” part of the rule can assert information into the data model or it can publish a result. The SIMILE Workbench allows a user to author the text of the result message or the assertion of data based on the data model through typing in a text box or using menus. A description of SIMILE Workbench customizations for GIFT is described below.

GIFT Integration

Motivations

GIFT is designed to support simulation-based training (SBT) practices across an array of environments and settings. The motivation behind producing this capability is to enable a system to monitor interaction in a virtual environment, void of a live instructor, and provide coaching and guidance when performance deviates from a desired standard. This is achieved through modeled representations of expert performance established in GIFT’s domain module. These constructed expert models serve as assessment criteria in gauging a trainee’s real-time performance by comparing trainee interaction against designated behaviors congruent with optimal performance. Identifying these deviations enables a system to intervene with a pedagogical strategy (i.e., provide guidance, adapt scenario, or do nothing) that is intended to influence subsequent interaction. The challenge is establishing tools and methods for identifying errors in performance that can be applied across multiple simulation environments.

In terms of SBT, these modeled representations are dependent on the type of data that can be passed from the simulation to GIFT. These data are typically in the form of defined message libraries and protocols, such as Distributed Interactive Simulation (DIS), High Level Architecture (HLA), Simple Object Access Protocol (SOAP), and other proprietary and standard communication formats. Based on recognized interactions and behaviors identified during a task analysis, a system developer is responsible for linking interaction characteristics as deemed by available data, regardless of the protocol in use, with concepts represented in GIFT’s DKF. Currently, this form of assessment authoring is done at the source code level and requires explicit writing of code to symbolize characteristics in data outputs with established concept representations.

This is not the ideal case with respect to the desired end users applying GIFT tools to build adaptive courseware. From this perspective, GIFT needs two functional components. First, GIFT requires tools and methods for creating expert models linked to any type of messaging protocol, and second, there needs to be a run-time capability that manages the interpretation of data against defined rule sets. For these reasons, SIMILE was identified as a potential solution to provide the intelligent tutoring system (ITS) community with a user-friendly authoring environment to establish rule sets GIFT can act on, along with a run-time engine that manages the data and makes inferences from patterns of that data.

Technical Considerations

When modifying SIMILE to support GIFT authoring and run-time processes, there were a number of functions that needed to be addressed. This is based on recognizing how GIFT operates pedagogically, and what information is used to make informed decisions on how best to instruct and guide a trainee. GIFT uses a relational hierarchy of defined concepts to represent the knowledge and skill associated with a training event. Each represented concept requires an associated assessment to be authored, which is used as evidence to determine how an individual is performing with relation to a defined expectation.

For each concept, metrics are defined to coincide with a GIFT performance state (e.g., at-, above-, or below-expectation). While the performance associated with a training event is domain dependent, it is assumed that transitions can occur between performance states as a training scenario unfolds. GIFT operates by communicating these transitions to the learner module where performance data are combined with other relevant learner information. This type of information includes affective states informed by sensor technologies, and individual differences linked to traits and aptitudes found to impact how one learns (e.g., prior knowledge, motivation, preferences).

In terms of informing the learner module of transitions in performance states, GIFT's domain module is set up to perform assessments based on authored rules and carries out domain specific implementations of pedagogical tactics linked with strategy requests received from the pedagogical module. A DKF specifies how GIFT's domain module will accomplish its purpose.

The DKF is where users define the relational hierarchy of concepts linked to a domain. It is also in the DKF where an author will designate how data are treated when received from the gateway module. This is where SIMILE fits in. While the DKF serves multiple functions, it operates as an .xml file within the domain module and currently is created through an .xml editor provided with a GIFT install. What SIMILE provides to alleviate this burden is an authoring environment (i.e., the Workbench). The SIMILE Workbench, described above, is a graphical interface that allows an author to build assessment rule sets around an established message library associated with a given simulation. For the SIMILE Workbench to support GIFT-based assessment, modifications had to be completed so the interface could link authored rule sets to established concepts represented in the DKF hierarchy.

Integrated Architecture

The SIMILE Workbench has been customized with GIFT-specific models and concepts. The GIFT Engine has been integrated into the Domain module of GIFT so a learner's state conditions can be authored as SIMILE rules. Additionally, DKF import and export plugins were written and incorporated into the SIMILE Workbench.

Learner State Conditions

For GIFT, the SIMILE Workbench has been customized and extended to allow the author to associate rules with specific concepts. Concepts define something about which a learner needs to demonstrate understanding. For example, a concept for a combat medic could be “Applies Tourniquet.” This describes a mental state of the learner where they know and understand when to apply a tourniquet to a wounded casualty. Concepts have four states associated with them: “At Expectation,” “Below Expectation,” “Above Expectation,” and “Unknown.” A combat medic learner that is not applying a tourniquet to a casualty that has an arm amputation in a timely manner would be “Below Expectation” for the “Applies Tourniquet” concept. SIMILE Rules define the conditions for concepts that indicate when the learner is in any of the four defined states.

The definition of rules require very explicit knowledge of the learning domain and may not be an appropriate technique for every possible tutoring situation. There are many other techniques in support of authoring intelligent tutors (Murray, 1999), such as constraint-based representations. They are best applied in domains requiring expert knowledge (Crowley, 2003). They also may require a significant amount of time and expertise from the tutoring author to create. However, the Workbench has been adapted to present rule-authoring through a GIFT-specific conceptual model that helps to alleviate the burden of having to approach the problem as an expert system. It is hoped, that further work can be done in the future to address other techniques of representation with the SIMILE Workbench in the future.

Domain Module

In the case of GIFT, the Domain module controls the SIMILE Engine as a DLL. Because GIFT is written in Java, a Java Native Interface (JNI) wrapper is used to call SIMILE Engine functions. The Domain module passes messages of interest to the SIMILE Engine. The SIMILE Engine uses the message to update its data model. This may activate rules and cause results to be passed out to the Domain module through its call to the SIMILE Engine.

For GIFT, SIMILE Workbench has a set of import/export plugins that handle DKFs. When an IXS file is built for a particular domain a concomitant DKF may also be exported. This DKF works in conjunction with the IXS file in the SIMILE Engine integrated in GIFT’s Domain module.

The SIMILE Workbench can also import a DKF into the SIMILE Workbench. SIMILE’s data model is populated from the defined domain model in the DKF. GIFT Tactics, Concepts, and Transitions are brought into the workbench for editing and addition. Previously authored DKF structures can be used to add SIMILE rules that define the conditions of a learner’s state with regard to their mastery of concepts. The DKF and the IXS work together to evaluate a learner’s performance and let GIFT figure out when to inject tutoring tactics.

GIFT+SIMILE Workflow

The typical author and learner workflow for how SIMILE works with GIFT follows this basic path (Figure 15):

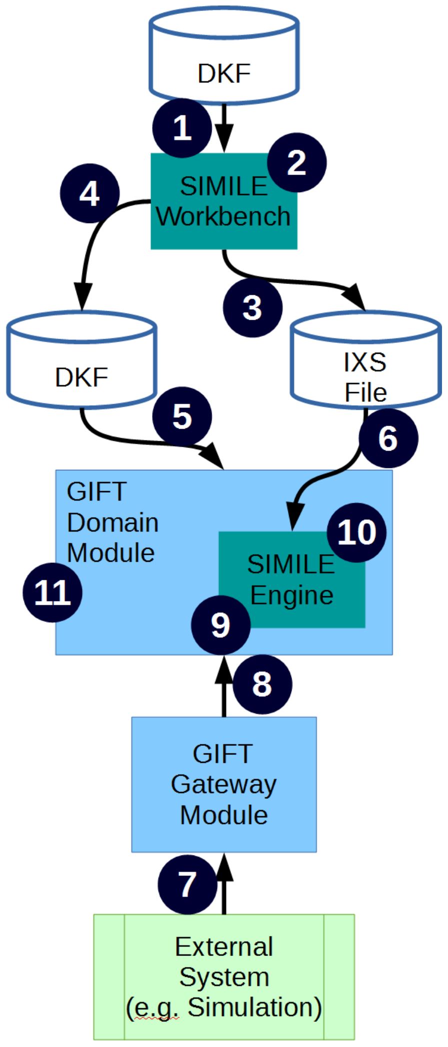


Figure 15. GIFT+SIMILE workflow

Optionally, if there is a pre-existing DKF, the assessment author can import that DKF into the SIMILE Workbench. The SIMILE Workbench will be populated with named concepts, learner state transitions, and defined tactics found in that DKF.

The assessment author writes learner state conditions as rules using SIMILE Workbench. The author also authors state transitions and their associated tactics for individual concepts as well.

- The author publishes their project as an IXS file.
- The author exports a DKF for use by GIFT's Domain module.

- At run-time, the GIFT's Domain module reads in the DKF and initializes SIMILE.
- SIMILE reads in the IXS file.
- Messages from the training simulation are routed through GIFT's Gateway Module.
- GIFT's Domain module determines which messages are of interest to SIMILE and passes those messages to the SIMILE process.
- SIMILE updates its data model based on those messages and rules are run against the new set of data looking for patterns.
- If a rule or set of rules is activated, the SIMILE Engine reports the learner's state to GIFT's Domain module.
- The GIFT architecture makes a determination of whether a tutoring tactic needs to be implemented and sends out messages to that effect in its standard way.

Basically, the SIMILE Engine behaves as a very sophisticated assessment function within GIFT. The SIMILE Workbench extends GIFT's authoring capability to include the logic as rules of those assessment functions.

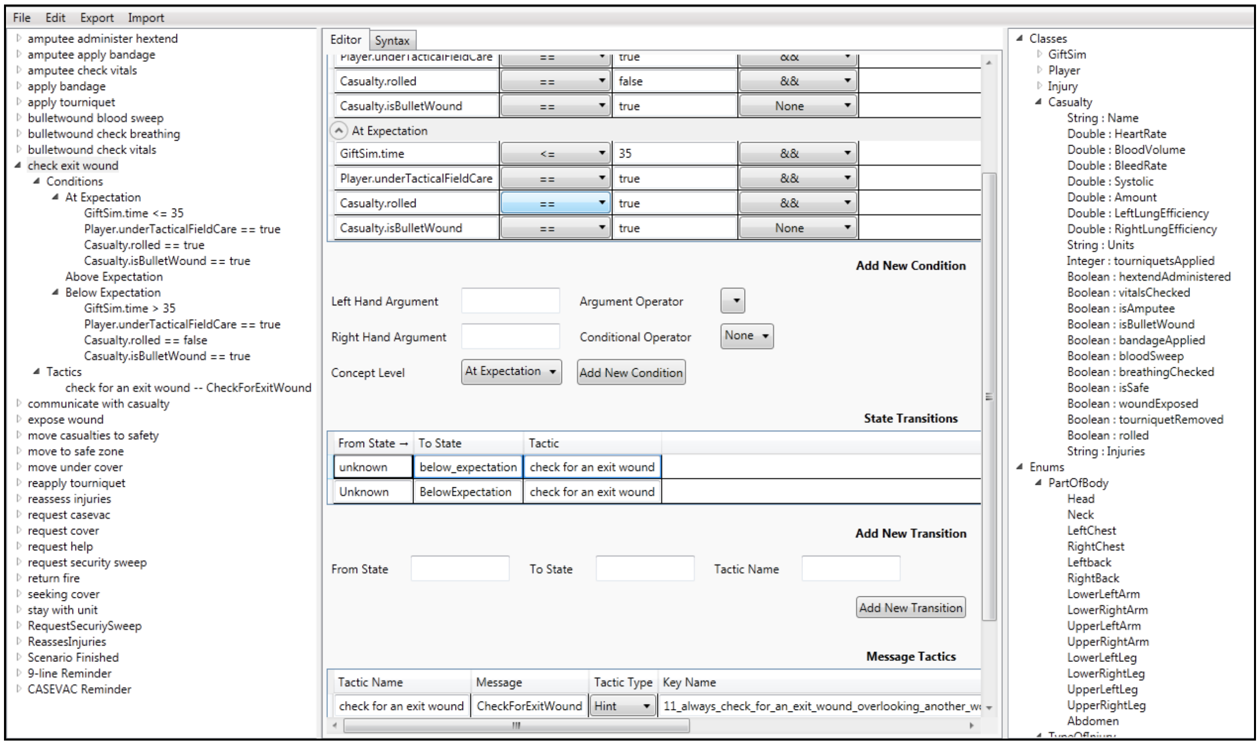
TC3sim Inside of GIFT

The initial integration point for GIFT and SIMILE was done by pairing the tutoring framework with the serious game TC3Sim. This integration offered an excellent use case as it provided a game-based learning environment that was already utilizing SIMILE to monitor interaction for the purpose of informing AAR materials. To support real-time assessment that GIFT can act on, SIMILE was modified to link established rules with performance states (i.e., at-, above-, and below-expectation) representative of the concepts defined in GIFT's DKF.

The Workbench now supports defining concepts, defining rules to establish a performance state for each concept, defining observable transitions between performance states, and assigning tactics to enact when a transition results in an intervention request from the pedagogical module. To test the utility of SIMILE acting as a run-time assessment engine, an experiment was developed looking at the effect real-time feedback had on training outcomes in TC3Sim (see Goldberg, 2013, for a complete review of the associated study). This integration of SIMILE enables GIFT to monitor a participant's interaction in the game TC3Sim, the ability to link data generated by the system against models of expert performance, the ability determine when rules associated with a model designate transitions in performance states, and the ability to implement instructional tactics when a strategy request is received by the pedagogical module.

For authoring purposes, SIMILE's workbench was modified to allow rules to be linked with concepts represented in GIFT's DKF (Figure 16). During this process, it was recognized that the workbench can provide an environment to author all components of a DKF while helping a user maintain orientation of how everything generated in SIMILE is linked to the prescribed tutoring effect chain that informs tutoring practices (Sottilare, Goldberg, Ragusa & Hoffman, 2013).

With this capability, a user can now author a notional DKF in the SIMILE workbench environment and export its representation in an .xml file that GIFT operates on. This includes defining the concepts that will be monitored during run-time in a simulation, linking rules to those concepts that designate performance states based on data communicated from the simulation, and establishing instructional tactics to be carried out when a performance state results in GIFT's pedagogical module producing a strategy request. Prior to these modifications, all of these elements were authored in the DKF in distinctly different sections of the file, making it difficult to maintain orientation of how each field was explicitly linked with the others. The new SIMILE Workbench simplifies this process by providing all of these fields into a single view, making it easier to see the associations between concepts, assessments, and instructional tactics.



```
amputee administer hextend
amputee apply bandage
amputee check vitals
apply bandage
apply tourniquet
bulletwound blood sweep
bulletwound check breathing
bulletwound check vitals
check exit wound
  Conditions
    At Expectation
      GiftSim.time <= 35
      Player.underTacticalFieldCare == true
      Casualty.rolled == true
      Casualty.isBulletWound == true
    Above Expectation
    Below Expectation
      GiftSim.time > 35
      Player.underTacticalFieldCare == true
      Casualty.rolled == false
      Casualty.isBulletWound == true
  Tactics
    check for an exit wound -- CheckForExitWound
communicate with casualty
expose wound
move casualties to safety
move to safe zone
move under cover
reapply tourniquet
reassess injuries
request casevac
request cover
request help
request security sweep
return fire
seeking cover
stay with unit
RequestSecuriySweep
ReassesInjuries
Scenario Finished
9-line Reminder
CASEVAC Reminder
```

Player.underTacticalFieldCare	==	true	&&
Casualty.rolled	==	false	&&
Casualty.isBulletWound	==	true	None
At Expectation			
GiftSim.time	<=	35	&&
Player.underTacticalFieldCare	==	true	&&
Casualty.rolled	==	true	&&
Casualty.isBulletWound	==	true	None

Left Hand Argument
Argument Operator
Right Hand Argument
Conditional Operator: None
Concept Level: At Expectation

From State →	To State	Tactic
unknown	below_expectation	check for an exit wound
Unknown	BelowExpectation	check for an exit wound

From State
To State
Tactic Name

Tactic Name	Message	Tactic Type	Key Name
check for an exit wound	CheckForExitWound	Hint	11_always_check_for_an_exit_wound_overlooking_another_w

```
Classes
  GiftSim
  Player
  Injury
  Casualty
    String : Name
    Double : HeartRate
    Double : BloodVolume
    Double : BleedRate
    Double : Systolic
    Double : Amount
    Double : LeftLungEfficiency
    Double : RightLungEfficiency
    String : Units
    Integer : tourniquetsApplied
    Boolean : hextendAdministered
    Boolean : vitalsChecked
    Boolean : isAmputee
    Boolean : isBulletWound
    Boolean : bandageApplied
    Boolean : bloodSweep
    Boolean : breathingChecked
    Boolean : isSafe
    Boolean : woundExposed
    Boolean : tourniquetRemoved
    Boolean : rolled
    String : Injuries
Enums
  PartOfBody
    Head
    Neck
    LeftChest
    RightChest
    Leftback
    RightBack
    LowerLeftArm
    LowerRightArm
    UpperLeftArm
    UpperRightArm
    LowerLeftLeg
    LowerRightLeg
    UpperLeftLeg
    UpperRightLeg
    Abdomen
```

Figure 16. SIMILE Workbench modified for GIFT integration

Improvements and Additions

Currently, the SIMILE Workbench is being extended with a plugin to allow its data model to be imported from GIFT's state protocol definitions defined as Java classes. These classes are used by GIFT to communicate internally about the state of a simulation that is communicating through the Gateway Module. SIMILE can use these state definitions within its data model in order to cut down on the amount of work required to use a new simulation or other training application within GIFT.

Additionally, SIMILE has defined a generic SIMILE message class as part of GIFT's state protocol definition. Objects and events specified within SIMILE's data model can be communicated as payload within those messages using a standard textual format (i.e., JavaScript Object Notation: JSON.) This means that a user who is integrating a new simulation into GIFT does not have to define new message types as part of GIFT's internal protocol. If the message payload format of the simulation is also JSON, then a generic Gateway Module plugin can be utilized without requiring any programming. Even if the training system uses a different protocol, the workload of programming a new Gateway Module plugin can be reduced.

Additional work is now being incorporated into the SIMILE Workbench that supports the nesting and ordering of concepts. Nesting allows a concept to be made dependent on the results of sub-concepts. The sub-concepts' states become entities that can be reasoned about while authoring the state conditions of a super-concept. Those conditions can use "AND" and "OR", but also the temporal operators, "BEFORE" and "AFTER," to reason about the ordering of concepts.

References

Crowley, R. & Medvedeva, O. (2003) *A General Architecture for Intelligent Tutoring of Diagnostic Classification Problem Solving*. American Medical Informatics Association (AMIA) Annual Symposium Proceedings.

Goldberg, B. (2013). Explicit Feedback Within Game-Based Training: Examining the Influence of Source Modality Effects on Interaction. Ph.D., University of Central Florida.

Gonzalez, A. J. & Dankel, D. D. (1993). The Engineering of Knowledge-Based Systems.

Murray, T. (1999). *Authoring Intelligent Tutoring Systems: An Analysis of the State of the Art*. International Journal of Artificial Intelligence in Education.

Shute, V., Ventura, M., Small, M. & Goldberg, B. (2013). Modeling Student Competencies in Video Games Using Stealth Assessment. In R. Sottilare, A. Graesser, X. Hu & H. Holden (Eds.), *Design Recommendations for Intelligent Tutoring Systems, Volume 1: Learner Modeling* (pp. 141-152).

Sottilare, R. A., Goldberg, B., Ragusa, C. & Hoffman, M. (2013). *Characterizing an Adaptive Tutoring Learning Effect Chain for Individual and Team Tutoring.* Paper presented at The Interservice/Industry Training, Simulation & Education Conference (I/ITSEC).

Theme II: Intelligent Tutoring System Interoperability

Interoperable Performance Assessment for Individuals and Teams Using Experience API

Tiffany Poeppelman[1], Mike Hruska[2], Rodney Long[3], and Charles Amburn[3]
[1] Aptima, Inc., Problem Solutions
[2]U.S. Army Research Laboratory, Human Research and Engineering Directorate
[3]Simulation and Training Technology Center

Introduction

Organizations are making substantial investments in new training technology and methods of delivering training, such as mobile learning, virtual collaborative workspaces, and distributed simulations (ASTD, 2012). With industry focused on personalized and tailored learning, more adaptive training approaches for customizing training experiences are being developed. Adaptive training (also referred to as accelerated learning) is a generic term for a family of approaches that use individual difference variables to personalize the individual training experience by selecting an appropriate training event or changing the content within the event (Poeppelman, Blacksmith & Yang, 2013).

Additionally, adaptive intelligent tutoring systems (ITSs) or computer-based tutoring systems (CBTSs) provide a method of guiding self-development by managing real-time instructional decisions. These systems select ideal instructional strategies to meet the specific learning needs of individuals or teams (Sottilare, 2012). Where human tutors are either unavailable or impractical, ITSs provide opportunities for reflection, and change the content, direction, pace, and challenge level of instruction to optimize learning.

Currently, the United States Army is focusing on creating a more competitive environment for leaders and Soldiers through its new campaign for lifelong learning. To expand learning beyond the schoolhouse, the Army is evolving their goals to include new training methods and forms of performance tracking throughout ones career. The Army Learning Concept (ALC) 2015 (Department of the Army, 2011) is pushing the boundaries of research and practice and has introduced new architecture designs and concepts.

One recent effort at the U.S. Army Research Laboratory (ARL), Human Research and Engineering Directorate (HRED), Simulation and Training Technology Center (STTC) on Interoperable Performance Assessment (IPA) is presenting a path forward to greater efficiency of training performance across training systems. IPA, a method of uniformly defining and describing data about a user's experience in a consistent way, has created a capability to enable different systems to leverage one another's data to macro and micro adapt learner experiences. Specifically, the effort at ARL has developed an approach for enabling the Generalized Intelligent Framework for Tutoring (GIFT) to adapt learner's future events or learning path based on past performance. This paper highlights the current IPA efforts and provides an overview of the future research efforts surrounding IPA and GIFT.

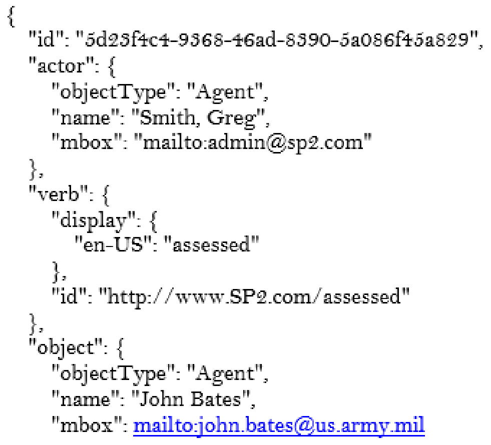
Experience API

Currently, the Department of Defense (DoD) has a number of efforts underway related to tracking and assessing performance in order to enable these adaptive learner-centric environments. With known limitations of the Sharable Content Object Reference Model for environments like mobile, games, and simulations, the Advanced Distributed Learning (ADL) Initiative is currently working to extend the future support of interoperability of learning systems through a Training and Learning Architecture (Advanced Distributed Learning, 2013). Specifically, ADL is focusing on new approaches to standards and frameworks, which include the Experience Application Programming Interface (Experience API or xAPI).

The xAPI specification reached 1.0 as of April 26, 2013. This method defines an API to track data about learning experiences and defines a protocol for data structure and transmission by software components to communicate with one other (Advanced Distributed Learning, 2013). The xAPI defines a method to capture data about the interaction between a learner and a learning experience.

The xAPI allows tracking outside of a learner management system (LMS) to capture data about digital and non-digital learning and user experiences across the formal, informal, and experiential spectrum. This includes experiences from newer training environments, such as serious games, mobile applications, simulations, virtual world, and augmented reality.

The xAPI is based upon an open format specification for activity stream protocols, which are used to syndicate activities (Activity Streams, 2013). The xAPI specification defines a format using JavaScript Object Notation (JSON) to create statements that allow the capture of learning experiences. JSON is a text-based open standard designed for human-readable data interchange. The format of a statement is <Actor, Verb, Object> or "I did this." **Error! Reference source not found.**1 is an example of an xAPI statement in JSON.



```
{
  "id": "5d23f4c4-9368-46ad-8390-5a086f45a829",
  "actor": {
    "objectType": "Agent",
    "name": "Smith, Greg",
    "mbox": "mailto:admin@sp2.com"
  },
  "verb": {
    "display": {
      "en-US": "assessed"
    },
    "id": "http://www.SP2.com/assessed"
  },
  "object": {
    "objectType": "Agent",
    "name": "John Bates",
    "mbox": mailto:john.bates@us.army.mil
```

Figure 1. xAPI example statement

The xAPI specification is providing a means to uniformly describe data in meaningful way that has both flexibility and structure. As the ITS community continues to enable new methods of providing personalized feedback and instruction, xAPI will allow ITSs to share user data and enhance the capabilities of a system by leveraging data about performance outcomes and results.

Interoperable Performance Assessment

The purpose of the current ARL research is to capture and leverage contextual performance measures from current systems and encode them into xAPI statements to tailor learning to the individual learner's experience and level of proficiency on a concept or capability. Through the work performed, the concept of IPA was defined as "a method of uniformly defining and describing experience and context to assess learning and performance over time; to adapt training across a variety of environments, systems, and modalities, whereby performance is observed, assessed, evaluated, or asserted by systems or observers." Figure 2 shows a visualization of the IPA concept to represent a learner and how is tracked across multiple environments to adapt their pathway.

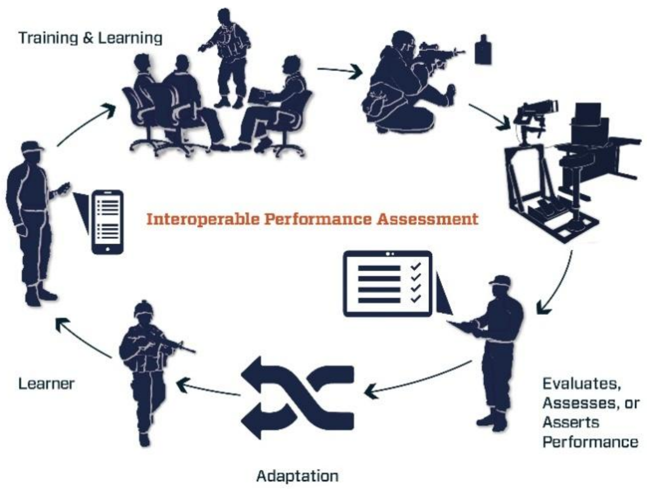

Training & Learning
Interoperable Performance Assessment
Evaluates, Assesses, or Asserts Performance
Adaptation
Learner

Figure 2. Interoperable performance assessment diagram

This definition not only combines the methods in which interoperable tracking occurs, but also (a) where the trainee, event, or training content is adapted and (b) how the data are collected. While the definition remains broad, the intent is to move toward a common understanding of what is meant by interoperable tracking of performance data and the goals of assessing performance over time.

The previous effort set out with the following objectives: (1) define best practices for defining and encoding performance measurement using xAPI statements; (2) develop a technical architecture for interoperable activity across current Army architectures; and, (3) build a proof-of-concept called the Soldier Performance Planner (SP^2). Below is an overview of each of the objectives and outcomes.

xAPI Encoding Best Practices Guide

The first research objective included the research of best practices for encoding individual performance data into xAPI statements from system-based (simulator) and observer-based (instructor) measures. Current efforts that capture individual and collective performance include an Extensible Markup Language (XML) activity structure known as the Human Performance Measurement Language (HPML). This schema was designed to capture and assess performance across distributed simulations by a language that identifies critical fields and stores them within an XML activity structure. The major goal of HPML is to focus on bridging the gap between the software implementation of raw data to measurements and computing measurements into overall assessments. Overall, HPML is designed to allow the expression of important concepts from the training world so that others, such as training professionals, instructors, operators, and researchers, can use, aggregate, and understand the data easily (Stacy, Ayers, Freeman & Haimson, 2006). The current effort uses HPML constructs as a basis for describing current performance data that is being collected by various Army simulators, as well as to understand what type of system-based and observer-based data is being tracked across environments.

The result of reviewing HPML included a list of pre-defined constructs along with the associated definition, examples, and usage requirements. Examples of these constructs include training objective, knowledge, skill, and roles. While some of the constructs were deemed useful as additional data sources or context, some are not required based on the current HPML schema.

The first objective resulted in the SP² Encoding Best Practices Guide that outlines data encoding considerations using the xAPI format to track data typically stored or described using HPML. The data formats used included xAPI version 1.0.1 and HPML version 3.1. The main purpose of the mapping was to provide guidance for encoding xAPI statements based upon consideration of the constructs of HPML.

Technical Architecture

The second research objective included developing requirements for ITSs, such as GIFT, to view this performance data. An xAPI statement is a human-readable set of data objects consisting of attribute-value pairs that use the JSON format. Data are sent in the form of time-stamped statements, which take on a natural language pattern of “subject-verb-object.” Storage of these xAPI statements is done via a learning record store (LRS) and allows access of that data by other systems. An LRS may be standalone or part of a larger system.

As described above, the key components of the technical architecture include (1) HPML, which captures raw data, measures, and computes assessments within systems; (2) xAPI, which can capture interoperable learning experience data at any level of granularity; (3) LRS, which enables centralized storage and retrieval of all learning data captured by the xAPI; and, (4) GIFT polling component, which allows the ITS to receive xAPI statements and make macro recommendations within GIFT. The GIFT polling component queries relevant statements for a learner from an LRS. Figure 3 shows the current SP² architecture diagram, which contains all described components.

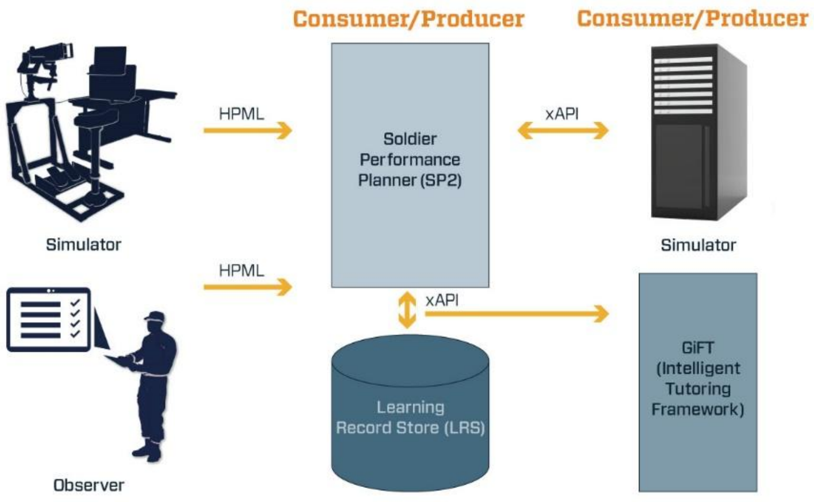
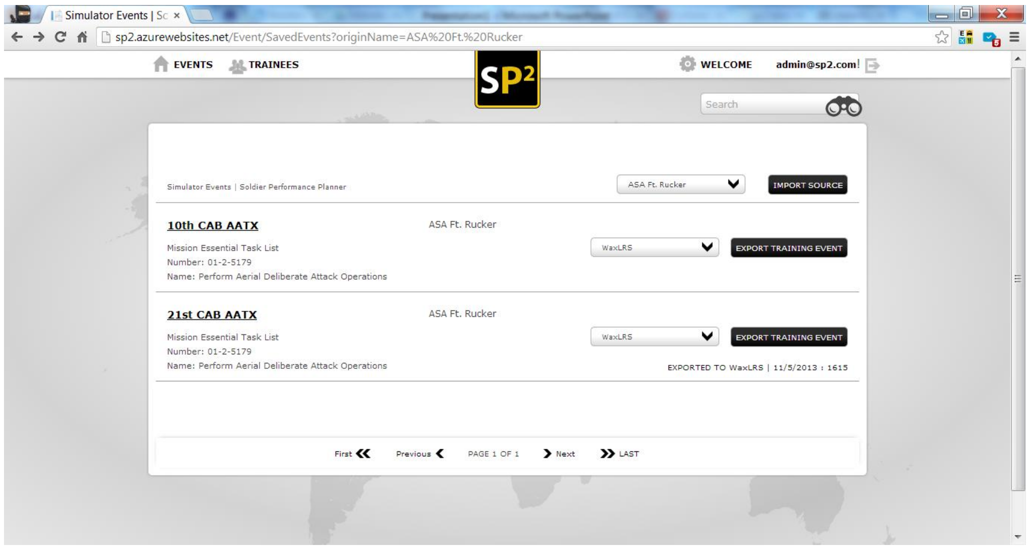

Consumer/Producer
Consumer/Producer
HPML
xAPI
Soldier Performance Planner (SP2)
Simulator
Simulator
HPML
xAPI
Learning Record Store (LRS)
GiFT (Intelligent Tutoring Framework)
Observer

Figure 3. SP² architecture diagram

Soldier Performance Planner (SP²)

Lastly, the third project objective included the development of a prototype called the SP², which leverages the key elements found in the technical architecture above. Those include HPML, xAPI, LRS, and GIFT. The SP² HTML interface allows systems that use an LRS or HPML or xAPI data formats to connect and share data, add new individual data, and track individual and group performance data that will ultimately support adaptive and tailored learning across ITSs and other systems. While the data tracking functionality has traditionally been associated with the LMS, an LRS is able to collected statements that contain new types of training data that are being stored. The current purpose of xAPI and LRS is to support lifelong learning through persistence by tracking services and aggregating learner data from multiple systems. Figure 4 shows a screenshot of the SP² interface.


Simulator Events | Sc
sp2.azurewebsites.net/Event/SavedEvents?originName=ASA%20Ft.%20Rucker
EVENTS
TRAINEES
SP²
WELCOME
admin@sp2.com!
Search

Simulator Events | Soldier Performance Planner

ASA Ft. Rucker | IMPORT SOURCE

10th CAB AATX ASA Ft. Rucker

Mission Essential Task List
Number: 01-2-5179
Name: Perform Aerial Deliberate Attack Operations

WaxLRS | EXPORT TRAINING EVENT

21st CAB AATX ASA Ft. Rucker

Mission Essential Task List
Number: 01-2-5179
Name: Perform Aerial Deliberate Attack Operations

WaxLRS | EXPORT TRAINING EVENT

EXPORTED TO WaxLRS | 11/5/2013 : 1615

Figure 4. SP² interface

Generalized Intelligent Framework for Tutoring (GIFT)

Developed by ARL, GIFT is a modular tutoring system framework to support the instantiation of adaptive tutoring capabilities. It aims to research and prototype a computer-based tutoring framework to evaluate adaptive tutoring concepts, models, authoring capabilities, and instructional strategies (Sottilare, Goldberg, Brawner, Holden, 2012). The effort aims to do this across various populations, training tasks, and conditions, thus enabling summative and formative evaluations.

GIFT services provide authoring of CBTS components, tools, and methods; management of instructional processes using best pedagogical practices based on the behaviors of expert human tutors; and an assessment methodology to evaluate the effectiveness of CBTS and CBTS components, tools, and methods. GIFT provides services for learners, instructional system designers, expert behavior modelers, training system developers, trainers, and researchers. The infrastructure provides generic tutoring or remediation strategies to integrating systems based on learner performance.

The current research and use of xAPI within GIFT is meant to start by informing macro-adaptation methods. The GIFT polling component allows GIFT to communicate with the LRS, determine relevant adaptations, and macro-adapt recommendations for courses. The polling component adapts based upon xAPI statements that state whether the learner is at or above a level of proficiency for a competency or concepts indicate that a user is performing below a level proficiency for a competency or concept. Based on the current rule sets, courses are either no longer recommended or presented based on their past performance.

Additionally, current ARL efforts are developing a writing component that outputs data from GIFT into the xAPI format. This data mapping will enable outside LMSs and other systems to leverage xAPI statements describing user experiences that come from within GIFT. IPA efforts are looking at macro approaches to inform course recommendations and create more granular learner profiles that have intersystem data value.

Community Development

The purpose of xAPI and IPA is to describe, store, and provide access to learning experiences including traditional records, such as scores or completion status, as well as assertions of proficiency or deficiency for concepts, competencies, knowledge or skills. Intersystem communication between LMSs, ITSs, and other systems allows domain-independent ITSs and training technology-based solutions to aggregate rich and meaningful data across a continuum of systems to increase efficiency and effective use of learning and training assets. By understanding the current state and granular historical data of a learner, these systems will ultimately be able to adapt learner pathways at the macro and micro level. Figure 5 shows the data and functional ecosystem of the ITS, LRS, and xAPI.

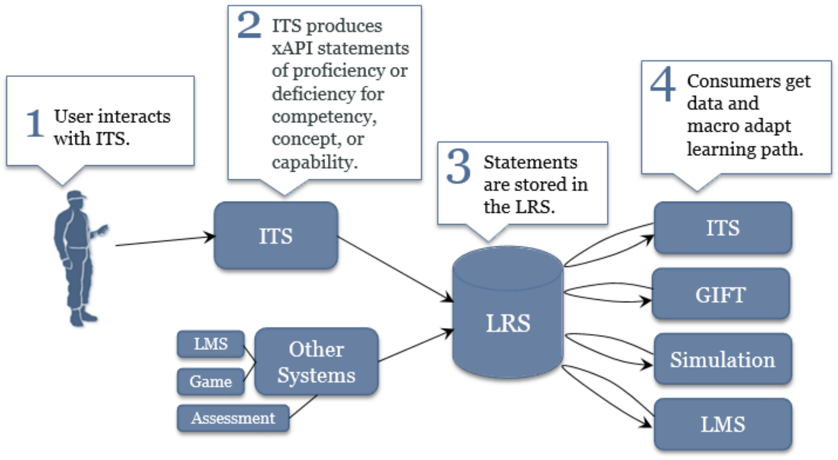

1 User interacts with ITS.
2 ITS produces xAPI statements of proficiency or deficiency for competency, concept, or capability.
3 Statements are stored in the LRS.
4 Consumers get data and macro adapt learning path.
ITS
LRS
LMS
Game
Assessment
Other Systems
ITS
GIFT
Simulation
LMS

Figure 5. ITS, LRS and xAPI ecosystem

Based on the current work to date surrounding SP² and the developed architecture, potential impacts to the design and development of GIFT might include the classification of additional data sets that inform GIFTs approach to intelligent tutoring. Additionally, outside simulators and systems including other ITSs might consider connecting to an LRS as an additional data store to support initialization of learner profiles, course selection within GIFT, and potential extraction of data from GIFT to modification of long-term learner profiles.

Future Research

While current ITSs are focused on reducing high training development costs, improving standards, and adapting to support the tailored needs of the learner, there is still much research and development left to be done to attain this goal. As new domains continue to be explored, IPA best practices documentation and methods will iterate based on domain-specific verbs, actors, actor taxonomies, activities, and more domain-specific context descriptions. Future IPA efforts should continue to explore tools and methods to convert these strategies into specific instructional tactics for implementation. For instance, statements from other systems about competencies, knowledge, and skills could be utilized to select teams, identify group scenarios, and determine individual mission injects that would leverage the strengths of the teams to mitigate team weaknesses.

The key to interoperability for performance assessment will also be to encode activity data across a number of streams (system-based measures, observer-based measures, computer-based training measures, physiological, etc.). To date, SP² has ingested system- and observer-based measures for individuals and teams. Encoding of additional streams will allow more data types to be used for testing and validation. Current efforts are underway to determine what data sets are applicable and what systems will be used to demonstrate interoperability. The goal is to continue to introduce more data sources that will lead to a comprehensive approach, a validation technique to confirm the data encoding process, and additional data sources to inform the community of encoding best practices across relevant domains.

Current community-driven efforts are being leveraged to support this encoding effort. A number of current projects from ARL and other Services are working toward a common end. An emergence of adoption of usage and tools is happening around the xAPI specification, which has the ability to accelerate the ALC 2015 goals. Key support tools and practices are being developed to share methods of tracking performance data to enable tailored and adaptive learning across current ITSs, such as GIFT.

Based on the community and data encoding efforts, enhancements should continue to be made to GIFT so that additional macro and micro approaches are evolved into the GIFT architecture. Additionally, other adaptive engines that leverage xAPI should also be leveraged for experimentation purposes. Furthermore, much of the future is focused on systems that provide adaptive and tailored learning not only for individuals but also teams. Leveraging encoded xAPI individual and team performance data from multiple systems still poses a unique set of challenges that need to be explored further. Focus remains on having a highly agile system that is capable of tracking performance, experiences, and group data, as well as individual levels of data, from the national and Joint levels.

Finally, research needs to continue to focus on the intersection between the variety of efforts underway at the DoD as they relate to tracking and assessing performance, which is an important step to enable these adaptive learner-centric environments. The current work recently completed at ARL presents an opportunity to conduct further analyses and technical exchanges among groups in order to make the community intersection possible around adaptive learning systems. These integration efforts will continue to grow with specifications such as Experience API in order to enable 21st century technologies to adapt based on learner profiles and performance data.

References

Activity Streams Project. (2013). Activity Streams Home Page. Available at http://activitystrea.ms.

Advanced Distributed Learning. (2013). *ADL Learning Record Store*. Retrieved 2013, from www.adlnet.gov: https://lrs.adlnet.gov/xapi/

Advanced Distributed Learning. (2013). *Sharable Content Object Reference Model*. Retrieved May 2013, from http://www.adlnet.gov/overview

Advanced Distributed Learning. (2013). *Training and Learning Architecture (TLA)*. Retrieved from Advanced Distributed Learning: http://www.adlnet.gov/capabilities/tla

Advanced Distributed Learning. (2013). *Training and Learning Architecture (TLA): Experience API (xAPI)*. Retrieved from http://www.adlnet.gov/tla/experience-api

Advanced Distributed Learning. (2013). *xAPI Specification*. Retrieved 2013, from Github: https://github.com/adlnet/xAPI-Spec

ASTD. (2012). *2012 State of the Industry Report.* Alexandria: ASTD Press.

Department of the Army. (2011). *The U.S. Army Learning Concepts for 2015.* Washington: Department of Defense.

Poeppelman, T. R., Blacksmith, N. & Yang, Y. (2013, Oct.). Application of modern technology and social media in the workplace. The Industrial-Organizational Psychologist, 51(2), 119-126.

Sottilare, R. A. (2012). Considerations in the Development of an Ontology for a Generalized Intelligent

Framework for Tutoring. In Proceedings of the International Defense and Homeland Security Simulation Workshop 2012, Vienna, Austria.

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).

Stacy, W., Ayers, J., Freeman, J. & Haimson, C. (2006). Representing Human Performance with Human Performance Measurement Language. Washington, DC. Aptima, Inc.

Game-based Architecture for Mentor-Enhanced Training Environments (GAMETE)

Jennifer Engimann[1], Thomas Santarelli[1], Wayne Zachary, Ph.D.[1], Xiangen Hu, Ph.D.[2], Zhiqiang Cai[2], Howard Mall[3], and Benjamin S. Goldberg, Ph.D.[4]
[1] CHI Systems, Incorporated
[2] University of Memphis
[3] ECS, Incorporated
[4] U.S. Army Research Laboratory

Introduction

The Generalized Intelligent Framework for Tutoring (GIFT) provides a flexible set of service-oriented tools to enable the construction of intelligent tutoring systems (ITSs). In its current state, GIFT provides authoring tools to enable training designers to parametrically map a learning domain and establish a common domain knowledge structure or ontology. This structure is represented across the learning space, tutoring strategy, student model, and game/simulation environment. The Game-based Architecture for Mentor-Enhanced Training Environments (GAMETE) seeks to increase the capability of training developers to incorporate advanced tutoring capabilities into simulation-based gaming environments within the GIFT framework (GIFT, 2012). GAMETE focuses on enabling training designers to integrate intelligent-tutoring models with game environments and parametrically control the interactions between the tutor, the game, and the learner.

In the simplest sense, GAMETE is middleware – an intermediary between the tutor and the game. Figure 1 shows the conceptual organization of GAMETE as a flexible middleware solution for integrating the experiential characteristics of game environments with the guided instructional and learning structure provided by ITSs. As shown in Figure 1, there are two conceptually distinct "users" of GAMETE. When a tutor/game pair is being executed, the user is the *learner*, to whom the distinctions between the ITS, GAMETE, and game components are transparent. When a tutor/game pair is being built or edited, the user is the *content author*, ideally an instructional designer, who uses tools in GAMETE as well as tools in the ITS engine and/or game engine to insert content in the training application.

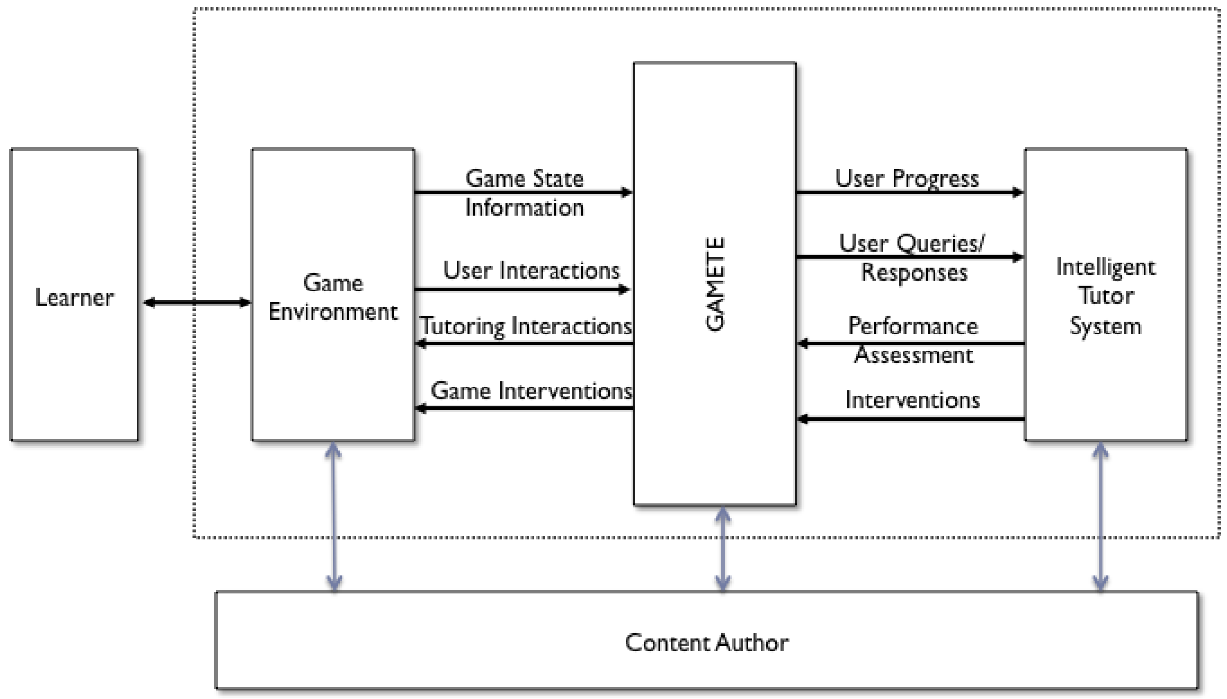


Figure 1. Basic role of GAMETE in game-based tutoring

From the user-learner's perspective, the main interaction is with a game or simulated environment. GAMETE captures both game state information and user interaction data as the learner interacts with the game. These are then supplied to the ITS as indicators of the user's behavior, from which the ITS can infer the user's progress. The ITS may then supply performance assessments to GAMETE so that GAMETE can apply them to the game as it proceeds forward. The ITS may also make interventions of a tutoring nature based on this same information. Both of these (assessments and interactions) can involve changes (e.g., interruptions, overlays, etc.) to the game display that involve additional learning-based interactions by the user. These learning-based interactions are also managed through GAMETE and passed to the ITS, from which the ITS can further update its understanding of the user's knowledge state and learning state against current learning objectives.

Much of the infrastructure that GAMETE uses to do this comes from the GIFT environment. GAMETE, however, adds visibility and control to this process at a level that is accessible to instructional designers rather than, say, game- or ITS-programmers. In addition to the general GIFT infrastructure, GAMETE also makes specific use of the Student Information Models for Integrated Learning Environment (SIMILE) tools within GIFT that support the development and assessment of a student model (ECS, 2012). In this initial version of GAMETE, the development has focused on one specific exemplar or use case, on both the game side – the Tactical Combat Casualty Care Simulation (TC3sim) game (also known as Virtual Medic: vMedic) – and the ITS side – the AutoTutor Lite system (Graesser, D'Mello, Hu, Cai, Olney & Morgan, 2012; Graesser, Conley & Olney 2012). Figure 2 shows this more functional organization of GAMETE in the game-tutor process as seen from this initial tutor/game pairing.

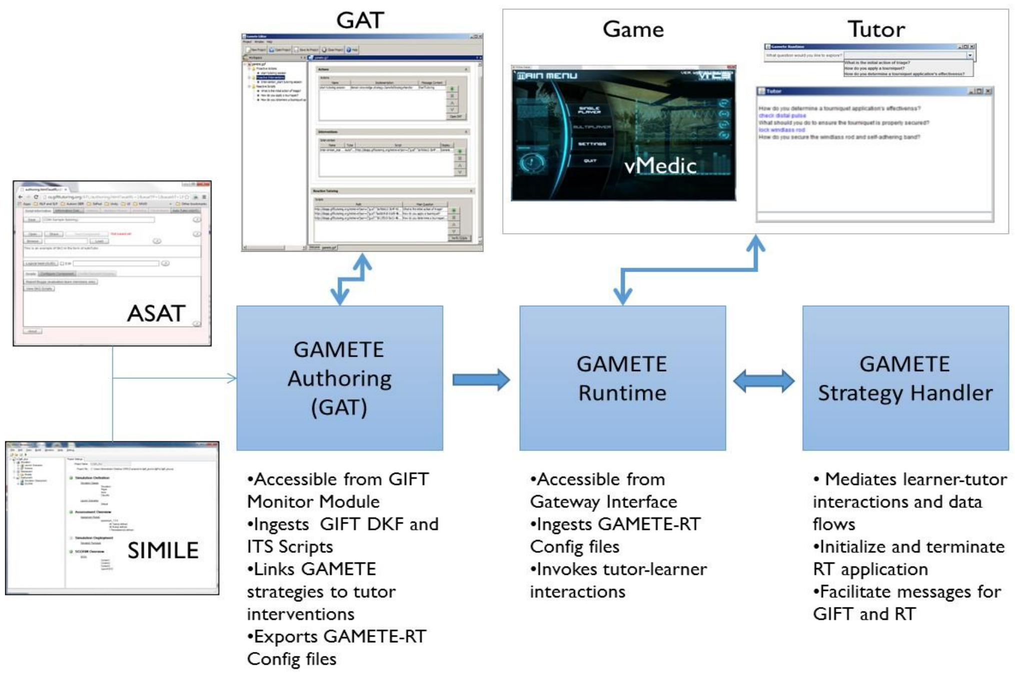

GAT
Game
Tutor
vMedic
ASAT
SIMILE
GAMETE Authoring (GAT)
GAMETE Runtime
GAMETE Strategy Handler
•Accessible from GIFT Monitor Module
•Ingests GIFT DKF and ITS Scripts
•Links GAMETE strategies to tutor interventions
•Exports GAMETE-RT Config files
•Accessible from Gateway Interface
•Ingests GAMETE-RT Config files
•Invokes tutor-learner interactions
• Mediates learner-tutor interactions and data flows
•Initialize and terminate RT application
•Facilitate messages for GIFT and RT

Figure 2. GAMETE functionality

GAMETE is organized as groupings of functionality that differentiate among aspects of function and use. The authoring cluster (the GAMETE Authoring Tools (GAT)) addresses the design and specification phase, in which an instructional designer interacts with GAMETE to develop an interoperation schema to specify system-initiated and learner-initiated tutoring options available during game execution. The GAMETE Domain Strategy Handler component serves as an interoperation point-of-contact between a specific game (through abstracted learning assessments), GIFT, and the GAMETE Runtime Environment. The GAMETE Run-Time Component provides graphical user interfaces to select learner-initiated tutoring options and serves as the communication interface for both tutor-initiated and learner-initiated interactions. The more detailed architecture of GAMETE is presented below, followed by discussions of each major functional area highlighted in Figure 2.

GAMETE Architecture

Figure 3 shows the high-level software architecture for the current GAMETE implementation, framed in the context of a learning application involving the AutoTutor Lite (ATL) ITS and the TC3Sim game. The authoring tools (green/light boxes) in Figure 3 provide the instructional designer with the means to grant the eventual learner two types of tutoring interactions, learner-initiated and tutor-initiated:

- *Learner-initiated Tutoring* allows the learner to explore the instructional space by asking specifically for help and choosing from a number of optional lessons.
- *Tutor-initiated Tutoring* allows the tutor, once it has identified a situation where it needs to interact with learner, to interrupt the learner-game interaction and interjects the desired transaction.

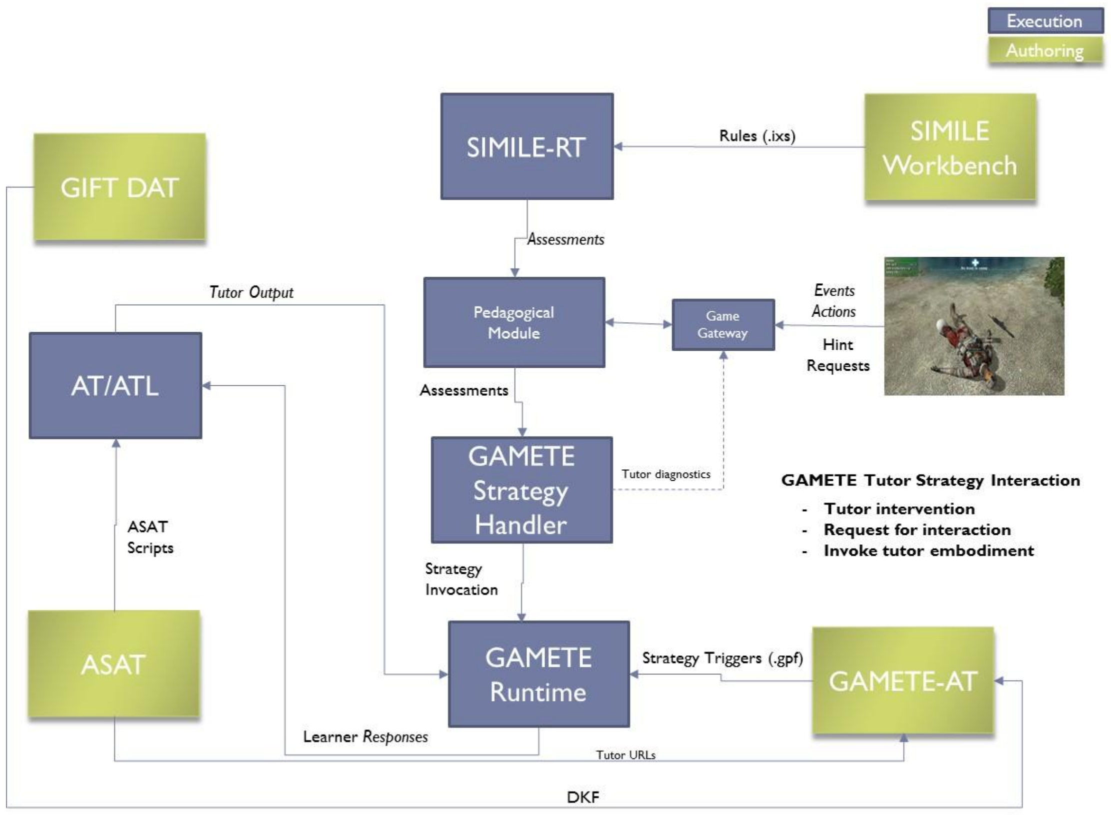


Figure 3. Core GAMETE component architecture

In the ATL/TC3Sim example, tutoring sessions for both the learner-initiated and tutor-initiated cases are specified (for execution) by the server locations of correlated AutoTutor scripts. The task assessments that are available for mapping to proactive tutoring sessions are defined in the correlated domain knowledge file (DKF) managed within GIFT. Once the mappings among the tutoring options, AutoTutor script locations and assessment tasks are defined, they are represented in a generalized .xml format, which is exported by the authoring component for use by the GAMETE run-time component.

The GAMETE Strategy Handler is an instructional strategy implementation within the GIFT domain module meant to handle changes to learner state that are linked to a GAMETE strategy. The GAMETE Strategy Handler mitigates calls to the GAMETE Runtime to produce tutor-initiated interventions as dictated by the DKF.

The GAMETE Runtime, in parallel, provides a vehicle to control and examine the execution of the domain-specific tutor being mediated by the GAMETE Strategy Handler (in the tutor-initiated tutoring case) and direct learner requests for help (in the learner-initiated tutoring case). This provides a means for learners to select tutoring options and interact with the tutor.

The key aspect used to maintain generalizability within GAMETE's authoring and runtime components is to conform them, as closely as possible, to the internal GIFT ontology and structure. Thus, at a more detailed level the three main architectural components of GAMETE have the following characteristics:

- *GAMETE Authoring:* An independent application, accessible from the GIFT Monitor module. This authoring application ingests pre-composed DKF, links specifically tagged instructional strategies to pre-composed tutoring scripts, and exports GAMETE project file appropriate for ingestion by the runtime application.

- *GAMETE Runtime:* An independent application, interfaced through both the GAMETE Strategy Handler of the GIFT Domain module and a set of AutoHotkey Scripts. The runtime application ingests pre-composed GAMETE project files (GPF), operates in concert with a training application to dictate simulation commands and tutor infusion as specified in the GPF.

- *GAMETE Strategy Handler:* This component responds to changes in learner state (as determined by a performance measure and as defined within the DKF) that requires an instructional intervention that is to be handled by the GAMETE Strategy Handler. This class passes information about the performance measure to the Runtime Application through interface methods. This information dictates the actions of the Runtime Application to possibly pause the simulation and start a tutoring session.

GAMETE Authoring - Design and Specification Tools

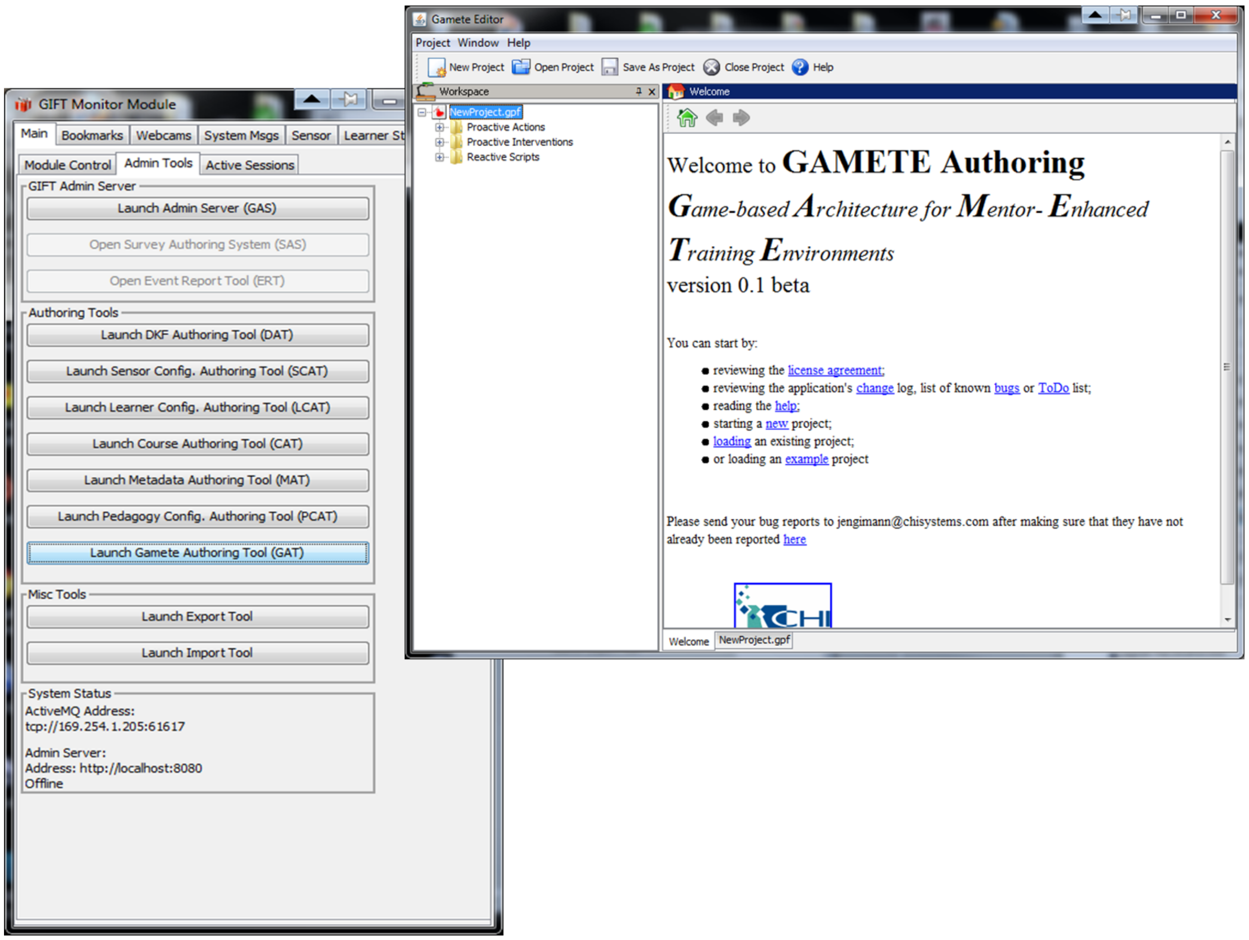
The GAT is a self-contained Java application that can be accessed through the GIFT Monitor module. GAT provides a graphic user interface that enables an author to:

- Configure GAMETE to link a tutor with a game environment (through actions such as importing a DKF, linking to a specific tutor service, and configuring high-level parameters within the tutor), and

- Map instructional interventions to tutoring functionality, to enable the GAMETE run-time to interface with the linked tutor in order to mediate the instructional interventions from the tutor to the learner. These mapping are contained within a GAMETE Project File (.gpf), which is checked for validity and exported at the conclusion of the authoring session.

Figures 4 and 5 show examples of the GAT user interface. Authoring within GAT allows instructional designers a number of linking possibilities. They can create learner-initiated linkages, tutor-initiated linkages or both. Tutor-initiated linkages will ultimately be displayed to the user through a forced pausing of the simulation followed by external presentation of the lesson script. Learner-initiated linkages are listed in a drop-down menu of choices that is displayed when the user requests a tutoring interaction at runtime by pressing Alt-h.

Authoring of tutor-initiated linkages begins by importing the DKF for the lesson currently being authored. Importing the DKF provides a list of strategies that will be used throughout the lesson. Only strategies predefined to be handled by the GAMETE Strategy Handler will be shown as these strategies will be routed through the GAMETE Strategy Handler interfaces to the GAMETE Runtime for tutoring display. After selecting a GAMETE Strategy, a second list will be generated allowing the author to associate the selected strategy with a tutor and tutor script URL. Additional parameters for tutor usage will also be editable from this window. Upon the conclusion of editing, the GAT will export the mapping into a GPF to be specified in the Domain Course File and run within the GAMETE Runtime Application.



System Status
ActiveMQ Address:
tcp://169.254.1.205:61617

Admin Server:
Address: http://localhost:8080
Offline

Welcome to **GAMETE Authoring**

Game-based Architecture for Mentor- Enhanced Training Environments

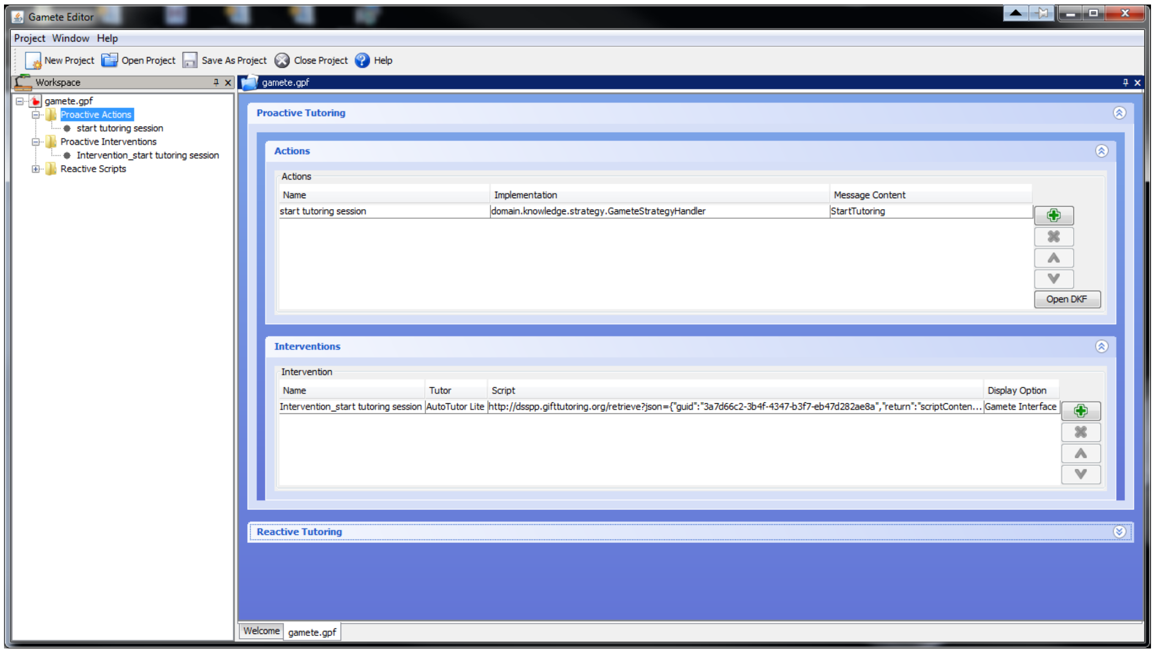
version 0.1 beta

You can start by:

- reviewing the license agreement;
- reviewing the application's change log, list of known bugs or ToDo list;
- reading the help;
- starting a new project;
- loading an existing project;
- or loading an example project

Please send your bug reports to jengimann@chisystems.com after making sure that they have not already been reported here

Figure 4. GAMETE authoring welcome screen



Actions

Name	Implementation	Message Content
start tutoring session	domain.knowledge.strategy.GameteStrategyHandler	StartTutoring

Intervention

Name	Tutor	Script	Display Option
Intervention_start tutoring session	AutoTutor Lite	http://dsspp.gifttutoring.org/retrieve?json={"guid":"3a7d66c2-3b4f-4347-b3f7-eb47d282ae8a","return":"scriptConten...	Gamete Interface

Figure 5. GAT tutor-initiated tutoring authoring tool

Authoring learner-initiated linkages involves creating a list of accessible tutor scripts from which the learner can choose at run-time, via a drop-down list. This is done within the Reactive Tutoring section of the authoring tool by pressing the green "+" to add a new script link. The author types the path to the AT script in the corresponding path column. After all script paths are added, pressing the "verify scripts" icon will check that the script path is valid and will populate the corresponding main question columns.

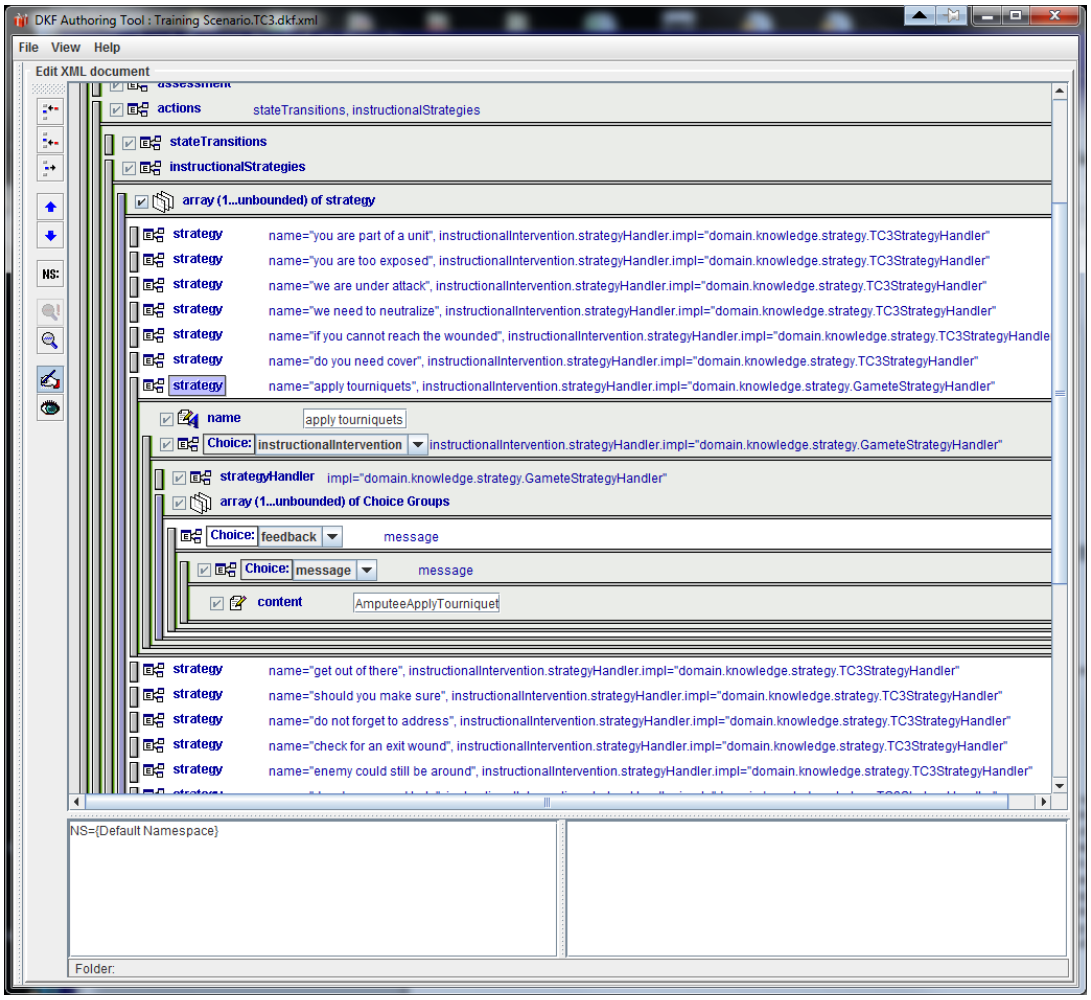
An important focus for GAMETE authoring is the ability to enable an author to control the flow and sequencing of tutor-learner interactions. The GAMETE authoring toolset was designed to enable an author to introduce a tutoring interaction (e.g., an ATL script) within a game-based scenario and tie those interactions to learner-specific data. We envision this control being tied to data at multiple levels of abstraction, ranging from learner state coming from SIMILE (e.g., "at/below" expectation) to more detailed parameters such as the number/type of errors within a scenario or prior feedback delivered within the scenario.

We have also identified more detailed GAMETE authoring design concepts where an author is linking two concepts and affixing a decision point (e.g., within the "*EvaluateInjury*" concept node within the tutor). In this simple example, an author may want to remind the trainee of the proper procedure to evaluate an injury prior to applying a tourniquet. To enable GAMETE to invoke the tutor accordingly, the author attaches a GAMETE Strategy (i.e., tutor intervention) to the "below expectation" learner state for EvaluateInjury within the DKF. This approach enables the GAMETE run-time within the GIFT Domain module to react to this and other calls for instructional intervention by GAMETE Strategies in the DKF. The *GAMETEStrategyHandler* within the GIFT domain model passes the content of the strategy message on through interface methods to be handled by the GAMETE Runtime Application as dictated by the content of the GPF.

GAMETE Middleware

The GAMETE baseline interoperability core consists of a module termed the GAMETE Strategies module. The GAMETE Strategies module is based upon the GIFT infrastructure and serves as the hub within the GAMETE formative architecture. It is the basis of the GAMETE baseline interoperability core, and provides a flexible capability to link GAMETE Strategies to particular changes in learner state based on rules authored within the GAMETE Authoring Tool.

For GAMETE to proactively influence a learner's progression through a lesson (by pausing/starting a simulation, beginning/ending a tutoring session, etc.), it must be commanded to take control of lesson adaptation. This is accomplished by assigning a GAMETE strategy to a particular change in learner state, and then linking a set of learner states to GAMETE strategies within the DKF. To keep the GAMETE abstraction away from the low-level states and processes of the simulation environment, the middleware was designed to use SIMILE to provide performance measurements against specified learning objectives throughout the simulation. For example, a SIMILE script would be authored to determine that a simulated medic within a TC3Sim scenario is near a bleeding patient and is not making progress towards applying a required tourniquet. SIMILE would communicate this information to GIFT's domain module designating a shift in performance. This information is routed to the learner module where a message is sent to the GIFT pedagogical model, informing interested parties that the apply tourniquet learning objective is being reported as below expectation. The content author decides that this would be an appropriate time to introduce a tutoring scenario, reminding the trainee of the proper procedure to follow towards applying the tourniquet. To instruct GAMETE to take over the lesson progression at this point, the content author attaches a GAMETE strategy to the "below expectation" learner state for the apply tourniquet concept within the DKF. Within the GIFT domain module, the GAMETEstrategyhandler class was added to react to calls for instructional intervention by GAMETE strategies in the DKF. This class passes the content of the strategy message on to the GAMETE runtime application and is handled by the content of the GAMETE project file. Figure 6 shows an example of what this interaction looks like to the GAMETE author/user.

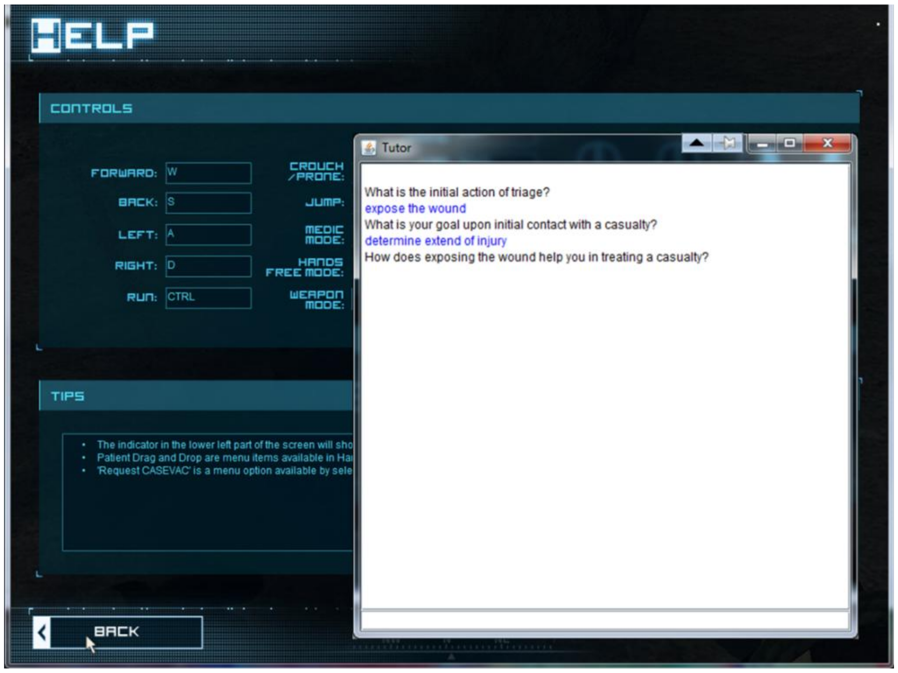


```
DKF Authoring Tool : Training Scenario.TC3.dkf.xml
File  View  Help
Edit XML document
actions                 stateTransitions, instructionalStrategies
  stateTransitions
  instructionalStrategies
    array (1...unbounded) of strategy
      strategy    name="you are part of a unit", instructionalIntervention.strategyHandler.impl="domain.knowledge.strategy.TC3StrategyHandler"
      strategy    name="you are too exposed", instructionalIntervention.strategyHandler.impl="domain.knowledge.strategy.TC3StrategyHandler"
      strategy    name="we are under attack", instructionalIntervention.strategyHandler.impl="domain.knowledge.strategy.TC3StrategyHandler"
      strategy    name="we need to neutralize", instructionalIntervention.strategyHandler.impl="domain.knowledge.strategy.TC3StrategyHandler"
      strategy    name="if you cannot reach the wounded", instructionalIntervention.strategyHandler.impl="domain.knowledge.strategy.TC3StrategyHandle
      strategy    name="do you need cover", instructionalIntervention.strategyHandler.impl="domain.knowledge.strategy.TC3StrategyHandler"
      strategy    name="apply tourniquets", instructionalIntervention.strategyHandler.impl="domain.knowledge.strategy.GameteStrategyHandler"
        name              apply tourniquets
        Choice: instructionalIntervention  instructionalIntervention.strategyHandler.impl="domain.knowledge.strategy.GameteStrategyHandler"
          strategyHandler   impl="domain.knowledge.strategy.GameteStrategyHandler"
          array (1...unbounded) of Choice Groups
            Choice: feedback        message
              Choice: message         message
                content         AmputeeApplyTourniquet
      strategy    name="get out of there", instructionalIntervention.strategyHandler.impl="domain.knowledge.strategy.TC3StrategyHandler"
      strategy    name="should you make sure", instructionalIntervention.strategyHandler.impl="domain.knowledge.strategy.TC3StrategyHandler"
      strategy    name="do not forget to address", instructionalIntervention.strategyHandler.impl="domain.knowledge.strategy.TC3StrategyHandler"
      strategy    name="check for an exit wound", instructionalIntervention.strategyHandler.impl="domain.knowledge.strategy.TC3StrategyHandler"
      strategy    name="enemy could still be around", instructionalIntervention.strategyHandler.impl="domain.knowledge.strategy.TC3StrategyHandler"
NS={Default Namespace}
Folder:
```

Figure 6. Assigning a GAMETE strategy using the DKF authoring tool

GAMETE Run-time Tools and Operation

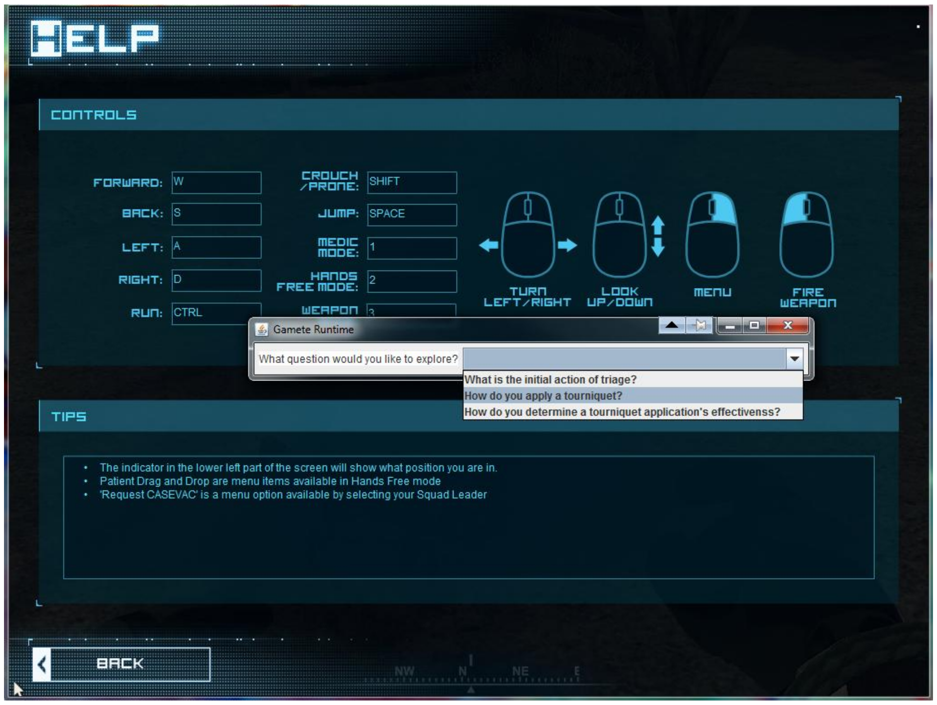
The GAMETE Runtime Application (GRA) is an independent Java application that is started during instantiation of the domain module, and interfaces with the GAMETE Strategy Handler via a set of AutoHotkey scripts. The GRA is intended to be a background process that runs simultaneously with a simulation such as TC3Sim. It receives strategy messages from the GAMETEStrategies module and compares them against the mapping specified in the GPF. From this mapping, the GRA determines which tutoring scenario to start, along with running parameters for that scenario, using AutoHotkey scripts to pause the active simulation and present the tutoring interface to the learner (Figure 7).



What is the initial action of triage?
expose the wound
What is your goal upon initial contact with a casualty?
determine extend of injury
How does exposing the wound help you in treating a casualty?

Figure 7. Tutoring interface over paused simulation

The GRA can also be accessed during the simulation by a learner wishing to request a tutoring session. By pressing *Alt-h* during a simulation, an AutoHotkey script calls for the presentation of a drop-down list of the available tutoring questions (Figure 8). When the learner chooses a question, the tutoring interface is presented. After tutoring is complete, the scenario resumes with normal game play.



What question would you like to explore?
What is the initial action of triage?
How do you apply a tourniquet?
How do you determine a tourniquet application's effectivenss?

- The indicator in the lower left part of the screen will show what position you are in.
- Patient Drag and Drop are menu items available in Hands Free mode
- 'Request CASEVAC' is a menu option available by selecting your Squad Leader

Figure 8. Tutor-initiated dropdown over paused simulation

As the GRA is intended to run as a background process, it will be invisible to a learner during the normal progression of a GIFT course until the dropdown is requested or the tutoring interface is presented.

Summary and Future Plans

This paper has described the initial steps in implementation of the GAMETE concept. The main innovations of GAMETE have been the design and implementation of the following:

- An extensible tutor/game middleware authoring/control capability for GIFT:
    - The GAMETE Strategy Handler to allow GAMETE to take control of a scenario and interject tutoring into it automatically;
    - A simple tutor/learner interface that can be extended as functionality is added;
    - A graphical user interface to allow a user to ask for help in the form of a tutoring session while in game (i.e., for learner-requested tutoring);
    - A comprehensive GAMETE Authoring Tool to allow an instructional designer to tie together proactive and reactive tutoring options to be mitigated by the GAMETE Runtime Environment;
    - A practical example of using GAMETE to link the TC3Sim simulation/game with the AutoTutor Lite tutor.

This research has also identified several areas for expansion of the initial GAMETE infrastructure as well as for future research. The ongoing GAMETE research will continue to expand GAMETE functionality in several areas:

- Expanded integration of student modeling capability via expanded support for SIMILE authoring tools,
- Expanded run-time integration to broader set of games/simulation, particularly to the Virtual Battle Space (VBS) family (e.g., Evertz et al., 2009) that is widely use in military training games, and
- The use of other measurement constructs to trigger tutor interventions such as learner affect and motivation states.

The research reported here has focused at the level of specific game-tutor pairs. One insight gained through working at that level was that greater interoperability could be gained from abstracting the interoperability interactions and developing interoperability authoring and communication *standards*, and building support for those standards into GAMETE. This principle could be applied both narrowly, (e.g., between specific tutors and games) as well as more abstractly (e.g., at the level of tutoring *engines* and game *engines*). An example of this principle might be evolving the .xml based communications mechanism between ATL and GAMETE to a more abstracted mechanism based on the Sharable Knowledge Object (SKO) model (Nye, 2013). Such an abstraction could enable the following:

- Real-time updating of tutoring services through semantic message-passing;
- Create a looser-coupling of tutoring services both within a tutoring system and between a tutoring system and external systems interoperated through GAMETE/GIFT;
- An ontology of message types that describe the majority of typical tutoring events and learning experiences that are encountered when interacting with a human learner; and
- Interoperability with the Foundation for Intelligent Physical Agents (FIPA) and Advanced Distributed Learning (ADL, 2013) xAPI standard for messaging learning experiences to learning record stores.

Such more abstracted standards could be supported by GAMETE-based plug-ins for both tutoring and game engines, thus offering broad and lower-development-effort GIFT-based interoperability to be built into games and/or tutors from initial development.

References

Advanced Distributed Learning (2013). Experience API http://www.adlnet.gov/tla/experience-api/

Bell, B., Snooks, Q., Engimann, J., (2012). *Game-based Architecture for Mentor-Enhanced Training Environments (GAMETE).* Phase I Final Report, CHI Systems Report, Fort Washington, PA.

ECS. (2012a). SIMILE. Retrieved May 01, 2014, from http://www.ecsorl.com/products/simile

Evertsz, R., Pedrotti, M., Busetta, P., Aca, H., Ritter, F. (2009). Populating VBS2 with Realistic Virtual Actors. In *Proceedings of the 18th Conference on Behavior Representation in Modeling and Simulation*, Sundance, UT, 31 March - 2 April 2009.

GIFT (2012) https://www.GIFTtutoring.org/projects/GIFT/wiki/Overview

Goldberg, B., Brawner, K., Sottilare, R, Tarr, R., Billings, D. & Malone, N. (2012) Use of Evidence-based Strategies to Enhance the Extensibility of Adaptive Tutoring Technologies. In *Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC).* Arlington, VA : National Training Systems Association

Graesser, A. C., D'Mello, S. K., Hu. X., Cai, Z., Olney, A., and Morgan, B. (2012). AutoTutor. In P. McCarthy and C. Boonthum-Denecke (Eds.), *Applied natural language processing: Identification, investigation, and resolution* (pp. 169-187). Hershey, PA: IGI Global.

Graesser, A.C., Conley, M., and Olney, A. (2012). Intelligent tutoring systems. In K.R. Harris, S. Graham, and T. Urdan (Eds.), APA *Educational Psychology Handbook: Vol. 3. Applications to Learning and Teaching* (pp. 451-473). Washington, DC: American Psychological Association.

Nye, B. D. (2013). Integrating GIFT and AutoTutor with Sharable Knowledge Objects (SKO). In R. A. Sottilare & H. K. Holden (Eds.) *Artificial Intelligence in Education (AIED)* 2013 Workshop on the Generalized Intelligent Framework for Tutoring (GIFT), (pp. 54-61).

Ryder, J., Graesser, A.C., McNamara, J., Karnavat, A., and Popp, E. (2002). A Dialog-Based Intelligent Tutoring System for Practicing Command Reasoning Skills. In *Proceedings of the Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC) 2002.* Arlington, VA : National Training Systems Association.

Adaptive Training in the Soldier-Centered Army Learning Environment (SCALE)

Rodney Long[1], Charles Amburn[1], Joanne D. Barnieu[2],
Shujing J. Hyland[2], Christopher E. Zoellick[2], Tracey A. Beauchat[3]
[1]U.S. Army Research Laboratory
[2]ICF International
[3]Dynamic Animation Systems

Introduction

In January 2011, the U.S. Army Training and Doctrine Command (TRADOC) published the "Army Learning Concept for 2015 (ALC 2015)," (Department of the Army, 2015) describing sweeping changes in the way Soldiers would be trained in the future. Noting that digital age learners are comfortable with technology, ALC 2015, now known as the Army Learning Model (ALM), describes how current and future technology can be leveraged to "make learning content more operationally relevant, engaging, individually tailored, and accessible." The concept moves away from instructor-led training using hundreds of presentation slides to small group, collaborative problem-solving environments; learning tailored to the individual's experience and skill level; access to content for learning and performance support using mobile technologies; and blended learning environments. In additional to being tailored to the needs of the learner, the instruction employs Live, Virtual, and Constructive simulations and games.

As part of collaborative research, the Soldier-Centered Army Learning Environment (SCALE) program, the U.S. Army Research Laboratory (ARL) and the Army Research Institute (ARI) are conducting research and development of advanced technology-enabled training methods and technologies. ARL is developing an architecture and tools for an integrated learning environment and assessing its effectiveness to develop, deliver, and track training and education. Consistent with ALC 2015 concepts, the SCALE architecture supports training and education across multiple hardware platforms (personal computer and mobile devices), using mobile applications, virtual classrooms, and virtual worlds. The architecture has been designed and developed to allow for researching the viability of providing different forms and capabilities in training (e.g., adaptive vs. non-adaptive training, intelligent tutoring capabilities, and reusable learning objects). In addition, several ongoing SCALE research initiatives include incorporation of the Experience Application Program Interface (xAPI), Social Media Framework, Interoperable Performance Assessments (IPA), and Generalized Intelligent Framework for Tutoring (GIFT).

As part of ARI's SCALE research in 2011–2013, a prototype training application was developed and used as a test bed at the Fort Gordon Signal School for conducting research on assessment strategies (Spain, Harris-Mulvaney, Cummings, Barnieu, Hyland, Lodato & Zoellick, 2013). The training content, which teaches Soldiers how to operate a common piece of signal equipment, the Joint Tactical Radio Systems (JTRS)-Enhanced Multiband Inter/Intra Team Radio (MBITR) (JEM), was delivered via emerging training technologies, such as a mobile device, virtual classroom, and game-based training platforms. The training also included a simulated radio for the learners to interact with during the instruction as well as during assessments. Since the prototype training was developed using a systematic Instructional System

Design (ISD) approach (Dick & Cary, 1990) and a detailed ISD map was created, the JEM content, including pre and interim assessment items, were learning-objective aligned. This made them suitable for incorporation into ARL's SCALE architecture to support research on adaptive training. Although the current research focuses on the impact of macro-sequencing (i.e., testing out of lessons via pre-assessment) and remediation strategies based on learner performance on interim assessments, future research could include other aspects of adaptive training, such as learner profile input (i.e., novice, journeyman, expert) or more customized remediation strategies for micro-sequencing, both of which could leverage features of GIFT 4.0.

The purpose of this paper is to provide background on the SCALE/GIFT architecture populated with JEM course content, a description of an in progress experimental plan to use the JEM course in the SCALE/GIFT prototype to answer research questions on adaptive training, and a preview of future research opportunities on adaptive training leveraging the SCALE/GIFT architecture.

Background

SCALE's data-driven architecture was designed to support adaptive training research by being highly tailorable through its external interfaces and its ability to include and /or develop new modules to support additional capabilities and technologies. Designed to support an ontology or hierarchy of concepts within a domain, the SCALE architecture currently supports adaptive training through the use of learning-objective driven content and assessments. Macro-sequencing via pre-assessment results and remediation strategies at the micro level, both representative of adaptive training, can be executed by leveraging GIFT (Sottilare, Goldberg, Brawner & Holden, 2012), an adaptive tutoring architecture that is also modular and service-oriented. Presenting learners with content that is matched to their aptitude or ability level provides learners with an appropriate level of challenge leading to optimal levels of learning performance (Orvis, Horn & Belanich, 2008; Vygotsky, 1978; Yerkes & Dodson, 1908).

SCALE's modular and web service-based architecture is illustrated in Figure 1. It was designed on top of the Drupal Content Management System (CMS) on the backend and provides a common Representational State Transfer (REST) interface on the front end. This design allows the SCALE architecture to be extended to support future requirements and allows the integration of existing and future technologies.

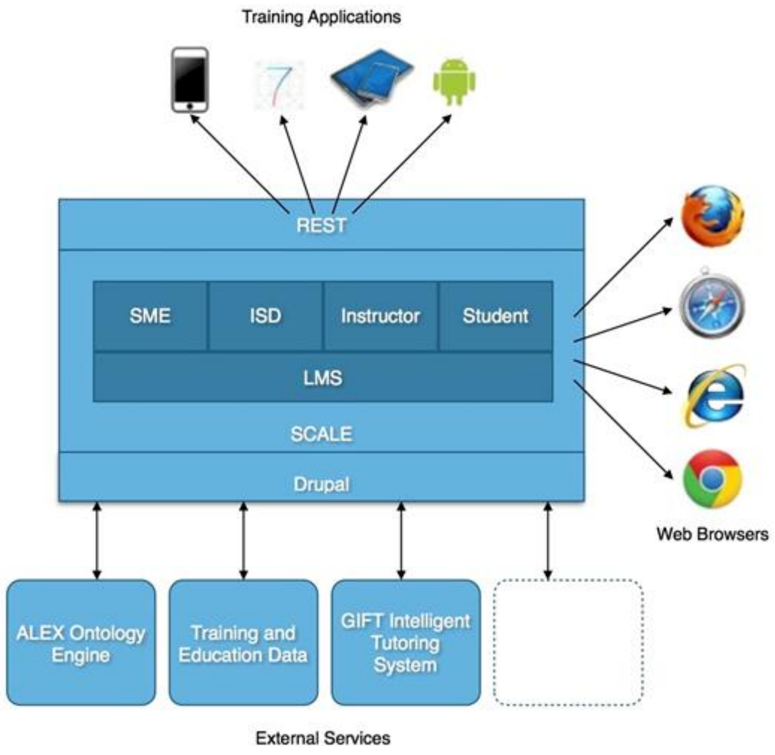


Figure 1. SCALE prototype architecture diagram

Training delivery structure

To support adaptive training and alignment with ISD principles, the SCALE training delivery structure was designed as shown in Figure 2. The JEM content was already segmented to support a systematic ISD approach with a main instructional goal, terminal objectives, and enabling objectives, which translated to a course, module, and lesson structure (Spain, Harris Mulvaney, Cummings, Barnieu, Hyland, Lodato & Zoellick, 2013). The SCALE architecture also includes a lesson element structure, i.e., a lower level structure resembling an authoring template that allows course authors to insert relevant assets and include transitional text. Assets, which are created by subject matter experts (SMEs), represent the smallest unit of learning for a particular concept. They include didactic text, procedural steps, audio files, video files, or animated sequences. Assessment questions are uploaded to SCALE and are accessible by the course author to create pre- and post-assessments at the module level and interim assessments at the lesson level, known as Check on Learning (COL) items.

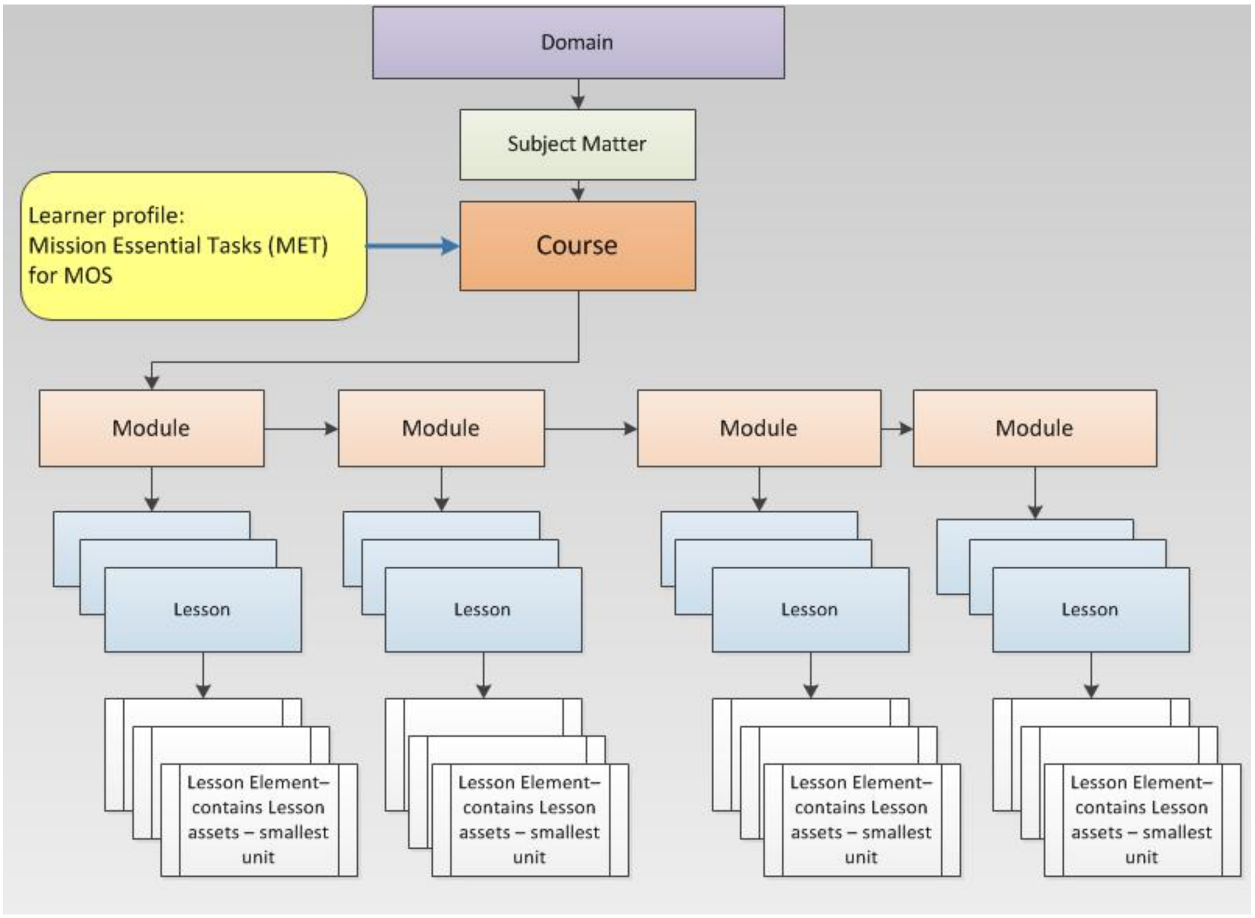

Domain
Subject Matter
Learner profile: Mission Essential Tasks (MET) for MOS
Course
Module
Module
Module
Module
Lesson
Lesson
Lesson
Lesson
Lesson Element– contains Lesson assets – smallest unit
Lesson Element– contains Lesson assets – smallest unit
Lesson Element– contains Lesson assets – smallest unit
Lesson Element– contains Lesson assets – smallest unit

Figure 2. SCALE training delivery structure

GIFT integration

As previously discussed, one of the external services that feeds into the SCALE architecture is GIFT version 2.0. Developed by ARL, GIFT is a modular tutoring system framework to support research and development of adaptive tutoring concepts, models, authoring capabilities, and instructional strategies (Sottilare, Goldberg, Brawner & Holden, 2012). The SCALE architecture passes learner assessment data through GIFT for input on appropriate remediation or advancement strategy. Currently, the SCALE prototype contains pre-assessments used to measure a learner's current knowledge and interim assessments used to measure a learner's gained knowledge. Based on their performance on a pre-assessment, learners can be alerted to those lessons for which they have already illustrated mastery, and the learner can choose to skip these lessons. When a learner answers an interim assessment item incorrectly, the learner receives a hint and can answer the question a second time. If the learner is still unable to answer the question correctly, the learner is returned to the lesson content to review the related material.

JEM Assessments

The SCALE/GIFT prototype uses the same assessment questions as those developed as part of the original ARI research project involving JEM training. To support the relevant research questions for the ARI research project, a suite of assessments were developed, including a Computer Adaptive Test (CAT) administered at three points throughout the training and COLs that were periodically administered throughout the training. Both the CAT items and the COLs were specifically aligned with the ISD map (one-to-one correspondence with each enabling learning objective). As a result of this alignment, the CAT items were easily reused in the ARL SCALE prototype to create pre-assessment items at the module level, allowing for macro-sequencing resulting in lesson collapsing for those students who demonstrate pre-requisite knowledge of particular lesson content. Additionally, the learning objective-driven COL items were used for the same purpose, specifically to assess knowledge transfer after each lesson. To set the foundation for multiple remediation strategies that could eventually be pushed out based on learner data, each COL has an associated hint in case the student selects the incorrect answer. At present, all students who answer the question incorrectly the first time receive the hint and a second opportunity to answer the question correctly. The second remediation strategy is implemented when the student still selects the incorrect answer even after receiving the hint. When this occurs, the student is returned to the lesson content for a second review.

Adaptive Training Research Study

The integrated assessments in the SCALE prototype as well as a fully developed JEM course have set the stage for conducting research on adaptive training, which was planned for the summer of 2014. There were additional advantages to using the JEM course as a test bed for the adaptive training research. Leveraging an existing relationship with the Fort Gordon Signal School, there was access to students as study subjects and to classrooms and equipment, as required. Additionally, the data collection from the ARI project yielded an experienced research team already familiar with the radio and prepared to administer post-assessment criterion measures.

Research study summary

ARL is currently leveraging the SCALE prototype to address research questions surrounding adaptive training. In addition to training efficiency and effectiveness, the research questions concern how adaptive training impacts learner motivation, learner engagement, and learner perception of the usefulness of certain adaptive training strategies. Although research has demonstrated that learners prefer a challenging environment (Belanich, Sibley & Orvis, 2008) and that learner reactions predict post-training motivation and post-training self-efficacy (Sitzmann, Brown, Casper, Ely & Zimmerman, 2008), not much is known concerning the ability of adaptive training to impact learner reactions, such as motivation and engagement. It is assumed that presenting training that best suits the their needs would lead to higher levels of engagement, higher levels of motivation, as well as fewer maladaptive responses elicited by individuals who find a task too challenging and demonstrate decreased persistence and effort. In context of the adaptive JEM course that was used for the study scheduled for the summer of 2014, students would be at various levels of proficiency with the JEM and similar radio equipment. We would expect that students at the expert level would be more engaged with the training if they are able to skip topics on which they are already knowledgeable. Similarly, we would expect students at the novice level may be more motivated during the training if they receive remediation to assist them through challenging topics.

Adaptive training has the potential to reduce overall training time and tailor the length of the training to the needs of the learner (Landsberg, Astwood, Townsend, Steinhauser & Mercado, 2012; Park & Tenny-son, 1980; Romero, Ventura, Gibaja, Hervas & Romero, 2006). In the study, the length of time spent on the training in conjunction with post-assessment results will provide data on training efficiency. Pre- and post-assessment data, as well as a capstone exercise, will determine training effectiveness. Also, students will provide data on learner motivation, engagement, and usefulness of adaptive training strategies.

Study design and measures

The adaptive training research study will involve two groups: an experimental group that will receive the adaptive version of the training (includes both macro and micro-sequencing) and a control group that will receive the non-adaptive version of the training (all lessons required with no remediation opportunities). Students will be utilizing a mobile version of the training using iPad technology. The two groups will be randomly assigned to maximize the likelihood of equivalence between groups. At least 30 participants should be assigned to each group. The students will begin with a pre-assessment to measure pre-requisite knowledge. Each group will be administered the appropriate version of the training. Following the training, students will participate in a post-assessment to measure knowledge transfer. Finally, students will participate in a capstone exercise for which measures were already created in the previous ARI project. This capstone measures student performance using an actual radio. To measure learner reactions, the students will complete surveys containing reaction items.

Research questions

Table 1 lists the research questions that the study will address. These results will be published in a report at the conclusion of the study.

Table 1. Adaptive training research questions

Measurement	Question
Reaction Data	Does adaptive training lead to higher levels of learner engagement and motivation?
Reaction Data – Useful of Adaptive Training Strategies	If a module does not contain all of the lessons due to the learner's performance on the pre-assessment, does the learner perceive the module content sequencing as disjoint-ed or disorganized? Does this sequencing impact the learner's engagement and motivation?
Reaction Data – Useful of Adaptive Training Strategies	If modules with one or more lessons are collapsed do they yield negative learner reactions? If so, what variations to the adaptive training strategy could reduce or prevent these negative consequences?
Training Efficiency	Does the adaptive training facilitate training efficiency? Do students spend less time on the training using the adaptive course structure strategies? Do students who spend more time due to remediation at least show improvement in knowledge transfer?
Training Effectiveness (assessments)	How does adaptive training impact acquisition of Knowledge, Skills, Abilities, and Other (KSAOs)? Do students receiving adaptive training show greater improvement on the post-assessment? What are the reasons why or why not?
Training Effectiveness (capstone)	How does this type of adaptive training impact job performance? Do students receiving adaptive training perform better on the post-training capstone exercise?

Future Research

Learner profile and macro-sequencing

As seen in Figure 3, one of the components of GIFT is the learner data, which informs the learner state and consequently the instructional strategy. This sequence of events is known as the adaptive tutoring learning effect chain (Sottilare, Goldberg, Brawner & Holden, 2012).

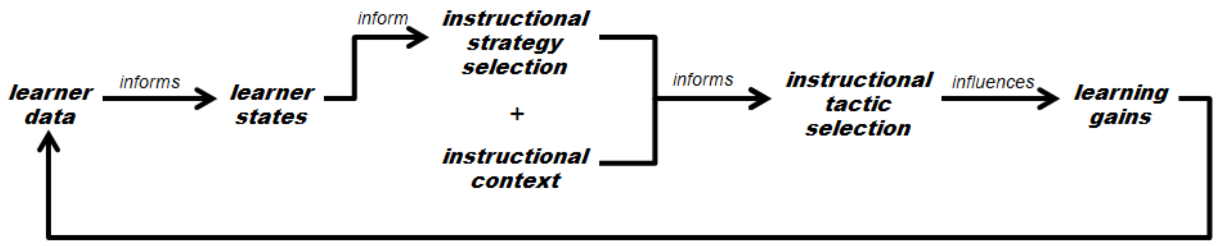


Figure 3 – Adaptive Tutoring Learning Effect Chain

The SCALE architecture currently contains assets that could be tagged as appropriate for certain learner profiles (i.e., expert level). For example, in the case of the JEM, one asset is an animated video of how to attach the battery to the Radio Transmitter Unit (RTU). Learner data such as the Soldier's Military Occupational Specialty (MOS) could inform learner state to be expert. This could inform the strategy of displaying only the animated video for this lesson, since the expert student may not need didactic text or procedural steps for attaching the battery.

Learner profile and micro-sequencing

A future functionality could be implemented to push out the appropriate remediation strategy based on COL performance per lesson. Currently, there is one hint available for all incorrect answers. In the future, customized hints could be authored that tailor to a particular answer the student selects. The answer a learner selects could be considered representative of the learner state. For example, selecting the second best answer could reveal that the learner is familiar yet still does not fully understand the concept. The associated hint could be minimal and just enough to provoke a thought. Furthermore, if the learner selects the least favorable answer, he or she may not receive a hint but would instead be remediated back to the lesson content.

Conclusions

ARL continues to mature the SCALE architecture and tools for an integrated learning environment and continues to assess its effectiveness to develop, deliver, and track training and education. Ongoing SCALE research initiatives are beginning to mature and will be integrated into the SCALE architecture, including incorporation of the Social Media Framework, Reusable Learning Objects research, and IPA. The SCALE architecture will also be upgraded to the latest version of GIFT, to investigate the macro-sequencing and micro-sequencing strategies described above. This will allow ARL the opportunity to continue research on the effectiveness of adaptive training in support of the ALM.

References

Belanich, J., Sibley, D. E. & Orvis, K. L. (2004). *Instructional characteristics and motivational features of a pc-based game*. U.S. Army Research Institute for the Behavioral and Social Sciences Technical Report 1822, Arlington, VA.

Department of the Army. (2011). The U.S. Army Learning Concepts for 2015. Washington: Department of Defense.

Dick, W. & Cary, L. (1990). The Systematic Design of Instruction, Third Edition, London: Harper Collins.

Landsberg, C.R., Astwood, R. S., Townsend, L. N., Steinhauser, N. B. & Mercado, A. D. (2012). Review of adaptive training system techniques. *Military Psychology (Taylor & Francis Ltd), 24*(2), 96-113. doi: 10.1080/08995605.2012.672903

Orvis, K. A. , Horn, D. B. & Belanich, J. (2008). The roles of task difficulty and prior videogame experience on performance and motivation in instructional videogames. *Computers in Human Behavior, 24*(5), 2415-2433. doi: 10.1016/j.chb.2008.02.016

Park, O. & Tennyson, R. D. (1980). Adaptive design strategies for selecting number and presentation order of examples in coordinate concept acquisition. *Journal of Educational Psychology, 72*(3), 362-370. doi: 10.1037/0022-0663.72.3.362

Romero, C., Ventura, S., Gibaja, E. L. , Hervás, C. & Romero, F. (2006). Web-based adaptive training simulator system for cardiac life support. *Artificial Intelligence In Medicine, 38*(1), 67-78. doi: 10.1016/j.artmed.2006.01.002

Sitzmann, T., Brown, K. G., Casper, W. J, Ely, K. & Zimmerman, R. D. (2008). A review and meta-analysis of the nomological network of trainee reactions. *Journal of Applied Psychology, 93*(2), 280.

Sottilare, R. and Goldberg, B. Designing Adaptive Computer-Based Tutors to Accelerate Learning and Facilitate Retention. *Journal of Cognitive Technology: Contributions of Cognitive Technology to Accelerated Learning and Expertise* 2012, 17, 1, 19–34.

Sottilare, R.; Goldberg, B.; Brawner, K.; Holden, H. 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, FL, December 2012.

Spain, R., Harris Mulvaney, R., Cummings, P., Barnieu, J., Hyland, J., Lodato, M. & Zoellick, C. Enhancing Soldier-Centered Learning with Emerging Training Technologies and Integrated Assessments. In Proceedings of the *Interservice/Industry Training Simulation & Education Conference*, Orlando, FL, December 2013.

Vygotsky, L. (1978). *Mind in society: The development of higher mental processes*. Cambridge, MA: Harvard University Press.

Yerkes, R. & Dodson, J. (1908). The relation of strength of stimulus to rapidity of habit formation. *Journal of Comparative and Neurological Psychology*, *18*, 459-482.

Theme III: User Perspectives of GIFT

An Analysis of GIFT User Experiences and Design Recommendations for Enhancing GIFT's Usability

Heather Holden[1] and Marcus Alexander[2]
[1]U.S. Army Research Laboratory
[2]Bethune-Cookman University

Introduction

Over the past four years, the developers of the Generalized Intelligent Framework for Tutoring (GIFT) have focused on providing Intelligent Tutoring System (ITS) researchers with an experimental test bed to assess learning effectiveness in a domain-independent fashion. The GIFT developers have conducted advisory boards and produced a book series to identify the key challenges relating to ITSs and provide design recommendations for how GIFT can address such challenges. An additional purpose of these efforts is to maximize GIFT's acceptance among the ITS community by understanding the essential features and functionalities necessary to support the community's research and development requirements.

Although GIFT is a research prototype, it is now increased its user population to over 300 users. Therefore, the most pressing concern is to maximize GIFT's usability and interaction design in order to promote positive user experiences. In order for interactive technologies to be truly effective and enjoyable from a user's perspective, they have to be designed for usability and the user experience. The purpose of this paper is to present the results of a survey distributed to GIFT users on their perceptions of GIFT's usability and their experiences. This paper also outlines the key elements of interactive design principles and provides the design recommendations for future enhancements of GIFT to maximize the user experience.

Usability vs. User Experience

What is the difference between usability and user experience? Usability is ensuring that an interactive product is easy to learn and effective to use. Common usability criteria include effective to use (effectiveness), efficient to use (efficiency), safe to use (safety), having good utility (utility), easy to learn (learnability), and easy to remember how to use (memorability) (Preece, Rogers & Sharp, 2001). Good usability ensures that the user can successfully accomplish his/her goals due to the technology collectively addressing these quantitative criteria. On the other hand, user experience relates to how a user feels about the system. What is the emotional connection, if any, between the user and the system? User experience goals are more subjective in nature and can include elements evaluating the degree to which as user perceives the technology as satisfying, enjoyable, engaging, motivating, aesthetically pleasing, cognitively stimulating, supportive of creativity, etc. (Preece et al., 2001). Negative perceptions of frustration, annoyance, and boredom exhibited by a technology are also metrics that can be used to access user experience. The evaluation of what is being built throughout the process and the user experience it offers are the core processes of interaction design (Preece et al., 2001). For example, the computer and video game industry have had great success in understanding the important relationship between and designing for both user experience and usability. This is evident by their continuous increase in revenue (in billions) yearly.

Designing for usability and user experience

While designing for the user experience may be more challenging than designing for usability, designing for both will produce maximum user acceptance and usage. There are many types of design principles of interaction design at have been developed by key experts in the human-computer interaction field:

- **Don Norman's "Principles of Design,"** which include visibility, feedback, constraints, mapping, consistency, and affordance (Norman, 2002).

- **Jakob Nielsen's "10 Usability Heuristics for User Interface Design,"** which 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; helping users recognize, diagnose, and recover from errors; and help and documentation (Nielson, 2005).

- **Ben Shniederman's "Eight Golden Rules of Interface Design,"** which include strive for consistency; enable frequent users to use shortcuts; offer informative feedback; design dialog 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).

Of the three lists of design principles presented above, visibility, consistency, and feedback seem to be the main reoccurring principles; nevertheless, these principles can be useful for designing, developing, and evaluating aspects of any interactive technology.

Unfortunately, thorough usability and user experience studies and testing takes time. A quicker way to understand users' perceptions toward an interactive technology is to survey them and include elements of the Technology Acceptance Model (TAM). TAM is a theoretical model that predicts how a user comes to accept and use a given information technology. It specifies casual relationships among external variables, belief and attitudinal constructs, and actual usage behavior (Davis, 1989). The model suggests that when users are presented with a particular information technology, a number of factors, notably Perceived Usefulness (PU) and Perceived Ease of Use (PEU), influence their decision of how and when they will use the technology. The original TAM evaluates users' cognitive, affective, and behavioral responses toward the particular technology in question. The PU and PEU elements represent users' cognitive responses to using the technology. These cognitive responses then influence the users' affective (attitude toward using) and behavioral (future usage intentions) responses toward using the technology (Davis, 1989). The survey of discussed in this paper only included the PU, PEU, and future usage intentions elements of the TAM.

Methodology

A GIFT satisfaction survey was developed and sent out to the GIFT community via email. Of the 300+ GIFT users registered on gifttutoring.org, 8 users responded to the survey. It is important to note that the GIFT developers did not participate in this survey to ensure the results were not biased. Four users of the respondents were researchers/analysts, two users were ITS authors/training developers, one user was an instructor/trainer, and one user was a software developer. The respondents identified the following as primary purposes for using GIFT: seven users reported using GIFT for research and development purposes; three users reported using GIFT for experimentation; and two users reported using GIFT for evaluation of its capabilities. The questions of the survey were divided into the following sections:

1. GIFT's Installation, Documentation and Support (4 questions)
2. GIFT's Perceived Ease of Use (11 questions)
3. GIFT's Perceived Usefulness (5 questions)
4. The Future Use of GIFT (2 questions)
5. Potential Features and Functionalities that could be added to GIFT (i.e., recommendations for improvement) (5 questions)

Results for results for each section are provided in the next segment of this paper.

Results and Discussion

GIFT's installation, documentation, and support

Table 1 presents the results of users' perceptions of GIFT's installation, documentation, and support. The questions of this section were based on a 6-pt Likert Scale where 1 = strongly dissatisfied and 6 = strongly satisfied.

Table 1: Results of perceptions of GIFT installation, documentation and support

#	Statement:	Min., Max.	Mean	Std. Dev.
1	The ease of installation of GIFT	1,6	4.00	1.852
2	The completeness and accuracy of GIFT installation instructions	1,6	4.13	1.553
3	The completeness and usefulness of the GIFT user documentation	1,5	3.63	1.408
4	The availability of technical support (i.e., gifttutoring.org forums, etc.)	4,6	4.88	0.991

These findings suggest that GIFT users generally find GIFT easy to install (M = 4.0, SD = 1.852). They are generally satisfied with the completeness and accuracy of the GIFT instructions (M = 4.13, SD = 1.553) as well as the completeness and usefulness of the GIFT documentation (M = 3.63, SD = 1.408). They are highly satisfied with the availability of GIFT technical support (M = 4.88, SD = 0.991).

GIFT's perceived ease of use

Table 2 presents the results of GIFT's perceived ease of use. PEU is defined as "the degree to which a person believes that using a particular system would be free from effort" (Davis, 1989). The questions of this section were based on a 6-pt Likert Scale where 1 = strongly disagree and 6 = strongly agree.

Table 2: Results of GIFT's PEU

#	Statement:	Min., Max.	Mean	Std. Dev.
1	My interaction with GIFT is clear and understandable.	1,4	2.63	1.188
2	Interacting with GIFT does not require a lot of mental effort.	1,5	2.75	1.488
3	I find GIFT to be easy to use.	1,5	2.38	1.302
4	I find it easy to get GIFT to do what I want it to do.	1,4	2.63	1.061
5	I have control over using GIFT.	1,4	3.25	1.165
6	I find using GIFT enjoyable.	1,4	2.88	0.991
7	I find GIFT to be flexible to interact with.	1,4	2.25	1.035
8	Learning how to perform tasks using GIFT was easy.	1,5	2.75	1.488
9	GIFT has good functionality (features).	4,5	4.50	0.535
10	I feel I have an intuitive sense on how to operate GIFT.	1,6	2.88	1.642
11	I find it easy to remember how to perform tasks using GIFT.	1,5	3.63	1.302

The primary findings suggest that users think GIFT has good functionality and features (*M* = 4.50, *SD* = 0.535) and that it's easy to remember how to complete tasks in GIFT (*M* = 3.63, *SD* = 1.302). However, they do not feel as though GIFT is easy to use. While this is the case, these results can serve a baseline for future satisfaction surveys. A recommendation based on these results for future GIFT development would be to increase the flexibility in the authoring capabilities that GIFT can support.

GIFT's perceived usefulness

Table 3 presents the results of GIFT's perceived usefulness. PU is defined as "the degree to which a person believes that using a particular system would enhance his or her job performance" (Davis, 1989). The questions of this section were based on a 6-pt Likert Scale where 1 = strongly disagree and 6 = strongly agree.

Table 3: Results of GIFT's Perceived Usefulness.

#	Statement:	Min., Max.	Mean	Std. Dev.
1	Using GIFT improves my performance in my job.	1.4	2.63	1.188
2	Using GIFT in my job increases my productivity.	1,4	2.50	1.195
3	Using GIFT enhances my effectiveness in my job.	1,4	2.75	1.035
4	I find GIFT to be useful in my job.	1,5	3.13	1.356
5	In my job, usage of this technology is relevant.	2,6	4.38	1.302

Based on these results, respondents do not feel as though GIFT improves their job performance (M = 2.63, SD = 1.188), increases their productivity (M = 2.50, SD = 1.195), or enhances the effectiveness of their job (M = 2.75, SD = 1.035). However, they do agree that GIFT is a relevant technology in their job (M = 4.38, SD = 1.302). In essence, GIFT's perceived usefulness among the respondents is currently low. This is not surprising for two reasons: (1) PEU is a primary element that contributes to PU in TAM and (2) the level of PU depends on the necessity of the technology directly impacting job productivity. For example, email applications will produce high levels of PU as it is heavily intertwined with most communication among employees. When revisiting the job categories of GIFT users and their purposes for using GIFT, it is apparent that the roles and responsibilities of such categories are vast and their use of GIFT is a small part. As PEU increases, so will PU, especially since the respondents feel that GIFT is a relevant technology for their job. One recommendation, based on these findings, to increase GIFT's PU is to highlight and extend the features and functionalities of GIFT based on the roles of different types of users.

Future use of GIFT

Table 4 presents the results of user's likelihood of future GIFT usage. The questions of this section were based on a 6-pt Likert Scale where 1 = very unlikely and 6 = very likely.

Table 4: Results of likelihood of future GIFT usage

#	Statement:	Min., Max.	Mean	Std. Dev.
1	What is the likelihood that you will use GIFT in the future for the purposes you previously identified?	3,6	5.13	1.126
2	What is the likelihood that you will contribute to the future of GIFT by expanding the code base and returning your findings to the GIFT community?	3,6	4.63	0.916

These findings suggest GIFT users are very likely to continue to use GIFT in the future (*M* = 5.13, *SD* = 1.126) and are likely contribute to the future expansion of GIFT code base by returning their findings/changes back to the GIFT community (*M* = 4.63, *SD* = 0.916). This is important as it shows that GIFT users appreciate the value and motivation behind GIFT development. This also supports the previous finding of GIFT being a relevant technology in the jobs of the respondents.

Additional features and functionality needed

This section consisted of five open-ended questions. Presented below are the questions and the responses for each.

1. **What authoring features do you need in your ITS framework that GIFT does not provide?**
    a. Natural Language Conversation
    b. Rapid iteration features (survey authoring, course authoring)
    c. A way to tutor teams
    d. A more logical course authoring system that isn't based on editing XML files. Something that is more flowchart based.
    e. Unsure/None

2. **What instructional features do you need in your ITS framework that GIFT does not provide?**
   a. Integration with an LMS
   b. Ability to prioritize which instructional strategy is selected for remediation if multiple conditions are simultaneously below expectation.
   c. In the XLM tree structures, it's easy to lose track of where you are in the tree and what you are adjusting or what you have to add.
   d. Have not tried long enough to say about this/Unsure
3. **What analysis features do you need in your ITS framework that GIFT does not provide?**
   a. For the expected time of finishing the activity, it would be nice to have a range or a probability distribution. Also, more granularity according to the performance (not just below, at or higher than expectation) such as author defined levels of performance, using more sophisticated rules.
   b. Merging multiple data channels into a dataset (e.g., sensor data, log data). Data exploration tools.
   c. The ability of analyze different members of a team at the same time.
   d. Have not tried long enough to say about this/None/Unsure/N/A
4. **Please list at least 1 suggestion for how to make GIFT more user-friendly or easier to use.**
   a. Produce a demo video (or set of videos) to describe each of the features/functions.
   b. The XML tree needs to be polished. The topic of each branch and the parent can be highlighted in a bigger font or something. The validation of the tree is not real-time. You need to save the file, then validate it, then save it again. The buttons are not easy to find at some places.
   c. Install wizard with options for configuring GIFT setup. Minimize editing XML files during installation. Also, handle dependency issues (e.g., MySQL, PowerPoint) gracefully. If non-essential software is missing, don't prevent GIFT from running. Also, could be useful to conduct heuristic evaluation on survey authoring and course authoring systems. While these are great features, there are too many mouse clicks to set things up. Feature discoverability could also be strengthened.
   d. Documentation might be enhanced a little bit to help new GIFT users and developers.
   e. Expanded technical documentation for engineers/system integration. (e.g., use of XML config files, expected order of message generation and receipt during processing).
5. **Are there any features and functionality that could be added to GIFT?**
   a. Additional API calls surrounding the message system. For example, ability to add new message types and tell the related components how to listen for the new messages and respond to them.
   b. A more robust domain logic API so that other simulations could send messages about their data without having to create new Conditions. And perhaps clearer documentation on how this could work in a non-trivial case.
   c. Previous feedback was based on GIFT 3.0 and earlier as opposed to the latest version (4.x). Despite critical feedback above, I'm very enthusiastic about GIFT. But I think there is a lot of opportunity for improving the user experience.
   d. Have not tried long enough to say this.

These five open-ended questions provide a qualitative aspect to this satisfaction survey. The answers to these questions provide insight of ways to increase GIFT's usability and user experiences. They also help explain the quantitative results of the survey previously presented.

Conclusions

Overall the respondents of the survey exhibited positive perceptions toward GIFT in terms of future usage intentions as well as its installation, documentation, and support. However, their perceived ease of use and usefulness of GIFT is low, except for a few notable questions discussed the findings. Nevertheless, these finding will serve as a baseline for future satisfaction surveys. They appreciate the motivation and purpose of what GIFT is aiming for, now and in the future. The good news is that most of the desired features and functionality provided in the survey results are on the GIFT developers' plans for implementation in future releases. Some of the requested features, such as a more user-friendly course development authoring tool, are in the latest GIFT version, GIFT-2014-1X (released April 2014). As the GIFT developers' continue to increase GIFT's usability and user experience, future satisfaction surveys will be presented to the GIFT community to ensure the progression of maximizing usage and acceptance.

References

Davis, F. D. (1989). Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology. *MIS Quarterly, 13*(3), 319-340.

Nielson, D. (2005). Ten Usability Heuristics for User Interface Design. Retrieved April 2014, from www.nngroup.com/articles/ten-usability-heuristics

Norman, D. (2002). *The Design of Everyday Things*. New York, NY: Basic Books.

Preece, J., Rogers, Y. & Sharp, H. (2001). *Beyond Interaction Design: Beyond Human-Computer Interaction*: 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.

The Research Psychologist’s Guide to GIFT

Anne M. Sinatra
U.S. Army Research Laboratory
Oak Ridge Associated Universities

DON’T PANIC

I have a vivid memory of being a first-year graduate student sitting in my psychology research lab. I had decided that I wanted to learn how to use our lab’s eye-tracking equipment for a study that I would be conducting. The first step I took in trying to achieve this seemingly daunting task was to read the documentation that a former graduate student in our lab had left behind. I noticed that on the top of the page the phrase “DON’T PANIC” was typed in big bold letters. At first I looked at it puzzled, and then I realized that it served two purposes: 1) it was a message to all incoming graduate and undergraduate students that despite looking complicated, it was possible to operate the eye-tracker, and 2) it was a clever reference to the *Hitchhiker’s Guide to the Galaxy*, which stated that “It is said that despite its many glaring (and occasionally fatal) inaccuracies, the *Hitchhiker’s Guide to the Galaxy* itself has outsold the *Encyclopedia Galactica* because it is slightly cheaper, and because it has the words ‘DON’T PANIC’ in large, friendly letters on the cover” (Adams, 1979). After that experience, I decided that I would include the phrase “DON’T PANIC” at the beginning of instructional manuals that I wrote to provide reassurance and a laugh to those who read them. That is why, I now present to you, *The Research Psychologist’s Guide to GIFT*, a wholly accurate and encouraging guide for those who wish to use the Generalized Intelligent Framework for Tutoring (GIFT) in their own research and that has the phrase “DON’T PANIC” in large friendly letters on the top of the page.

The Generalized Intelligent Framework for Tutoring

What is GIFT?

GIFT is an open-source, domain independent intelligent tutoring framework, which has multiple functions (Sottilare, Brawner, Goldberg & Holden, 2013; Sottilare & Holden, 2013). It provides the tools for an individual instructor or researcher to create complete intelligent tutoring systems (ITSs) that can be used as a component of the classroom or as independent instruction. Additionally, GIFT can be used to conduct research. The functionality of GIFT and the tools included with it allow for the development of both experiments that examine individual components of ITSs, as well as more traditional psychology experiments in a computerized environment. In fact, GIFT has been used to conduct experiments on the self-reference effect and on receiving feedback from in-game vs. out of game characters (Goldberg & Cannon-Bowers, 2013; Sinatra, 2013).

Why should a research psychologist use GIFT?

For myself, as a research psychologist, I have found GIFT to be very useful in running experiments. While GIFT offers the capability of adapting to the individual learner and giving specific feedback based on performance, it also provides a means to present a traditional experiment.

Many of the experiments that research psychologists conduct involve providing materials (with a manipulation of interest) and questionnaires to individual participants. Generally, in an experiment, except for the manipulation of interest, all of the materials, surveys, and their order of presentation are expected to remain constant. One difficulty that researchers often run into is the need to have undergraduate research assistants and research proctors present in order to run an experiment. The role that these proctors serve at times includes reading instructions to the participant, opening and closing programs, opening and closing websites, and providing printed surveys/questionnaires to fill out.

GIFT provides researchers with the ability to create an individual course, which can automate many of the connective processes that research assistants traditionally are responsible for during an experimental session. With GIFT, the researcher can create a linear flow for their experiment, such that participants are given guidance, surveys, and program materials in a desired order. Currently, GIFT can interface with PowerPoint (in the form of a PowerPoint show, .ppsx or .ppsm), TC3Sim/vMedic, and VBS2. The ability to integrate a PowerPoint show into GIFT is extremely powerful, as Visual Basic for Applications (VBA) can be used to make PowerPoint interactive through the use of macros and simple visual basic programming. Additionally, many instructional PowerPoint files are readily available, or can easily be created by an individual that does not have a computer science background. Once the GIFT course is constructed and run, the desired information, surveys, and programs (e.g., PowerPoint) can be automatically opened and closed as appropriate during participation. As the number of research assistants that are available is often limited, automating these experimental processes can increase the efficiency of those who are running participants and allow for multiple participants to be run simultaneously. Further, it removes the possibility of human error in opening and closing the wrong computerized windows or providing the wrong surveys. A further benefit of automating these processes is that research assistants can be shifted to different tasks that may offer a richer research experience for them, such as coding and analyzing data.

GIFT provides researchers with the ability to create their own surveys and questionnaires that can be administered on the computer as part of a GIFT course. These surveys are hosted on the computer and do not need an internet connection in order to be completed. After data collection has been completed, the responses that were entered by the participants can be exported into a format that is easily compatible with Excel and SPSS using the Event Reporting Tool (ERT).

Using GIFT to Create an Experiment

To create a traditional psychology experiment using GIFT, the researcher will use separate tools that are provided to create surveys, create the flow of their "course," and export the data. This paper focuses on the three tools and steps that are most crucial to creating an experiment using GIFT. There are also additional tools available to researchers who wish to include sensors and real-time adaptive feedback in their experiments.

Designing your course

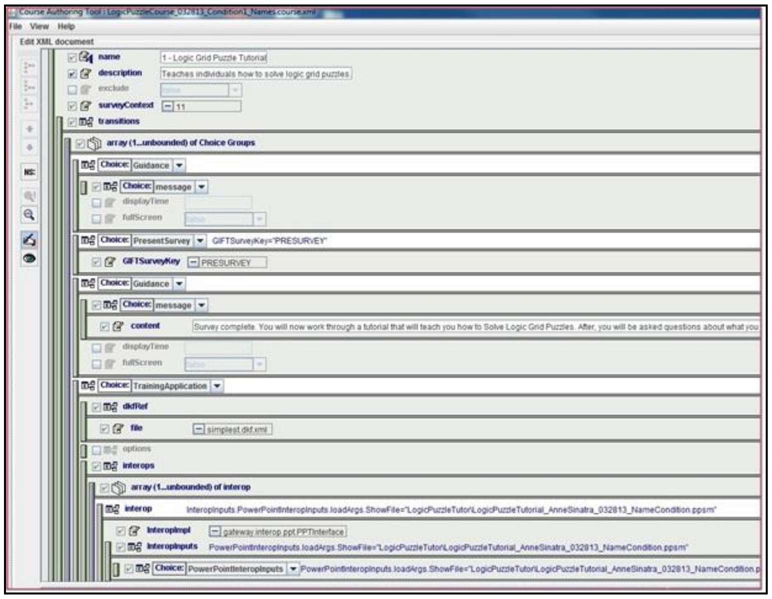
In GIFT, a "course" is the file that is generated by the researcher, which determines the flow of experiment and the activities that occur during it. Your "course" is designed with the Course Authoring Tool (CAT) or the GIFT Authoring Tool (GAT), which is included in the Administrator Tools of GIFT. In

GIFT 4.0 and earlier versions, the CAT is the only tool provided to design a course. However, in post-GIFT 4.0 versions, starting with GIFT 2014-1X, a user friendly CAT, the GAT, is also available in addition to the original CAT.

It is advisable that before sitting down to design your course, that you create an outline for your study, including each of the components that you want in it and their order. Doing this ahead of time will save time once you begin to use the authoring tools and will alert you to other steps that you may need to take before completing the design of your course.

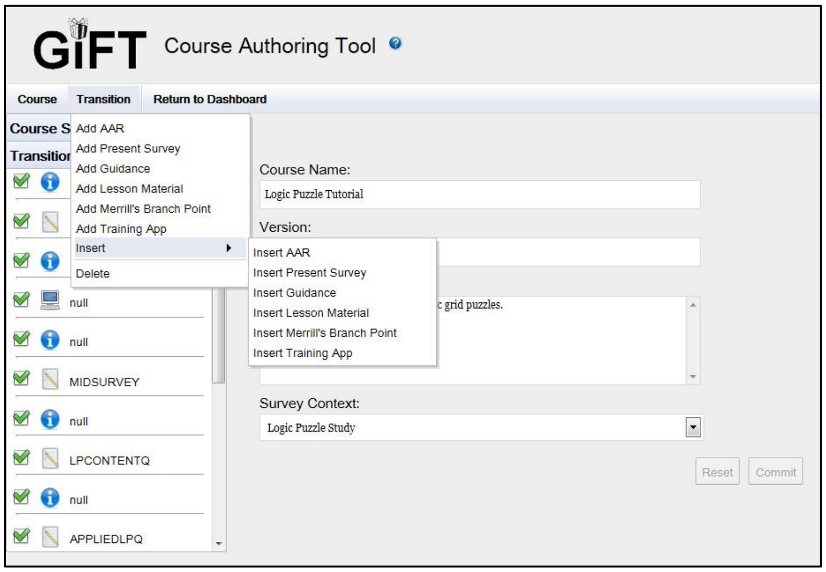
Both the CAT and GAT provide similar features and functionality in regard to GIFT course design. However, the user-interface that the experimenter interacts with to create the course is very different. The CAT is an .xml editor based tool, which allows users to add and expand nodes that represent components of their course. The GAT is a more visual tool that leads the user through the experience of designing a course with the use of more traditional menus and includes drag-and-drop functionality to rearrange course elements. For new users, and those who do not have a background in computer science, the GAT will likely be the preferred tool to use to create a GIFT course.

Both course authoring tools allow you to provide guidance (messages given to the participants about what to expect and a continue button), provide questionnaires/surveys, go to external web sites, as well as open and close programs (such as PowerPoint). When you first create a new course using either tool, you will be prompted to give it a name and a description. Next, you will need to select a survey context, which indicates the pool of surveys that you will be using during your experiment. The survey context is created using the Survey Authoring System (SAS) prior to creating your course. The CAT is set up in the form of a customized .xml editor that has nodes that represent each course element. Nodes can be added and deleted. Additionally, you can reorder the nodes if you wish, after they are created. See Figure 1 for an example of a course loaded in the CAT. In the GAT, course elements are added through the use of a menu and are represented on the left side of the users screen. See Figure 2 for an example of a course loaded in the GAT. When you create a node in the CAT or create a new course element in the GAT, you have the option of what you would like it to provide. In an experiment the options that you are most likely to use are: Guidance, Training Application, and Present Survey. “Guidance” are text-based messages that are given to the participant during the experiment. “Training Applications” are external programs that your course will be launching. Currently, PowerPoint, VBS2, and TC3Sim/vMedic are supported. For each Training Application that you add, you will need to also provide an associated domain knowledge file (DKF). If you do not intend to use adaptive feedback during the application, then you can use the provided “simplest.dkf”. However, if you wish to have adaptive feedback you will need to create your own DKF using the DKF Authoring Tool (DAT) or Student Information Models for Integrated Learning Environment (SIMILE) workbench, which is included with GIFT. As the majority of psychology experiments do not offer adaptive feedback, creating a DKF is beyond the scope of this paper. When the “Present Survey” option is selected, you can choose from the surveys that are in the survey context, which you set for your specific course.



```
Course Authoring Tool : LogicPuzzleCourse_032813_Condition1_Names.course.xml
File  View  Help
Edit XML document
  name            1 - Logic Grid Puzzle Tutorial
  description     Teaches individuals how to solve logic grid puzzles.
  exclude         false
  surveyContext   11
  transitions
    array (1...unbounded) of Choice Groups
      Choice: Guidance
        Choice: message
          displayTime
          fullScreen      false
      Choice: PresentSurvey   GIFTSurveyKey="PRESURVEY"
        GIFTSurveyKey   PRESURVEY
      Choice: Guidance
        Choice: message
          content         Survey complete. You will now work through a tutorial that will teach you how to Solve Logic Grid Puzzles. After, you will be asked questions about what you
          displayTime
          fullScreen      false
      Choice: TrainingApplication
        dkfRef
          file            simplest.dkf.xml
        options
        interops
          array (1...unbounded) of interop
            interop         InteropInputs.PowerPointInteropInputs.loadArgs.ShowFile="LogicPuzzleTutor\LogicPuzzleTutorial_AnneSinatra_032813_NameCondition.ppsm"
              InteropImpl     gateway.interop.ppt.PPTInterface
              InteropInputs   PowerPointInteropInputs.loadArgs.ShowFile="LogicPuzzleTutor\LogicPuzzleTutorial_AnneSinatra_032813_NameCondition.ppsm"
                Choice: PowerPointInteropInputs   PowerPointInteropInputs.loadArgs.ShowFile="LogicPuzzleTutor\LogicPuzzleTutorial_AnneSinatra_032813_NameCondition.p
```

Fig. 1. An example of a course loaded in GIFT's CAT. Each node represents a different course element, which has sub-nodes that can be expanded to add more details.



Course Name: Logic Puzzle Tutorial
Version:
c grid puzzles.
Survey Context: Logic Puzzle Study

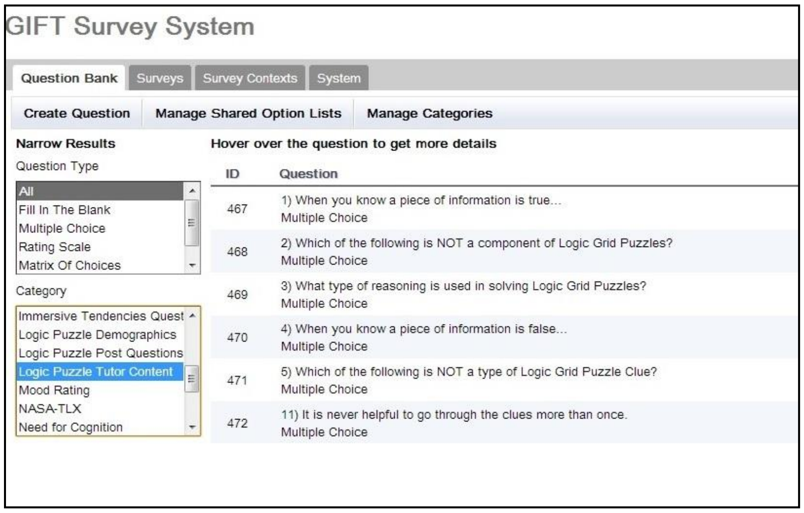
null
null
MIDSURVEY
null
LPCONTENTQ
null
APPLIEDLPQ

Fig. 2. An example of a course loaded in the GIFT's GAT. The transitions list indicates the elements of the course, and once selected, information can be entered for each element. Additional course elements are added using the menus.

If you have multiple conditions in your experiment, you can make multiple courses that are identical except for your manipulation of interest. For instance, if each condition receives a different PowerPoint, then you can create one course that opens a specific PowerPoint and a second identical course that opens a different PowerPoint. In order to do this, you can provide individual names for both of these courses, so that the system realizes that they are different, and to make it clear to yourself/research assistants which course needs to be opened in which condition. For instance, in a study that I conducted, I named the course for Condition 1: “1 – Logic Puzzle Tutorial”, and the course for Condition 2: “2 – Logic Puzzle Tutorial.”

Creating surveys

Surveys can be created using the Survey Authoring System (SAS), in the administration tab of the Module Monitor. The SAS allows the researcher to create individual questions, combine the questions into a survey, and then create a survey context. Each experiment or course has an individual survey context, which lists all of the surveys that will be used in that specific course. See Figure 3 for a screen-shot of the SAS.



ID	Question
467	1) When you know a piece of information is true... Multiple Choice
468	2) Which of the following is NOT a component of Logic Grid Puzzles? Multiple Choice
469	3) What type of reasoning is used in solving Logic Grid Puzzles? Multiple Choice
470	4) When you know a piece of information is false... Multiple Choice
471	5) Which of the following is NOT a type of Logic Grid Puzzle Clue? Multiple Choice
472	11) It is never helpful to go through the clues more than once. Multiple Choice

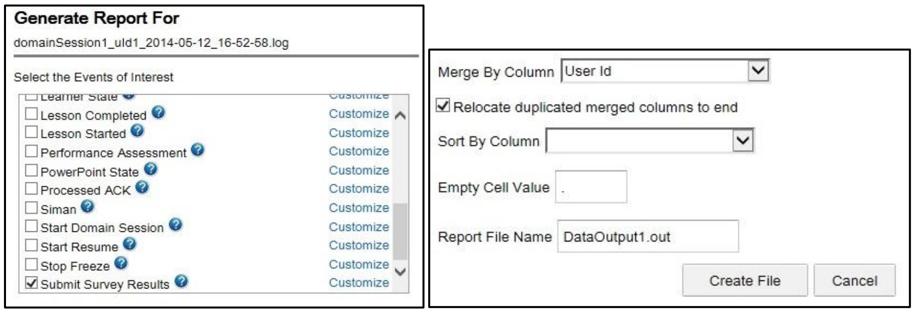
Fig. 3. An example of GIFT’s SAS. The tabs at the top of the screen allow the user to switch between creating questions, surveys, and survey contexts.

The SAS is a relatively straightforward and easy to use tool that provides the ability to create different types of questions such as fill in the blank, multiple choice, ratings scale, matrix of choices/matching, slider bar, essay, and true/false. After the questions have been created, they can be combined into full surveys. When a survey is created, the order of questions is determined and additional text/instructions can be added. Surveys can be created using questions that are already available in the SAS or can be created using questions entered by the researcher. When a survey is created, a “tag” (abbreviated name) can be provided for each question, if desired. This tag will be exported with the data and will be present at the top of data column associated with the specific question. Using a tag will make it easier to work with the data that is output after the experiment is completed.

After all of the desired surveys are created for an experiment, the next step is to create a survey context. The survey context is a list of all the surveys that will be used for a specific course. Each survey will be associated with a specific “key” name that is entered by the researcher. When a course is created using the CAT or GAT, it is necessary to set an associated survey context. When you are designing your course and add a survey element, it will provide with a list of all the “key” names that you established during the survey context creation. Only surveys in the associated context will be available for use. When you export your data, the “key” names are also present as part of the column titles in addition to the provided “tag” names for each question.

Exporting data

The ERT is used by the researcher to export the completed data. The researcher selects the desired data files, then can choose a merge type and the types of data he or she wishes to export. It is likely that for a traditional psychology experiment, the only necessary data will be the survey responses, which are called “Submit Survey Results” in the ERT. The ERT provides a number of different ways to merge data; however, merging by UserID is very useful for a researcher. This merge type provides an output file that has each participant represented by a row, and each survey question response is represented in a column. This output can be easily opened as an Excel file, then imported into SPSS for analysis. See Figure 4 for examples of the report generation and merge options in the ERT.



Generate Report For
domainSession1_uId1_2014-05-12_16-52-58.log

Select the Events of Interest

- [ ] Learner State — Customize
- [ ] Lesson Completed — Customize
- [ ] Lesson Started — Customize
- [ ] Performance Assessment — Customize
- [ ] PowerPoint State — Customize
- [ ] Processed ACK — Customize
- [ ] Siman — Customize
- [ ] Start Domain Session — Customize
- [ ] Start Resume — Customize
- [ ] Stop Freeze — Customize
- [✓] Submit Survey Results — Customize

Merge By Column: User Id
[✓] Relocate duplicated merged columns to end
Sort By Column:
Empty Cell Value: .
Report File Name: DataOutput1.out

Fig. 4. Left: Screenshot of the events of interest that can be selected in the ERT. Right: Screenshot of the merge selection and column sorting options in the ERT.

Tips for Running Experiments with GIFT and Future Directions

Assigning/managing participant numbers

In GIFT 4.0 and earlier, an important issue to be aware of is how to organize participant numbers. When you create a new participant, you are asked to enter an LMSName and the participant’s gender. Afterward this new participant is assigned a participant number, which is in sequence of the previous participants that were created on that specific instance of GIFT. When you log the individual into the computer, you will use this UserID number, rather than the LMSName that you entered. Additionally, when you export your data, it can be merged by UserID, but not LMSName. Therefore, it is important to determine a way to match the participant number that you have assigned to your participant to the appropriate UserID. The entered LMSName will not be available in relationship to the exported data. There are a few different solutions/workarounds for this, which include creating a demographics questionnaire in the SAS that asks the participant to enter their participant number that you gave them. Additionally, you can assign your participant numbers based on the actual UserID that is given from the GIFT system, rather than a self-entered number.

In post 4.0 versions of GIFT, starting with GIFT 2014-1X, the UserID and login process remains the same; however, the exported data merge will be based on the entered LMSName, rather than the UserID.

For example, if you would like to name your participant "P05," you would enter that as the LMSName when you register them, and if you merge the data by UserID when you export it, that is the name that will be associated with it.

Editing surveys

In GIFT 3.0, and GIFT 4.0, the SAS became more sophisticated. In addition to this, it also lost the ability for the researcher to be able to edit surveys after data has been stored in them. This means that if you create your GIFT course, run through it to make sure everything works, while either answering or leaving the survey questions blank, you will no longer be able to make any edits to the surveys themselves. One of the simpler workarounds for this issue is to use the "Test" function within each survey in the SAS. This gives you a preview of what the survey will look like and how you can interact with it without saving the data. Additionally, if you have already run through the surveys and need to edit them, you can do so by copying the surveys, making your changes, and updating/creating your survey context to reflect the appropriate surveys.

GIFT 2014-1X includes an option that allows you to delete any previous survey data that were stored, which would allow you to once again make edits to your surveys. This future feature should be used with caution; however, it should give researchers more flexibility in making edits to their surveys.

Exporting your data

In GIFT 4.0, when you merge your survey data by UserID (such that each participant's data are on one single line), duplicate entries are present. When the data are output in spreadsheet form there will be one column with the initial copy of the data (e.g., "PreTestMoodRating"), then there will be a column next to it with the same name and a "(2)" (e.g., "PreTestMoodRating (2)"). The second column is a duplicate of the first. To deal with this issue, the researcher can go into Excel or SPSS and delete every other column, to ensure that the duplicate data are not accidentally analyzed.

In GIFT 2014-1X, these duplications will still exist. However, once the researcher has exported the survey data using the ERT, there will be a popup indicating the existence of duplicate data. Additionally, rather than duplicate data being present in every other column, there will be an empty column at the end of the first copy of all the data. This empty column will then be followed by the duplicate data columns. This allows the researcher to easily identify the duplicate data and quickly delete them rather than having to do considerable edits to the entire outputted data file.

The Future of GIFT and Conclusions

GIFT is constantly evolving and user feedback can have a great impact on GIFT's future directions. The changes to the UserIDs, SAS, and ERT mentioned in the above tips section were a result of user feedback. One of the current goals of GIFT is to become more user-friendly. In GIFT's current form, it is a great tool for research psychologists. As GIFT continues to develop, it will gain new functionality, and through improvements in usability, it will be accessible to researchers of all skill levels. While at first glance, GIFT may seem complicated; after getting some experience with it, it can be a powerful and easy to use tool for conducting research. Hopefully this guide will assist you in creating your own psychology research experiments using GIFT, and when you approach a new challenge in GIFT, remember, “DON’T PANIC”.

Acknowledgements

Research was sponsored by the U.S. Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-12-2-0019. The views and conclusions contained in this paper are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

References

Adams, D. (1979). *The hitchhiker’s guide to the galaxy*. New York: Pocket Books.

Goldberg, B. & Cannon-Bowers, J. (2013, July). Experimentation with the Generalized Intelligent Framework for Tutoring (GIFT): A Testbed use Case. In *AIED 2013 Workshops Proceedings Volume 7* (p.27).

Sinatra, A.M. (2013, July). Using GIFT to Support an Empirical Study on the Impact of the Self-Reference Effect on Learning. In *AIED 2013 Workshops Proceedings Volume 7* (p.80).

Sottilare, R.A. & Holden, H.K. (2013, July). Motivations for a Generalized Intelligent Framework for Tutoring (GIFT) for Authoring, Instruction, and Analysis. In *AIED 2013 Workshops Proceedings Volume 7* (p.1).

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 and Engineering Directorate (ARL-HRED).*

Structure of Instructional Interactions Within GIFT

Bob Pokorny[1], Dustin Chertoff[1], Lisa Holt[1] , and Benjamin Goldberg[2]
[1] Intelligent Automation, Inc.
[2] U.S. Army Research Laboratory

Introduction

Simulation-based instruction can motivate students and more readily lead to the transfer of skills to real-world situations. This transfer can be increased when the simulation is paired with instruction based on principles of human learning. In this paper, we describe an instructional environment in which students receive performance feedback that follows good instructional principles. An assessment engine triggers instructional remediation based on 1) student violations of good performance policies and 2) the severity of the violation. In our approach, we present adaptive instruction to students while not overloading their cognitive resources by identifying the aspect of the student's performance that is most discrepant from good performance and guiding the student through an intervention that targets that topic. Thus, our system's design goal is to create instruction that is contextualized, adaptive to student violations, interactive, and not taxing on student cognitive load. This paper describes our approach to building this kind of instruction and how we implement it using the Generalized Intelligent Framework for Tutoring (GIFT).

Presenting instructional interactions within simulations using GIFT

Simulations on their own are a motivating and contextualized tool for helping students learn. However, the instructional power of simulations can be greatly increased if feedback is adapted to students' needs. Two important challenges we address are to identify 1) the topic(s) on which students need assistance and 2) the kind of instructional intervention that can be provided within a simulation. One of the primary challenges in developing instruction is knowing how and when to intervene when a student needs help.

In this paper, we describe an instructional system that presents instructional interactions within a simulation environment using GIFT. We provide the rationale for the instructional design, followed by a discussion of mechanisms within GIFT that support this instructional approach.

Instructional principles

The following research-based instructional principles can be used to develop effective instruction:

1. Instruction is embedded within a realistic representation of the targeted environment for using the content of the instruction. Most commonly, this will be a computer simulation of the real environment. These environments are instructionally beneficial because students can see how the content they are learning can be used, and students can use the learning environment to practice the skills required to perform well in the real-world target situation (Collins & Greeno, 2010).

2. Within the instructional environment, instructional interventions will target specific areas of student performance and knowledge based on an assessment of student performance. By following this principle, the instruction adapts to student needs (Woolf, 2009).

3. Over time, the instructional environment will present scenarios that change and adapt to what students need. Generally, simulation scenarios will become more complex. Additionally, if students need to practice specific skills, scenarios will be selected in which those skills are likely to be exercised (vanLehn, 2006).

4. The instruction will consist of more than telling students how to perform better—the instruction will engage students in an interaction with the content. Interactions with content support more performance improvement (Chi, 2009) so that students learn more about the area in which their performance needs to improve.

5. The instruction needs to consider cognitive load. The instructional interaction should not be so complex as to overwhelm the student's attention. This is particularly important for instruction within a simulation, as the students need sufficient cognitive attention to keep the simulation scenario in mind (DeJong, 2009).

6. The simulation-based instruction should provide opportunities for the students to apply what they need to learn in a variety of contexts. With simulations, this is most often accomplished through the use of different scenarios (Bransford, et. al, 2000).

7. The instructional environment should foster positive student affect. Students should feel good about their learning and their progress (Graesser, et. al. 2006).

Micro-Strategy Pedagogy

While the seven principles just described are all important to instruction, our focus is on micro-strategies within a lesson rather than macro-strategies involving sequencing of scenarios and pacing. Thus the two principles we address are 1) instructional interactions and 2) cognitive overload. Some of the interactions students can have with automated instructional systems include hints, feedback, challenges, articulations, reflections, explanations, and scaffolding. Cognitive overload can be reduced by not attempting to address all areas in which students should improve at once. Simulations heighten the need to consider cognitive overload because the thread of the scenario must be kept in mind by the student.

When the balance between inducing cognitive processing and cognitive limitations is appropriate, simulations can be a superb learning environment. In a recent study investigating the effectiveness of embedding instruction within a simulation, an additional hour of instruction distributed across a 12-day class led to the students in the instructional treatment condition performing 24% better than students who used the simulation without instruction (Pokorny, Haynes, Gott, Chi & Hegarty, 2013). The instruction was crafted to both induce cognitive processing of the domain knowledge (in this case, system function and maintenance procedures) that students needed to know while not interfering excessively with the flow of the scenario.

The context for this study was a procedural maintenance task in which students practiced conducting procedures within a simulation. The procedural task was occasionally interrupted when students performing the procedure reached the end of a sub-goal. At this point, the students were asked a question which targeted the primary function of the system examined by the last set of procedural steps. Examples of primary function questions included a) How was the system configuration changed by the last few steps? b) What was the rationale of the test being conducted? and c) What interpretations should be made if a specific reading of a test was obtained? The instructional sequence is shown below:

- Students are asked a question
- Students answer the question
- Students are given an expert's answer to the question
- Students compare their answer to the expert's answer on a few key features.

We are studying the effectiveness of this kind of instructional intervention in a training environment that focuses less on procedural knowledge and more on decision making. In a turn-based game, we are creating instruction that mirrors the pattern described above. The decision-making instructional environment requires students to do more than follow a procedure; students have to select choices. The rest of this paper explains the instructional design we are implementing, and how we are using GIFT to implement the pattern.

instructional Design

The instructional design balances the instructional need to induce cognitive processing while limiting cognitive overload. The instructional context is a turn-based game in which students make many decisions per turn. An assessment system identifies student decisions that differ most from experts' decision-making policies. To help the students make decisions that are more like experts, instruction is provided on the aspect of performance identified as needing attention.

Three instructional flows

A model of the instructional flow which is similar to the pattern described above for the procedural task is shown in Figure 17. This model of instruction assumes a turn-based instructional game. The same basic mechanism would be applied to continuous simulations, though the instruction is easier to describe with turn-based remediations.

The most significant difference between the current study (turn-based, decision-making game) and the earlier study (procedural maintenance simulation) is that the turn-based design enables scaffolding. On the first turn in which a topic is identified for remediation, the instruction consists of a relatively simple question to the student. This question might lead to better student understanding and performance. If it does not, a more focused question is asked of the student.

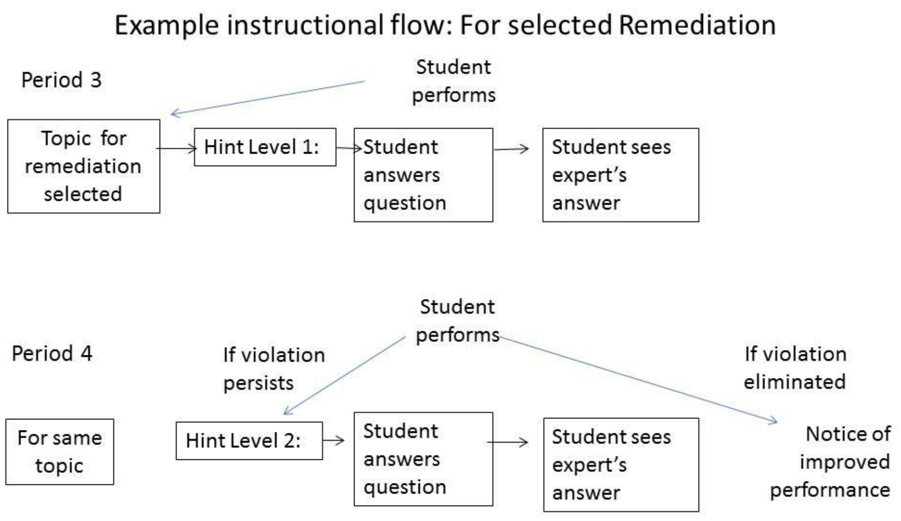


Figure 17. Instructional flow; two levels, with response to coaching

We are investigating the above instructional flow, as well as two others. It is possible that the intervention presented in Figure 17 will yield excellent results. It is also possible that the this intervention is too demanding for students; students might be cognitively overloaded, not improve much, and complain of losing the context of the simulation scenario. If this occurs, we could make the instructional interventions less complex, by removing the need for students to directly respond to the questions as shown in Figure 18.

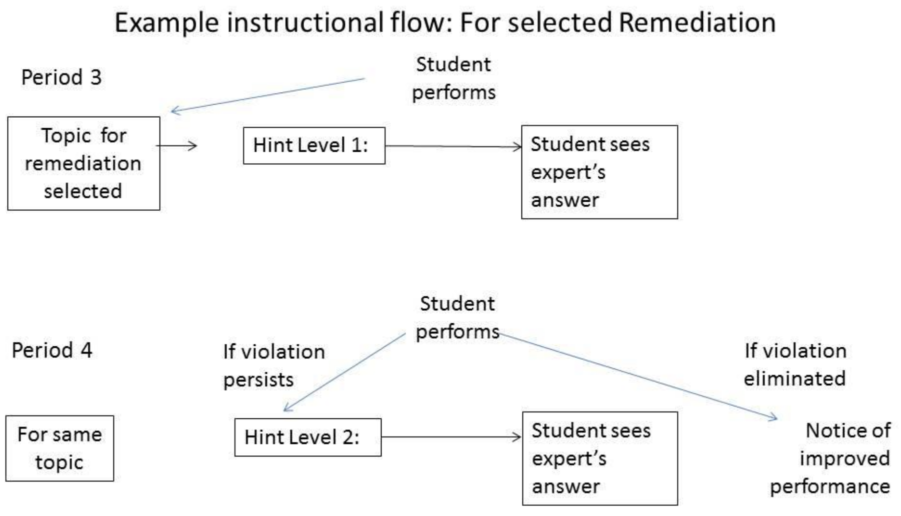

Example instructional flow: For selected Remediation
Period 3
Student performs
Topic for remediation selected
Hint Level 1:
Student sees expert's answer
Period 4
Student performs
If violation persists
If violation eliminated
For same topic
Hint Level 2:
Student sees expert's answer
Notice of improved performance

Figure 18. Instructional Interactions; two levels; no response to coaching

Another possible result of the experiments is that the instructional remediation designed to increase student knowledge and performance is not powerful enough, and neither students' knowledge nor performance will improve. If this is the case, we would move the instructional balance more toward inducing cognitive processing by adding another level of hints. This instructional flow is depicted in Figure 19.

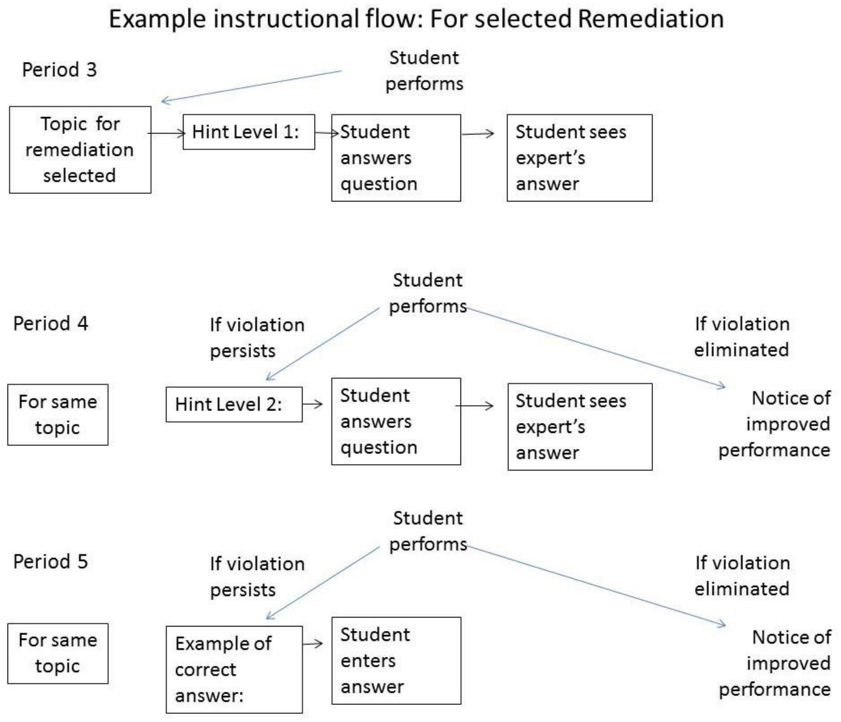


Figure 19. Instructional Interaction; three levels, with response to coaching

Instructional content

In addition to the instructional design flow, we also discuss the content of the instructional remediations. The domain of our example instruction is counter-insurgency and the training environment is UrbanSim, a counter-insurgency game developed by the University of Southern California, Institute of Creative Technologies.

The assessment system we are building evaluates and scores each decision that students make. The worst decisions that students make will become targets for instruction. This assessment approach is based on expert evaluations of student performance. For more details, see Pokorny, Haynes, and Gott (2010). In this paper, we describe the instructional interactions that students should have (as determined by our assessment system) and how they can be implemented in GIFT.

As an example, a student playing UrbanSim may need remediation on the topic of "applying excessive force." Using the instructional flow of Figure 17, the first question shown to the student could be the following:

- Are your security actions consistent with doctrine?

After the student enters an answer, the instructional system presents an expert's answer to the question:

- While security is a priority of your Commander’s intent, you also have to address other concerns.

If the student’s next action improves, i.e., the student does not use too much force, the instructional system responds with a message indicating compliance to good performance:

- Your security actions showed a more even-handed approach to balancing security concerns and other needs of your mission.

If the student continues to use too much force, the next hint would be more specific:

- As tribal zones transition from Clear to Hold, are you ordering excessive security actions given the current security risks?

After the student enters an answer, the instructional system follows up with a message that provides specifics about the student’s actions with respect to desired performance:

- Your security risks were the continued existence and activity of the Shia Death Squads.

- Your security actions were a patrol in the Market District, seizing a residence in the Merkel District, joint investigations of IEDs, and vehicle checkpoint near the cement factory.

- Experienced commanders, applying Clear, Hold, and Build, would use fewer security actions

The pattern of instruction uses the first hint to direct the student’s attention to the major category of violation. After the student responds, the instructional system gives a simple recommendation about what category of performance the student should focus on.

For the second level of hints, the question attempts to focus the student on the specifics of the violation. After the student responds, the instructional system provides feedback about what the student should do to improve, using specific examples from the student’s action.

Integrating the Assessment Engine with GIFT

The first step towards integrating our assessment engine with GIFT is to establish a method for sending student violation scores to GIFT. To achieve this, we wrote a custom Gateway module plugin. This plugin opens a socket connection between the assessment engine and GIFT, allowing for scores to be sent to GIFT after each UrbanSim turn. Scores are sent as binary information that is converted by the Gateway plugin into a GameState class. Once all data about a turn have been received by the plugin, the current GameState is sent to the other GIFT modules using the sendMessageToGIFT function.

A series of Conditions for each of the scores were also implemented and placed in the Domain module. Each of these Conditions processes the GameState data in the message sent by the Gateway plugin in order to determine whether a score is above, at, or below expectation. The result of this expectation assessment is returned and eventually sent to the GIFT Pedagogy module for instructional feedback selection. This process is illustrated by Figure 20.

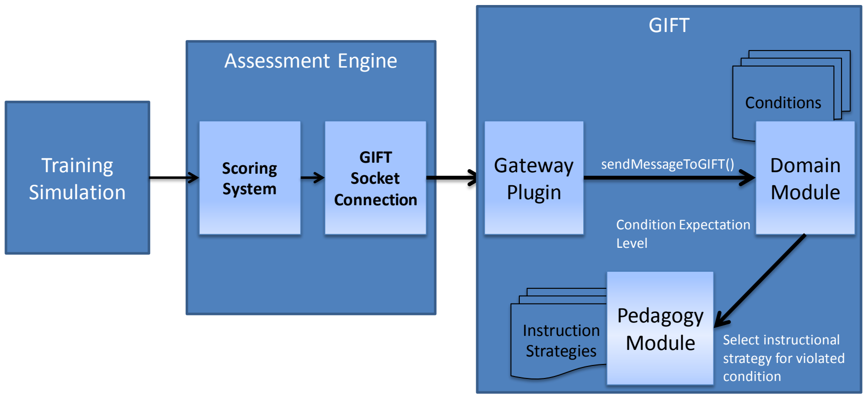


Figure 20. Flow of data from a training simulation through GIFT

Potential instructional feedback is entered into the domain knowledge file (DKF) associated with our assessment engine. GIFT automatically handles the selection of which level of feedback is needed for a given score. So, the first time a Condition is found to be below expectation, the first level of feedback is provided. If that same Condition is again found to be below expectation, the second level of feedback is provided. Thus, integrating our instructional interventions simply requires that we specify a series of tactics linked to strategy calls in the DKF.

For example, the instructional intervention for excessive force outlined previously could be implemented in GIFT through the following strategy/tactic pairing:

```
<strategy name="Excessive Security Strategy">
 <p0:instructionalIntervention>
 <p0:strategyHandler>
 <p0:impl>domain.knowledge.strategy.DefaultStrategyHandler</p0:impl>
 </p0:strategyHandler>
 <p0:feedback>
 <p0:message>
 <p0:content>Are your security actions consistent with doctrine?</p0:content>
 </p0:message>
 </p0:feedback>
 <p0:feedback>
 <p0:message>
 <p0:content>As tribal zones transition from Clear to Hold, are you ordering
excessive security actions given the current security risks?</p0:content>
 </p0:message>
 </p0:feedback>
 </p0:instructionalIntervention>
</strategy>
```

Gift Extensions

We are making two extensions to GIFT to support student learning without inducing cognitive overload: providing remediation for the worst performing concept and integrating context relevant feedback into the remediation messages.

Each GameState message sent to the Domain module contains scores for many concepts. Our strategy to reduce cognitive load is to focus on the concept that stands to gain the most improvement. As such, we only provide remediation for the concept on which the student has performed worst relative to all other concepts. Essentially, this is a prioritization problem. Thus, we need a way to dynamically prioritize concept scores as the Domain module receives them.

While GIFT does include a method to set concept priority, by default it is intended for *a priori* use. Therefore, we have expanded the GIFT concept priority methods within the Domain module to support the dynamic adjustment of concept priority as the Domain module receives new concept scores.

Next, for students who need multiple levels of remediation, it is helpful to provide specific context relevant information to help them perform better. For example, students who are performing infrastructure improvements in the UrbanSim game should receive feedback such as “You repaired the school first, but it is more important to repair the concrete plant and gas station first.” Including specific student actions in the remediation helps students to target the cognitive resources conceptually important to improving their performance.

Supporting this in GIFT requires a multi-stage approach. First, contextual information must be sent from the external simulation to GIFT. In our case, we bundle the contextual information with the score messages. As the information is sent to the Domain module, it is stored for later use with a custom StrategyHandler. Next, within the DKF we support the use of wildcard symbols that can be replaced with stored contextual information.

```
 <p0:content>Your repaired the * first, but it is more important to repair the
concrete plant and gas station first.</p0:content>
```

Conclusion

Our approach demonstrates one method for using GIFT to implement instructional interactions for students using a training simulation. The approach we took is especially beneficial if the student performance assessment engine already exists or needs to perform scoring logic that would be quite complicated to implement in a GIFT DKF. Nevertheless, even when scoring calculations are performed outside of GIFT, it is still possible to use the other GIFT modules. In particular, the Pedagogy module can be used to handle the selection and presentation of instructional content, which can be further enhanced by GIFT’s Learner and Sensor modules.

References

Bransford, J., Brown, A. & Cocking, R. (2000). *How People Learn: Brain, Mind, Experience, and School.* Washington, D.C. : National Academy Press.

Chi, M. (2009). Active-Constructive-Interactive: A Conceptual Framework for Differentiating Learning Activities. *Topics in Cognitive Science*, 73-105.

Collins, A. & Greeno, J. (2010). A Situative View of Learning. In *International Encyclopedia of Education.* London: Elsevier.

De Jong, T. (2009). Cognitive Load Theory, Educational Research, and instructional Design: Some Food for Thought. *Instructionmal Scneice*, 105-134.

Greasser, A., McDaniel, B., Chipman, P., Witherspoon, A., D'Mello, S. & Gholson, B. (2006). Detection of Emotions During Learning with AutoTutor. *Proceedings of the 28th Annual Meetings of the Cognitive Science Society.* Mahwah, N.J.: Erlbaum.

Pokorny, R., Haynes, J., Gott, S. (2010). Performance Assessment in Complex Simulations. In Proceedings from I/ITSEC, Orlando, Fl. *Proceedings from Interservice/Interagency Training, Simulation and Education Conference.* Olrando, FL: NTSA.

Pokorny, R., Haynes, J., Gott, S., Chi, M. & Hegarty, M. (2013). Infusing Simulations with Expert Mental Models, Adaptivity, and Engagining Instructional Interactions. *Proceedings from Interservice/Interagency Training, Simulation and Education Conference.* Olrando, FL: NTSA.

vanLehn, K. (2006). The Behavior of Tutoring Systems. *Internation Journal of Artificial Intelligence in Education*, 227-265.

Woolf, B. (2009). *Building Intelligent, Interactive Tutors" Student-Centered Strategies for Revolutionizing e-Learning.* Burlington, MA: Morgan Kaufmann.

Theme IV: Affect and other Non-Cognitive Factors in Intelligent Tutoring Systems

Investigating the Role of Physiological Measurement in Intelligent Tutoring

Jennifer S. Murphy, Meredith B. Carroll, Roberto K. Champney, Christina K. Padron
Design Interactive, Inc.

Introduction

Traditionally, student models in intelligent tutoring systems (ITSs) are based on the student's knowledge of the content area. Recently, there has been increasing interest in additionally assessing student cognitive and affective states as the basis for training adaptation. Several research challenges exist in realizing this goal. We have developed a model for adapting training using physiological measurement based on Vygotsky's (1934) model of the zone of proximal development (ZPD). This approach not only provides the ability to assess the student's workload level without disrupting the training process, but it also enables the prediction of when the student is about to fall out of the optimal training state. This paper summarizes work done to date in support of the U.S. Army Research Laboratory's (ARL) Generalized Intelligent Framework for Tutoring (GIFT) and related efforts to further the science of integrating cognitive and affective physiological measurement into ITSs, and describes potential ways forward.

The U.S. Department of Defense has expressed an increasing interest in incorporating learning technology in training as a means of providing a flexible, engaging, and cost-effective means of maintaining Warfighter readiness in an ever-evolving operational environment. For example, the Army Learning Model as described in *The Army Learning Concept for 2015* (Department of Army, 2011) calls for the inclusion of adaptive training technologies in Soldier education as part of developing a persistent culture of learning within the Army. Toward achieving this goal, there has been a recent resurgence in the popularity of ITSs as a means of adapting instruction. Initiatives such as ARL's GIFT have provided a test bed for research into addressing some of the challenges associated with incorporating intelligent tutoring into military settings, including reusability of content, cross-platform training, and content authoring.

One area of research interest, both within the GIFT user community as well as in the intelligent tutoring field more broadly, is the development of multi-faceted student models. The student model (or learner model) refers to the intelligent tutor's representation of the learner's internal state. Traditionally, the student model reflects the status of the student's knowledge of the domain area, but recently, research efforts have focused on how to incorporate additional aspects of the learner into the student model (Holden et al. 2012). Stable, or "trait-based" aspects of the learner include a student's information processing capability, attentional capacity, motivation, and emotional regulation ability (Gully & Chen, 2010). In addition, more fluid aspects such as cognitive and emotional state have shown to play a role in student learning (see Carroll et al., 2011, for a review). By incorporating this additional information into a student model, the ITS can adapt the learning experience not only based on how much the student has learned, but also how much the student is capable of learning.

While research in this area is promising, it is far from complete. Two research questions involved in developing complex student models are first, how to obtain them, and second, what to do with them once we have them. The first question speaks to identifying both the critical learner states that should be assessed and integrated into a student model and the technology required to do so. The second speaks to the development of advanced instructional models that can incorporate information about both students' performance but also their internal state into ITS. In this paper, research conducted to address these questions is described, with the goal of providing a framework for the inclusion of multi-faceted student models in GIFT. First, research to evaluate low-cost sensor technology for the assessment of learner cognitive and affective state is reported. Second, the concept for an instructional framework incorporating a student's cognitive state based on Vygotsky's (Vygotsky, 1978) model of the ZPD is described. The goal of this paper is to inform the development of multi-faceted student models within GIFT and ITSs more generally.

Assessing Learner Internal States

In order to tailor training to a learner's internal state, measures of this state must first be implemented. Stable learner characteristics such as intelligence, personality, and motivation, while not typically included in student models, could be assessed prior to training through existing, validated questionnaires (Holden et al. 2012). In terms of inclusion in GIFT or other military ITSs, a major barrier to their inclusion is cost, as many measures of constructs of interest are proprietary. However, the Department of Defense has conducted considerable research into developing their own measures of personality constructs of interest, and these could be leveraged for inclusion in student models. Examples of these include the Army's Tailored Adaptive Personality Assessment System (Drasgow et al., 2012) and the Navy Computer Adaptive Personality Scales (Schultz et al, 2011).

Assessing more fluid cognitive and emotional states is considerably more complex because these states have the potential to fluctuate over the course of training. To account for this variability, real-time assessment of these states is ideal. Cognitive and emotional states have been assessed in ITS through self-report questionnaire (Sottilare and Proctor, 2012), inferred through behavior (D'Mello and Graesser, 2007), and assessed physiologically (Cooper et al., 2009) although each of these methods leaves something to be desired. Self-report measures are inherently subjective and are therefore prone to effects of positive self-presentation and other biases. More importantly, administering them with the goal of real-time assessment requires interrupting the flow of training, which is particularly disruptive to scenario-based training. Assessments of cognitive and emotional state through behavior such as facial expressions and vocalizations is promising (D'Mello and Graesser, 2007); however, these measures often require the participants to speak, which may disrupt the flow of training. The primary barriers to leveraging physiological measurement have historically been the expense of sensors and the expertise required to implement them; however, recent advances in sensor technology have reduced the expected costs associated with implementing them. While these new sensors are relatively inexpensive, their accuracy with regard to enabling classification of emotional and cognitive state is not clear.

In support of the GIFT initiative, a suite of low-cost sensors was evaluated to determine whether they could reliably predict targeted cognitive and affective states (Kokini et al., 2012). Specifically, the goal of this research was to determine if correlations exist between low-cost sensor metrics and associated ground-truth measures of targeted cognitive and affective states in a training context, and determine if such low-cost sensors could accurately and reliably measure distinct cognitive and affective states. The methodology and data analysis techniques used in this research are described in detail elsewhere (Kokini et al., 2012). This paper provides a brief overview of the research, but focuses on the issues involved with implementing low-cost sensors to assess cognitive and affective state, and then discusses the potential uses for the findings.

Two experiments were completed to compare low-cost physiological sensor output to validated bench-mark measures of affective and cognitive states while participants performed a number of tasks designed to elicit these states. These tasks included a visual vigilance task, video clip observation, and videogame-based training scenarios. Both experiments involved the same experimental sensor suite, target cognitive and emotional states, validated measures of these states, and performance tasks. In the interest of brevity, the experimental procedure is described once. They differed in participant population (25 civilians in Experiment 1; 20 U.S. Military Academy (USMA) cadets in Experiment 2) and order of task performance, but these differences were unrelated to results. The findings of the first experiment did inform the second experiment, but this was more in terms of how the low-cost sensor data were collected, and is discussed later.

Experimental protocol

Methodology. A review of the literature identified three target affective states (anger, fear, and boredom) and three cognitive states (engagement, distraction, and workload) for evaluation based on 1) their impact on learning, and 2) their potential to be measured with inexpensive sensor technology. Each of these states had previously been correlated with physiological data (Lisetti and Nasoz, 2004; Woolf et al., 2010). In addition, each of these states was shown to impact learning performance.

To measure these states, a suite of low-cost, minimally intrusive sensors was selected from a market survey of sensors completed during this effort. Final requirements for each of the selected sensors were 1) candidate sensor can measure at least one target state, 2) cost less than $500, 3) good access to sensor data stream, 4) low level of intrusiveness, and 5) preferably commercial off the shelf (COTS). The sensors selected are listed in Table 1.

Table 1. Low-cost sensors for measuring affective and cognitive states

Type	Product Name	Associated State
EEG	NeuroskyMindSet	Distraction, engagement, workload
Heart Rate	Zephyr HxM Developer Kit	Anger/frustration, fear/anxiety, boredom, engagement
Eye Tracking	Developed in-house	Distraction, workload
Motion Detection	VernierGo!Motion	Anger/frustration, boredom, engagement
Chair Pressure	Trossen Robotics 4-6 1.5 inch Force Sensor Kits; Phidgets Control Board	Anger/frustration, boredom, engagement

For purposes of comparison, a number of "ground truth" measures of affect and cognitive state were identified. These included EmoPro™, an electronic emotional profiling tool validated to accurately measure emotions (Champney and Stanney, 2007) and ABM's B-Alert™ X-10 EEG headset. The associated B-Alert™ analysis software includes electroencephalography (EEG) indices of workload, engagement, distraction, and drowsiness.

To conduct the comparison, participants performed three tasks while wearing both the low-cost sensor suite and the EEG headset. At specified intervals, participants completed an EmoPro™ assessment to report their emotional state. The tasks included a visual vigilance task in which the user had to press the space key every two seconds with a visual reminder, observation of movie clips validated to induce affective states including anger, fear, and a neutral state; and completing videogame-based training scenarios in Virtual Battlespace 2 (VBS2). In these scenarios, the overall goal was to search and eliminate enemy threats in an urban environment (e.g., a building or street).

Procedure. After participants were briefed and consent was received, they donned the ABM headset and completed an EEG baseline task. They then donned the Zephyr heart rate sensor around their chest and the NeuroSky EEG headset was placed on their head. The participants sat in the pressure sensor chair in front of a display, motion detector, and eye tracker, and completed a calibration session with the eye tracking system. Adjustments were made to all sensors as needed until continuous data collection was attained. Once all sensors were in place and successfully collecting data, participants performed the series of tasks outlined above. First, participants performed a three-minute vigilance task on a personal computer, which consisted of pressing the space bar every time a red circle appeared on the screen. Participants completed an EmoPro™ evaluation just before and just after this task. Next, participants observed three video clips, completing an EmoPro™ evaluation just after each video clip. Next, experimenters described the VBS2 task in detail and had participants go through training to familiarize them with how to interface with the software. Participants were then asked to complete a trial scenario to gain an understanding of what would be expected of them during the experimental task. Next, participants completed a total of four scenarios, with 3-6 critical events per scenario. Following each critical event within a scenario, participants were prompted to complete an EmoPro™ evaluation to indicate their emotional state during the event. Upon completion of the VBS2 scenarios, participants received a short debriefing.

Results. For a more detailed discussion of data analysis techniques, see Kokini et al. (2012). Findings from the first experiment were largely inconclusive. First, both the eye-tracking and chair pressure data were found to be unreliable and were excluded from further analysis. To calculate classifier strength, three runs of a 10-fold cross-validation procedure were executed for each model, and their corresponding receiver operating characteristic (ROC) curves were plotted. The overall quality of each classifier was assessed by averaging the areas under its ROC curves (AUC values; Fawcett, 2006). As a rule of thumb, excellent classifiers are those having AUC values between 0.9 and 1. Classifiers with AUC values from 0.8 to 0.9 are typically considered good, and those having AUC values from 0.7 to 0.8 are considered fair. Each classifier evaluated in this research showed an AUC value of about 0.5, which is very weak. While other analyses showed significant effects, the lack of reliable classifiers called into question all other findings.

The inconclusive results in the first experiment were explainable by complications with the low-cost sensor hardware. Sensor-related issues identified included the following:

- The eye tracker was unreliable in tracking pupilometry due to the movements of the participants and changes in ambient lighting.
- The chest strap of the heart rate monitor did not have good conductance due to a lack of moisture.
- The chair sensors showed low variability, due to a tendency to detach from their original locations.
- Administering the EmoPro^{TM} required a break in task performance, breaking immersion in the scenarios.

To address these issues, methodological changes were made and incorporated into Experiment 2. A new eye tracker was developed to address the unreliability found in the original apparatus. The pads on the heart rate monitor were moistened prior to application to ensure good conductance. In addition, small changes to the scenarios were made to enhance emotional response to account for the influences of EmoPro^{TM} administration time.

As a result of these changes, in Experiment 2, good classification models based on data from the low-cost sensor suite were developed for the affective states of fear and boredom, and the cognitive states of distraction, engagement, and workload. The methodology used to induce identified states proved successful for five of the six states, as a significant difference in presence/absence or high/low levels of states were identified using the "gold standard" metrics. The one exception was anger, where there was not a clear distinction between high/low experiences of this state using the methodology outlined. Of the low-cost sensors utilized in this effort, it was found that low-cost EEG and motion sensors were effective in capturing the majority of affective states, while heart rate, chair pressure, and motion sensors were effective in capturing cognitive states. Using this combination of sensors, the set of five states can be captured unobtrusively for approximately $600. For more detail on how these classifiers were calculated, see (Kokini et al. (2012).

Lessons Learned. The findings of this research are promising; however, the research team encountered considerable issues with regard to implementing the low-cost sensors. The most substantial challenge was the noise inherent in the sensor data. Even after identifying a new eye-tracking solution in Experiment 2, 15% of these data were removed due to noise. While this was a vast improvement, losing 15% of data in any situation is not ideal. The chair sensor, while a significant contributor to classifiers of the cognitive states (engagement, distraction, and workload), was consistently noisy, perhaps due to the amount of padding between the chair and the participant. Finally, single channel EEG data have a high potential of contamination from electromyographic (EMG) or electro-oculographic (EOG) activity, which may have added noise.

This research suggests one could indeed leverage low-cost sensors for real-time classification of cognitive and emotional states; however, another major takeaway is that you get what you pay for. Sensor technology has become less prohibitively expensive in recent years, and popular fitness tracking trends and interest in the “quantified self” will push low-cost sensors forward toward a price point accessible to the general public. Assessments based on these sensors may be sufficient for utilization in training, but further research must be conducted to verify that this is the case.

Adapting Training Based on Learner States

The findings from the two experiments described here imply that affective and cognitive state can be assessed using low-cost sensors and potentially integrated into multi-faceted student models. The question remains: how should the data from these advanced student models be leveraged to enhance student learning? That is to say, if a student model incorporated data about a student’s affect or cognitive state, how should training content and feedback be adjusted to maximize the learning potential of the student? For example, if a student shows signs of being bored, perhaps the ITS should increase the difficulty of the training to maintain the student’s interest by increasing the complexity of problems to solve. Similarly, if a student is experiencing a high cognitive workload, the ITS may decrease the training difficulty by slowing the pace of instruction. While ongoing research focuses on developing complex student models, guidelines for adjusting the ITS instructional model based on them have yet to be developed.

The zone of proximal development

The benefit of incorporating cognitive and emotional states into student models is the potential to adapt training content and feedback on not only the student’s knowledge, but also on their readiness to learn. One way of conceptualizing student readiness is through Vygotsky’s concept of the ZPD (Vygotsky, 1978). According to Vygotsky, the ZPD reflects the difference between what a learner can perform on his or her own and what is not possible for the learner without assistance from an instructor. This reflects the ideal learner’s state between being overwhelmed by difficulty and bored by a lack of challenge. If the student is too challenged, scaffolding on the part of the instructor alleviates the difficulty. On the other hand, if the student finds the content too easy, fading removes this scaffolding. The ZPD concept is depicted in Figure 1.

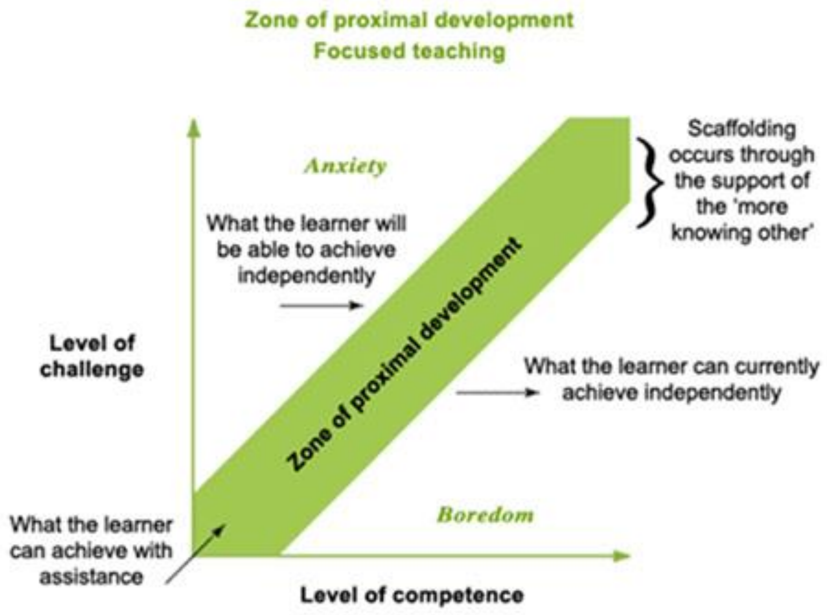

Zone of proximal development
Focused teaching
Anxiety
What the learner will be able to achieve independently
Scaffolding occurs through the support of the 'more knowing other'
Level of challenge
Zone of proximal development
What the learner can currently achieve independently
Boredom
What the learner can achieve with assistance
Level of competence

Fig. 1. Zone of Proximal Development

Conceptualizing the ZPD for training has been discussed by several authors, largely in the context of performance-based student models. For example, Murray and Arroyo (2002) propose using a mastery learning criterion to determine when the student is ready to move on and using the ZPD measurement to determine the efficiency of student learning. First, the number of hints (i.e., assistance) needed to correctly answer an item are recorded, and the sequence of this metric across the problem set is analyzed to understand the student learning model. Their proposed ZPD method then has three parameters: 1) a goal number of hints for each problem set (H), 2) the allowed variation in the goal number of hints to consider the ZPD (DH), and 3) the minimum number of problems the student will see (P). These authors instantiate their ZPD approach in their ITS for elementary school mathematics, *AnimalWatch* [19] In this research, a student's ZPD was defined by performance; specifically, if a student made on average one mistake per problem, he or she was considered to be within the ZPD. It is not clear why the authors chose this particular value; however, it does follow the general idea that if the problems were too easy, no mistakes would be made.

Another approach to assessing ZPD could involve integration of real-time sensor data to represent cognitive and affective states such as boredom, frustration, or engagement. While Murray and Arroyo's (2002) methodology makes assumptions about cognitive and affective state based on performance, integrating the physiological classifiers allows for a more granular diagnostic approach, ensuring that the instructional strategies are most effectively and efficiently inserted into the content. The findings of the research described here suggest that classifiers of states such as these can be developed and validated using low-cost, non-intrusive physiological sensors. Ideally, the data from these sensors could be combined with training performance data to define a student model based on what a student has learned and what the student has the *potential* to learn. Table 2 describes potential instructional interventions based on cognitive state.

Table 2. Instructional Interventions by Cognitive State

ZPD Status	Boredom	Workload	Intervention
Unable w/o Assistance	Low	High	Scaffolding
Able w/Assistance (ZPD)	High	Low	Scaffolding & increase challenge
	Low	High	Scaffolding & decrease challenge
Able w/o Assistance	High	Low	Increase challenge & fade scaffolding

Consider an ITS that adapts based on student performance. If the student is challenged but not overwhelmed, the addition of an assessment of cognitive state would not prescribe a change in instructional strategy. However, if the student demonstrates domain knowledge but is under high workload, it may be an indication that he or she is about to fall out of ZPD into "unable with assistance." In this instance, scaffolding may not enable the student to maintain ZPD; rather a decrease in complexity may also be needed to reduce the student's cognitive workload. Alternatively, if the student is performing well but is experiencing high boredom, it may be an early indication that he or she is about to fall into the "able without assistance" status, meaning that challenge should be increased. In addition, fading could be used to increase the student's challenge.

Under an effort performed for the Office of Naval Research, a framework for adapting training based on maintaining a student within the ZPD was conceptualized Carroll et al., 2013). This framework leveraged a combination of performance data with measures of trainee state to predict when a student is on the verge of leaving ZPD. This is achieved by increasing difficulty until performance starts to decrease, at which point scaffolding is applied. The framework adjusts both training challenge and scaffolding in real time to maximize performance; however, it does not specify the specific cognitive and emotional states that would be incorporated or how they would be weighted relative to performance criterion.

The science of adapting training based on emotional state within the context of ITS is relatively new. Some remaining foundational research questions are described below:

- *Do we adapt the training content or the context?* If a student is failing and measure of cognitive state implies that the student's workload is high, for example, is the appropriate intervention an adjustment of training content (e.g., reducing difficulty) or adjusting the training environment? While research has been conducted in both these areas, findings have not been operationalized in fielded systems (Sottilare et al., 2013). Although reducing problem difficulty should in turn reduce workload, it may be the case that the problem is sufficiently difficult, but aspects of the training environment are preventing learning. Unfamiliarity with training technology, excessive ambient noise, and training in group settings may cause increased workload. These aspects are also characteristic of scenario-based training often implemented by the military, and currently utilized within GIFT.

- *What instructional strategies should be employed?* The ZPD model described above calls for scaffolding, but instructional interventions span the range of adaptive spacing to hints to metacognitive prompting (see Durlach and Ray (2011)). Which interventions are most effective for alleviating negative states has yet to be researched.

- *Which cognitive and emotional states are most relevant?* The ZPD model is currently conceptualized in terms of challenge and competence, which suggests states such as workload, fatigue, boredom, and anxiety may be the most relevant for inclusion in a multi-faceted student model. However, other states may be just as relevant to success in ITS-based training, particularly within a military context. These states may include stress, trust, or sadness. Frustration in particular has been researched in the context of intelligent tutoring (Arroyo et al., 2003).

- *How should these states be weighted with regard to domain knowledge?* If cognitive and affective states are integrated into a student model, to what extent should these states be given importance relative to a student's understanding of the training domain? It would stand to reason that a multi-faceted student model would involve an amalgamation of both factors, but how that should be computed is not clear, and is probably domain dependent. In addition, training platform should be represented in this calculation, as the same content presented in a videogame-based scenario may be more inherently cognitively demanding (and thus result in more variability along this dimension) than a text-based presentation.

These questions, and others, will doubtlessly be the focus of research going forward. This is an exciting time to be involved in ITS research and implementation, as current technology is enabling training personalization in new, more comprehensive ways. However, clearly more research is needed to inform how these new capabilities, including the incorporation of cognitive and affective state, should be implemented to maximize the utility of ITS.

References

Arroyo, I., Beck, J.E., Beal, C.R. & Woolf, B.P. (2003). Learning within the ZPD with the AnimalWatch intelligent tutoring system. Paper submitted to AERA, Chicago, IL.

C.L. Lisetti, C.L & Nasoz , F.(2004). Using noninvasive wearable computers to recognize human emotions from physiological signals. EURASIP Journal on Applied Signal Processing 2004 (11), 1672-1687.

Carroll, M., Kokini, C., Champney, R., Fuchs, S., Sottilare, R. & Goldberg, B. (2011). Modeling trainee affective and cognitive state using low cost sensors. Paper presented at the presented at the Interservice/Industry Training, Simulation and Education Conference (I/ITSEC), Orlando, FL.

Carroll, M., Kokini, C., Ramirez-Padron, R., Surpris, G., Champney, R., Reni, J. & Hale, K. (2013) Multi-axis Assessment for Diagnosticity Enhancement in Individualized Training (MADE-IT). Final Report.

Champney, R.K. &Stanney, K.M. (2007).Using emotions in usability. Proceedings of the Human Factors and Ergonomics Society 51st Annual Meeting. Human Factors and Ergonomics Society: Santa Monica, CA, 1044-1049.

Cooper, D.G., Arroyo, I., Woolf, B.P. Muldner, K., Burleson, W.,& Christopherson, R. (2009), Sensors Model Student Self Concept in the Classroom, International Conference on User Modeling and Adaptive Presentation, Trento, Italy.

D'Mello, S. & Graesser, A. (2007). Mind and Body: Dialogue and Posture for Affect Detection in Learning Environments. In R. Luckin, K. Koedinger & J. Greer (Eds.), Proceedings of the 13th International Conference on Artificial Intelligence in Education (AIED 2007) (pp 161-168). Amsterdam, The Netherlands: IOS Press.

Dept. of the Army, (2011). The U.S. Army learning concept for 2015. TRADOC PAM525-8-2. Retrieved from http://www-tradoc.army.mil/tpubs/pams/tp525-8-2.pdf

Drasgow, F., Stark, S., Chernyshenko, O.S., Nye, C.D. & Hulin, C.L. (2012). Development of the tailored adaptive personality assessment system (TAPAS) to support Army selection and classification decisions (ARI Technical Report 1311). Arlington, VA: U.S. Army Research Institute.

Durlach, P. J. & Ray, J. M. (2011). Designing adaptive instructional environments: Insights from empirical evidence (ARI Technical Report 1297). Arlington, VA: U.S. Army Research Institute.

Gully, S. & Chen, M. (2010). Individual differences, attribute-treatment interactions, and training outcomes. In S. W. J. Kozlowski & E. Salas (Eds.), Learning, training, and Development in Organizations (pp. 3-64). New York: Routledge.

Holden, H.K., Sottilare, R.A., Goldberg, B.S. & Brawner, K.W. (2012) Effective learner modeling for computer-based tutoring of cognitive and affective tasks. Paper presented at the presented at the Interservice/Industry Training, Simulation and Education Conference (I/ITSEC), Orlando, FL.

Kokini, C., Carroll, M.B., Ramirez-Padron, R., Hale, K.A., Sottilare, R., Goldberg, B (2012). Quantification of Trainee Affective and Cognitive State in Real-time. Paper presented at the presented at the Interservice/Industry Training, Simulation and Education Conference (I/ITSEC), Orlando, FL.

Murray, T. & Arroyo, I. (2002). Towards Measuring and Maintaining the Zone of Proximal Development in Adaptive Instructional Systems. Sixth International Conference on Intelligent Tutoring Systems. Springer.

Schultz, R.A., Alderton, D.L. & Hyneman, A.B. (2011) Individual differences in learning performance in computer-based training. (NPRST-TN-11-4) Millington, TN: Navy Personnel Research, Studies, and Technology.

Sottilare, R., Graesser, A., Hu, Xiangen & Holden, H. (Eds.) (2013). Design Recommendations for Intelligent Tutoring Systems – Volume 1, Learner Modeling. Orlando FL: U.S. Army Research Laboratory.

Sottilare, R.A. & Proctor, M. (2012) Passively Classifying Student Mood and Performance within Intelligent Tutors. Educational Technology & Society, 15 (2).

Vygotsky, L.S. (1978). Mind in Society: The development of higher psychology processes. Cambridge MA: Harvard University press.

Woolf, B., Arroyo, I., Muldner, K., Burleson, W., Cooper, D., Dolan, R. & Christopherson, R. (2010). The effect of motivational learning companions on low achieving students and students with disabilities. International Conference on Intelligent Tutoring Systems (327–337).

Toward Detection of Engagement and Affect in a Simulation-based Combat Medic Training Environment

Ryan S. Baker, Jeanine A. DeFalco, Jaclyn Ocumpaugh, Luc Paquette
Columbia University Teachers College

Introduction

Recently, there has been increasing research in intelligent tutoring systems (ITSs) on improving learning by automatically detecting affect and providing affective support. Systems have been developed to detect affect from both physiological sensors and students' interactions with an online learning system. In this paper, we discuss ongoing efforts to construct sensor-free detectors of trainee affect and behavioral engagement for vMedic, an immersive software used to train military trainees in combat medic field procedures. This paper reviews rationale and plans to develop models of affect and engagement based on log files and field observations from a September 2013 study at West Point conducted jointly by the U.S Army Research Laboratory (ARL), Teachers College, and North Carolina State University. Additionally, this paper discusses the preparation for the distillation process, plans for development of the affect detector model, and the implications of future integration into the Generalized Intelligent Framework for Tutoring (GIFT) Framework's Pedagogical module.

In recent years, developers of ITSs have turned their attention toward understanding student affect and engagement. It is widely known that the interaction of affect and engagement shapes learning in complex ways (Baker et al., 2010; Baker et al., 2011; D'Mello et al., 2007; Dragon et al., 2008; Lee et al., 2011; Sabourin et al., 2011). Take boredom, an affective state replicably shown to be associated with poorer learning and educational outcomes (Pardos et al. 2013). For instance, boredom can lead either to gaming the system (Baker et al., 2010) and off-task behavior (Baker et al., 2011). When boredom leads to gaming, boredom persists, creating a vicious cycle (Baker et al., 2010). By contrast, off-task behavior relieves boredom (Baker et al., 2011). Since gaming the system is much more strongly correlated with learning and educational outcomes (negatively) than off-task behavior (Pardos et al. 2013; Cocea et al., 2009), the different student responses to boredom matter. So, too, the affective states of confusion and frustration can be associated with positive or negative learning outcomes, depending on the length of their duration (Lee et al., 2011; Liu et al., 2013). Affect also influences learning through its effects on memory, attention, and strategy use (Pekrun, 1992; Schunk & Zimmerman, 2007). Learners who experience positive emotions are better able to retrieve information connected with such feelings (Forgas, 2000). Research has also shown that negative states may trigger greater cognitive load, which reduces working memory (Linnenbrink & Pintrich, 2000). While these processes are not thoroughly understood, it is likely that affective states such as frustration and anxiety can draw cognitive resources away from the task at hand to focus on the source of the emotion (Zeidner, 1998). Because of the range of impacts that affect can have on learning, systems that explicitly consider affect have been able to positively impact students' engagement and learning (Baker et al., 2010).

Researchers have relied on a variety of strategies when constructing models of engagement and affect. One approach that has been popular and successful is to utilize physical sensors such as video cameras that capture facial expressions, posture sensors that detect when a student is shifting positions, and galvanic skin response sensors that detect sweating (Conati, 2002; Mohammad & Nishida, 2010; Alzoubi et al., 2009). The use of sensors has yielded successful detectors of a range of affective states (Alzoubi et al., 2009), but it is not always feasible to implement the resulting detectors into the field. As such, other researchers have turned their analysis toward the fine-grained interaction logs produced by online learning systems, developing sensor-free detectors of affect and engagement.

Previous research on sensor-free detection has shown that this modeling strategy can be successful and affords analysis of learner actions at multiple levels. Successful results have been obtained for a variety of systems, including straightforward problem-solving ITSs (e.g., Baker et al., (2012) and Rodrigo & Baker (2009)), dialogue tutors (e.g., Litman & Forbes-Riley (2006)), science simulations (e.g., Paquette et al. (in press)), and narrative-based virtual environments (e.g., Baker et al. (in press)). Across these systems, models have been developed for affective states such as boredom, frustration, confusion, and engaged concentration, and for disengaged behaviors such as gaming the system and off-task behavior. Although the resulting models are often quite complex, detectors constructed solely from interaction log data offer a significant advantage: they can be used in settings where physical sensors are unavailable, enabling greater scale.

So far, one of the important lessons of developing sensor-free models of affect and engagement for a range of different environments is that the types of behaviors associated with the same affect may differ substantially between systems. For example, affective models developed for ASSISTments (Pardos et al. 2013), an ITS that provides scaffolded instruction for middle-school mathematics problems, demonstrate that the timing of learner actions is almost as important as the actions themselves. On the other hand, models constructed for EcoMUVE (Baker et al., in press), an immersive virtual environment that teaches environmental science, relied more heavily on the type of action the student was making to predict affect, such as whether the student was repeating specific kinds of measurement and what information pages the student was accessing in the virtual data manual (Baker et al., in press). Understanding the different behaviors commonly associated with affect and engagement in different systems is an important step toward the development of general frameworks for affect/engagement detection that facilitate the development of affect and engagement detectors for new learning environments.

In this paper, we discuss plans to build upon this body of previous research in the context of vMedic, an immersive ITS integrated with GIFT (Sottilare et al., 2012) provided through the GIFT framework’s Trainee module. We provide an overview of the methods that will be used to construct these detectors, developed through a combination of quantitative field observation (QFO) methods and educational data mining (EDM) techniques. These detectors will serve as an example of how to integrate this type of model into GIFT, will offer insight into the sorts of behavior that are correlated with learner engagement, will be useful for automated interventions, and will serve as a springboard for further research on learning and engagement within the vMedic software.

Data and Methods

Two sources of data will be used for this study: log file data produced by learners using the vMedic software and quantitative field observations of those learners while using the system. This section describes both sources of data, providing information about learning system (and module) that the subjects will participate in, the subjects themselves, and the field observations. It then describes the data mining techniques that will be used to generate our automated detectors.

Learning System and Subjects

We will model engagement within the context of hemorrhage control learning materials in vMedic (a.k.a. TC3Sim) serious game used to train U.S. Army combat medics and lifesavers on tasks associated with dispensing tactical field care and care under fire. It is part of the U.S. Army's objective to devise portable learning platforms that can be deployed quickly, effectively, and inexpensively to U.S. warfighters stationed around the world. vMedic has been integrated with GIFT (Sottilare et al., 2012) that monitors real-time interaction and can trigger feedback scripts for participants based on actions in the game and performance.

Designed to support real-time performance messages, current and predicted cognitive and affective states, GIFT is modular and service-oriented. GIFT contains the components of Sensor, Trainee, Pedagogical, learning management system (LMS), and Domain modules. The automated detectors developed in this project will enhance the Trainee module, toward providing support for creating pedagogical interventions (driven by the Pedagogical model) associated with game-specific actions in vMedic. By better modeling trainees' states, and understanding the roles different affective states and disengaged behaviors play during learning, we can generate pedagogical support tailored to individual trainees, helping to realize ARL's vision of tailored, self-regulated, individual tutoring experiences for U.S. Army trainees.

Compared with other systems, game-based environments like vMedic, where trainees interact with a virtual world through an avatar, place fewer constraints on learner actions than problem-solving systems (Pardos et al. 2013; Baker et al., 2012) or dialogue tutors (Litman & Forbes-Riley, 2006)]. Some virtual environments may present more constraints on learner behavior than others. vMedic is more restrictive than EcoMUVE (Baker et al., in press), where students have considerable freedom to explore the virtual world as they please. While vMedic allows a considerable amount of learner control, scenarios impose structure on trainee experiences through events that are triggered within the scenarios independent of the participant's actions (e.g., explosions and injuries occur and require attention irrespective of the actions of the participant within the scenario). These scenarios provide help in focusing the participant's attention to the objectives of the game (administering care) and implicitly guide trainee experiences toward key learning objectives.

For this study, predominantly first-year cadets from West Point will be observed at the beginning of the academic year. The cadets' voluntarily participation in a one-hour session will be held in a West Point computer laboratory. At the start of the session, cadets will be fitted with Q-sensors and synchronized with Kinect depth sensors (for subsequent analyses, outside the scope of this paper, of the relative value of sensor-based detectors and interaction-based detectors). Cadets then initiate a unique user profile by logging into GIFT. Cadets will be asked to answer questionnaires on self-efficacy, complete a pre-test on hemorrhage control, and review a PowerPoint presentation on hemorrhage control. Following the PowerPoint presentation, the cadets engage in approximately 20 minutes of the vMedic game, and then complete a post-test.

Quantitative Field Observations (QFOs)

In this study, QFOs will be collected using the Baker-Rodrigo Observation Method Protocol (BROMP) (Wixon et al, 2012). As mandated by BROMP, coders must be certified, achieving an adequate inter-rater reliability of Kappa = 0.6 with another BROMP-trained coder. BROMP has been used for several years to study behavior and affect in educational settings (Baker et al., 2010; Baker et al., 2011; Baker et al., 2012; Rodirgo & Baker, 2009) and has been used as the basis for successful automated detectors of affect (Pardos et al. 2013; Baker et al., 2012). Observations in this study will be conducted by two BROMP-certified coders.

Within the BROMP protocol, behavior and affective states are coded separately but simultaneously using the Human Affect Recording Tool (HART), an application developed for the Android platform (and freely available as part of the GIFT distribution). HART enforces a strict coding order, determined at the beginning of each session. Students or trainees are coded individually, and coders are trained to rely primarily on peripheral vision in order to minimize observer effects. The coder has up to 20 seconds to categorize each trainee's behavior and affect, but records only the first thing he or she sees. In situations where the trainee has left the room, where his or her affect or behavior do not match any of the categories in the current coding scheme, or when the trainee can otherwise not be adequately observed, a "?" is recorded, and that observation is eliminated from the training data used to construct automated detectors.

The typical coding schemes used by BROMP will be modified to accommodate the unique behaviors and affect that manifest for this specific cadet population. Affective states observed will include frustration, confusion, engaged concentration, boredom, surprise, and disdain. Behavioral categories will consist of on-task behavior, off-task behavior, psychopath (friendly fire, killing bystanders; may not be observed in practice), and WTF ("without thinking fastidiously") behavior (Goldberg et al., 2011).

Feature Distillation

In order to distill a feature set for our affect detectors, trainee actions within the software will be synchronized to the field observations. During data collections, both the handheld computers and the GIFT server will be synchronized to the same internet NTP time server. Actions during the 20 seconds prior to data entry by the observer will be considered as co-occurring with the observation. It is anticipated that the following features will be engineered using data from the actions that co-occurred with or precede the observations:

- Changes in the state of the casualty, both recent and since injury, including:
    - Blood pressure and volume
    - Heart rate
    - Bleed rate
    - Lung efficiency

- Attempts by player, both during clip and overall, to treat patient
    - Application of tourniquet
    - Checking vitals
    - Conducting a blood sweep
    - Communicating
    - Requesting medevac
- Player state in terms of attackers
    - Is player under fire?
    - Is player under cover? (Are they currently taking cover?)
    - Is player with unit?
    - Is casualty safe from further attack?
- Time between actions

These features will be initially constructed within Microsoft Excel as a rapid prototyping method. After we have determined which features are predictive of affect, these features will be integrated into the GIFT platform and automatically distilled. Other features may also be developed through the process of feature engineering and studying their effects when integrated into the system.

Machine Learning Process

It is anticipated that separate detectors will be built for each affective and behavioral construct studied, although detectors for very rare behaviors or affective states may not be developed, due to lack of data.

Each detector will be evaluated using leave-one-out trainee-level cross-validation. In this process, a detector is built using data from every trainee except one before being tested on that student. By cross-validating at this level, we increase confidence that detectors will be accurate for new trainees. In addition, re-sampling will be used to make the class frequency more equal for detector development. However, all performance calculations will be made with reference to the original dataset, as in Baker et al. (2012).

Conclusion

It is the intent of the investigators on this project that the results will contribute to ARL's goal of further developing the GIFT framework (Sottilare, et al, 2012). Developing automated detectors that can integrate sensor and interaction (including performance and history of the trainee) but can function effectively absent sensor data, is an important step in developing affect interventions for U.S. Army trainees across a range of settings.

Optimally leveraging these detectors will depend on several additional steps. First, it is essential to study the relationship between affect, engagement, and outcome variables, toward understanding which affective states and engagement variables need to be responded to in a suite of optimally effective computer-based tutoring systems for Army use. The data collected to develop these detectors can be expected to also be useful in accomplishing this goal. Second, interventions will need to be developed which leverage the detectors to effect change in trainees. There is an extensive literature on interventions for behavioral disengagement and affect, in online learning systems; however, it is not fully known which of these approaches will be effective with military trainees, a different population than the much younger populations these interventions have typically been developed for. Once interventions have been developed and tested, integrating automated detectors and interventions into vMedic through GIFT's Trainee module and Pedagogical module will provide a valuable example of how to respond to trainees' negative affect and disengagement, a valuable contribution in improving vMedic and similar interventions used by the U.S. Army.

Acknowledgments

This research was supported by a grant by ARL, Grant #W9NF-13-2-0008. We thank our research colleagues at ARL, Dr. Robert Sottilare, Dr. Benjamin Goldberg, Dr. Keith Brawner, and Dr. Heather Holden, as well as at North Carolina State University, Dr. James Lester, Dr. Bradford Mott, and Dr. Jonathan Rowe. Any opinions, findings, and conclusions expressed in this paper are those of the authors and do not necessarily reflect the views of ARL or NCSU.

References

Alzoubi, O., Calvo, R.A., Stevens, R.H. Classification of EEG for affect recognition: An adaptive approach. In AI 2009: Advances in Artificial Intelligence, Springer, Berlin Heidelberg, (2009), 52-61.

Baker, R.S., Ocumpaugh, J., Gowda, S.M., Kamarainen, A., Metcalf, S.J. (in press) Extending log-based affect detection to a multi-user virtual environment for science. To appear in Proceedings of the 22nd Conference on User Modelling, Adaptation, and Personalization.

Baker, R.S.J.d., D'Mello, S.K., Rodrigo, M.M.T., Graesser, A.C. 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. In International Journal of Human-Computer Studies, 68 (2010), 223-241.

Baker, R.S.J.d., Gowda, S.M., Wixon, M., Kalka, J., Wagner, A.Z., Salvi, A., Aleven, V., Kusbit, G., Ocumpaugh, J., Rossi, L. Towards sensor-free affect detection in cognitive tutor algebra. In proceedings of Educational Data Mining 2012, (2012), 126-133.

Baker, R.S.J.d., Moore, G., Wagner, A., Kalka, J., Karabinos, M., Ashe, C., Yaron, D. The Dynamics Between Student Affect and Behavior Occurring Outside of Educational Software. In Proceedings of the 4th bi-annual International Conference on Affective Computing and Intelligent Interaction, (2011).

Cocea, M., Hershkovitz, A., Baker, R. S. J. d. The impact of off-task and gaming behaviors on learning: Immediate or aggregate? In Proceedings of the 14th international conference on Artificial Intelligence in Education, (2009), 507-514.

Conati, C. Probabilistic assessment of user's emotions in educational games. In Journal of Applied Artificial Intelligence, 16 (2002), 555-575.

D'Mello, S.K., Taylor, R., Grasser, A.C. Monitoring affective trajectories during complex learning. In proceedings of the 29th Annual Cognitive Science Society, (2007), 203-208.

Dragon, T., Arroyo, I., Woolf, B.P., Burleson, W., Kaliouby, R.e., Eydgahi, H. Viewing student affect and learning through classroom observation and physical sensors. In Proceedings of the International Conference on Intelligent Tutoring Systems, (2008), 29-39.

Forgas, J.P. Managing moods: Toward a dual=process theory of spontaneous mood regulation. In Psychological inquiry, 11(2000), 172-177.

Goldberg, B., Holden, H., Brawner, K. & Sottilare, R. Enhancing performance through pedagogy and feedback: Domain considerations for ITSs. In Interservice/Industry Training, Simulation, and Education Conference, I/ITSEC, (2011).

Lee, D.M., Rodrigo, M.M., Baker, R.S.J.d., Sugay, J., Coronel, A. Exploring the relationship between novice programmer confusion and achievement. In Proceedings of Affective Computing and Intelligent Interaction 2011, (2011), 175-184.

Linnenbrink, E.A., Pintrich, P.R. Multiple pathways to learning and achievement: The role of goal orientation in fostering adaptive motivation, affect, and cognition. In Intrinsic and Extrinsic Motivation: The Search for Optimal Motivation and Performance, C. Sansone & J.M. Harackiewicz, Eds. Academic Press, San Diego, (2000), 195-227.

Litman, D.J., Forbes-Riley, K. Recognizing student emotions and attitudes on the basis of utterances in spoken tutoring dialogue with both humans and computer-tutors. In Speech Communication, 48 (2006), 559-590.

Liu, Z., Pataranutaporn, V., Ocumpaugh, J., Baker, R. S. Sequences of frustration and confusion, and learning. In Proceedings of the 6th International Conference on Educational Data Mining, (2013), 114-120.

Mohammad, Y. & Nishida, T. Using physiological signals to detect natural interactive behavior. In Spring Science & Business Media, 33 (2010), 79-92.

Paquette, L., Baker, R.S., Sao Pedro, M.A., Gobert, J.D., Rossi, L., Nakama, A., Kauffman-Rogoff, Z. (in press). Sensor-free affect detection for a simulation-based science inquiry learning environment. To appear in Proceedings of the 12th International Conference on Intelligent Tutoring Systems.

Pardos, Z., Baker, R.S.J.d., San Pedro, M.O.Z., Gowda, S.M., Gowda, S. Affective states and state tests: Investigating how affect throughout the school year predicts end of year learning outcomes. In Proceedings of Leaning Analytics and Knowledge 2013, (2013), 117-124.

Pekrun, R., Educational environments and intrinsic/extrinsic academic motivation: Expectancy value theory and longitudinal evidence. In paper presented at the Annual Meeting of the American Educational Research Association, San Francisco, CA. (1992).

Rodrigo, M.M.T., Baker, R.S.J.d. Coarse-grained detection of student frustration in an introductory programming course. In Proceedings of the International Computing Education Research conference 2009: the International Computing Education Workshop, (2009).

Sabourin, J., Rowe, J., Mott, B., Lester, J. When off-task is on-task: The affective role of off-task behavior in narrative-centered learning environments. In Proceedings of Artificial Intelligence in Education 2011, (2011), 534-536.

Schunk, D.H., Zimmerman, B.J. Influencing children's self-efficacy and self-regulation of reading and writing through modeling. In Reading and Writing Quarterly 23 (2007). 7-25. .

Sottilare, R.A. Goldberg, B. & Holden, H. The Generalized Intelligent Framework for Tutoring (GIFT), (2012).Ocumpaugh, J., Baker, R.S.J.d., Rodrigo, M.M.T. Baker-Rodrigo Observation Method Protocol (BROMP) 1.0. Training Manual version 1.0. Technical Report. In EdLab: New York, NY, Manila, Philippines: Ateneo Laboratory for the Learning Sciences (2012).

Wixon, M., Baker, R.S.J.d., Gobert, J., Ocumpaugh, J., Bachmann, M. WTF? Detecting students who are conducting inquiry Without Thinking Fastidiously. In Proceedings of the 20th International Conference on User Modeling, Adaptation and Personalization, UMAP (2012), 286-298)..

Zeidner, M. Test Anxiety: The State of the Art. Berlin, German, (1998).

Personalization, Non-Cognitive Factors, and Grain-Size for Measurement and Analysis in Intelligent Tutoring Systems: Implications for GIFT

Stephen E. Fancsali[1], Steven Ritter[1], John C. Stamper[2], Susan Berman[1]
[1]Carnegie Learning, Inc., Pittsburgh, PA, USA
[2]Human-Computer Interaction Institute, Carnegie Mellon University, Pittsburgh, PA, USA

Introduction

One major goal of our Advanced Distributed Learning (ADL)-funded "Hyper-Personalized Intelligent Tutor" (HPIT) project is to develop an architecture that allows for personalization of educational software based on non-cognitive factors including student preferences, motivation, affect, and other features (Fancsali, Ritter, Stamper & Nixon, 2013). We are particularly interested in using such information to improve instruction within intelligent tutoring systems (ITSs) like Carnegie Learning's Cognitive Tutor® (CT) (Ritter, Anderson, Koedinger & Corbett, 2007) and in educational games like the mathematics "fluency challenges" developed by the Math Fluency Data Collaborative project (http://fluencychallenge.com). To better understand architectural features conducive to such "hyper-personalization," we have built on learning science researchers' recent work focusing on particular aspects of personalization and non-cognitive factors in both the CT and a middle school mathematics ITS based on the CT called MATHia.™

We review results of two recent observational studies with respect to the U.S. Army Research Laboratory's (ARL) Generalized Intelligent Framework for Tutoring (GIFT). In the first, we provide a causal model of measures of goal orientation and self-efficacy (at multiple grain-sizes), behavioral measures in the CT, and learning outcomes (Fancsali, Bernacki, Nokes-Malach, Ritter & Yudelson, in press). These findings are consistent with research (Bernacki, Aleven & Nokes-Malach, 2012) showing that "domain-level" (for the mathematics domain) and "unit-level" (for a CT unit of instruction) measures of goal orientation and self-efficacy provide different information about learners and their outcomes. Such results suggest that these measures are contextually bound and thus need to be assessed relatively frequently.

In the second study, we consider aggregate associations of in-tutor measures of performance with "honoring" student preferences for interest areas (e.g., "sports & fitness") around which mathematics word problems have been authored. We adopted a different approach from past research by considering aggregate associations between honoring preferences and learning outcomes, in order to assess whether, at the timeframe of a school year, we see influences of honoring student preferences. We found little if any association between honoring student preferences and learning outcomes (Fancsali & Ritter, 2014; Ritter, Sinatra & Fancsali, in press). Past experimental studies (Walkington 2013) have found effects at finer levels of granularity (e.g., at the level of problems and skills). We speculate that the failure to find such effects in our study may reflect the fact that, in the context of the full-year course, the opportunities to honor student preferences are limited. This finding suggests that taking advantage of preference honoring in an extended course may require more frequent or overt signaling about attention to student interests.

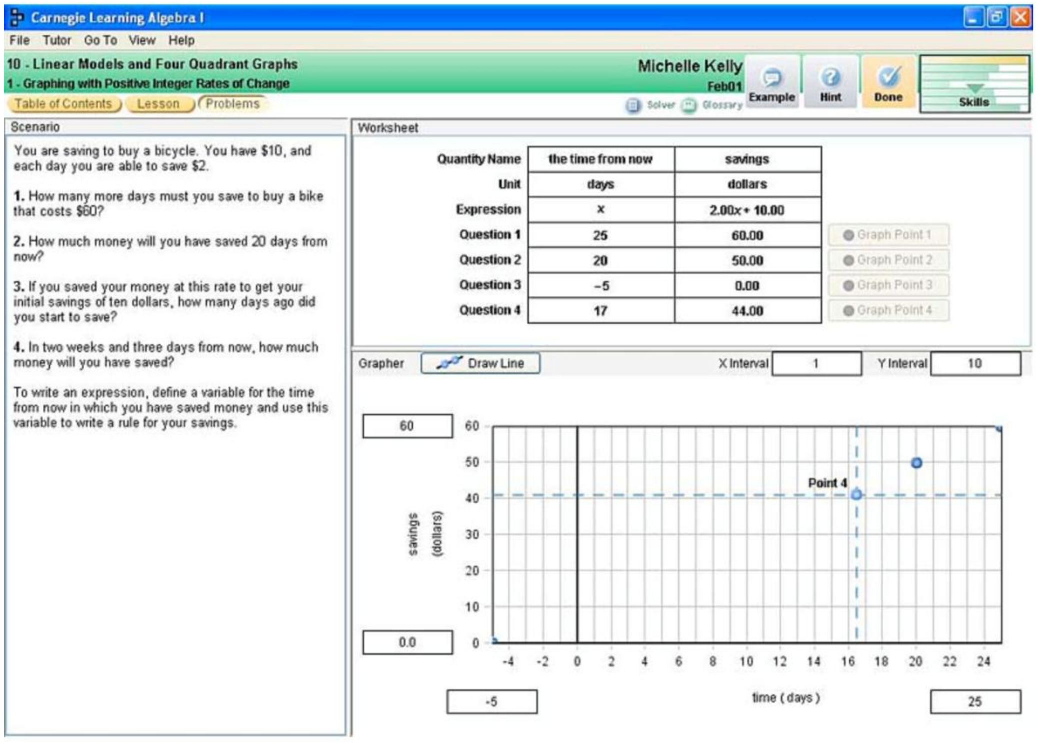
With respect to the GIFT architecture, we discuss a response to recent work (Otieno, Schwonke, Salden & Renkl, 2013), due to Fancsali et al. (submitted), suggesting that using relatively simple log traces from CT as “online measures” of goal orientation and how the aforementioned goal orientation and self-efficacy study informs this response. Generally, we provide new recommendations for GIFT and a better, data-driven foundation for recommendations concerning GIFT made by Fancsali, Ritter, Stamper & Nixon (2013). We suggest future research in the areas of non-cognitive factors and personalization for ITSs and development areas for HPIT and GIFT. Specifically, we emphasize that the grain-size of measurement and analysis required for adaptation within a particular ITS or instructional system may be different than that at which meaningful inferences (and corresponding adaptive instruction or tutoring decisions) might be made across *different* ITSs or instructional systems.

Preliminaries

We begin by describing Carnegie Learning’s CT, the ITS which provides the impetus for our thinking about hyper-personalization of instructional systems as well as the target platform for the research projects we explicate. We then describe the HPIT project as well as two non-cognitive factors and a facet of personalization along which such a hyper-personalized instructional system might tailor instruction. As we proceed, we note an important aspect of adaptation or personalization in any such system: the level of granularity or grain size at which adaptation or personalization occurs.

Cognitive Tutor (CT)

Carnegie Learning’s CT (cf. Figure 1 screenshot) is an ITS for mathematics that has been demonstrated effective by one of the largest experimental trials of its kind (Pane, Griffin, McCaffrey & Karam, 2013). CT provides adaptive instruction based on cognitive factors, specifically by tracking learners’ mastery of fine-grained knowledge components (KCs) (i.e., skills) using a probabilistic framework called Bayesian Knowledge Tracing (BKT) (Corbett & Anderson, 1995).



You are saving to buy a bicycle. You have $10, and each day you are able to save $2.

1. How many more days must you save to buy a bike that costs $60?

2. How much money will you have saved 20 days from now?

3. If you saved your money at this rate to get your initial savings of ten dollars, how many days ago did you start to save?

4. In two weeks and three days from now, how much money will you have saved?

To write an expression, define a variable for the time from now in which you have saved money and use this variable to write a rule for your savings.

Quantity Name	the time from now	savings	
Unit	days	dollars	
Expression	x	2.00x + 10.00	
Question 1	25	60.00	Graph Point 1
Question 2	20	50.00	Graph Point 2
Question 3	−5	0.00	Graph Point 3
Question 4	17	44.00	Graph Point 4

Figure 21. Problem solving in Carnegie Learning's CT

Mathematics curricula are divided into topical sections, each of which is associated with a set of KCs that a student is required to master before progressing on to another section of instruction. Each multi-part problem in a section is associated with a subset of its section's KCs. Students are provided immediate feedback about the correctness of attempts to complete steps in each problem; sometimes CT provides just in time, context-sensitive feedback when students make errors that reflect known misconceptions. At any step of problem solving, students can also request context-sensitive hints. While using ITSs to adapt mathematics instruction based on cognitive factors like skill mastery is relatively well-understand (but still a topic of on-going research), the use of non-cognitive factors to personalize and adapt instruction in ITSs presents a wide variety of unanswered and open questions.

Hyper-Personalized Intelligent Tutors (HPIT)

Our HPIT project (Fancsali, Ritter, Stamper & Nixon, 2013) aims to develop a plug-in based architecture and infrastructure to support personalized and adaptive learning, based on both cognitive and non-cognitive factors, in a variety of computer-based instructional settings, including, but not limited to, ITSs like CT and educational games. To truly "hyper-personalize" ITSs and other instructional systems, it is necessary to better understand how non-cognitive factors can be harnessed to enhance learning in such systems, and the HPIT project will provide a framework within which such research can take place, for example, by allowing developers to integrate plug-ins to drive personalized instructions based on particular non-cognitive factors in which they are interested.

To facilitate the development of such a software framework, a better understanding of particular methodological and measurement issues that will arise in such personalization efforts is also required. This understanding will not only improve the development of HPIT but will also help to drive integration and interoperability (as well as illustrate possible contrasts) between HPIT and projects like GIFT. Findings relevant to HPIT may also drive new recommendations for future development of GIFT. Specifically, we briefly summarize recent work on goal orientation and self-efficacy as they relate to performance in the CT as well as "honoring" student interest area preferences in an ITS for middle school mathematics that is based on the CT. From methodological concerns and findings about specific non-cognitive constructs, we hope to generalize insights to provide recommendations for GIFT. We focus on issues of measurement and analysis grain-size and discuss implications for adaptation and tutoring within an instructional system as well as open problems about making transferable inferences across instructional systems, both of which are important issues relevant to GIFT.

Grain size and non-cognitive factors

We now turn our focus to analyses of process data and learning in the CT environment relative to two non-cognitive factors: goal orientation and self-efficacy[1]. We begin by briefly introducing these constructs before considering issues about their measurement and grain-size before considering relationships of these factors with learning outcomes in the CT in the following section.

Goal orientation and self-efficacy

Students' achievement goals for learning are a key component of their motivation for learning. Dweck (1986) identifies performance and achievement as two particular goal orientations. Students with a performance goal orientation tend to define their goals relative to the performance of their peers or some other group (i.e., to perform better than their peers) while those with a mastery goal orientation take understanding a particular task or aspect of a domain as their goal. Elliot & McGregor (2001) later distinguish between two "valences," approach (success) and avoidance (of failure), for mastery and performance goal orientation. For example, performance avoidance as a goal orientation is attributed to students who seek not to perform worse than their peers (i.e., to demonstrate that they are no worse than their peers). A performance approach orientation is attributable to students who seek to demonstrate their competence by out-performing their peer group. Students with mastery approach goals tend to seek to develop competency for a learning task by achieving at levels above their past performance or by setting other criteria for judging increased or developing mastery (Ames, 1992; Elliot, 1999).

Learner self-efficacy, or beliefs a learner has about his or her abilities in performing in a particular domain or task may also play a role for setting goals, self-regulation, and other factors important for learning (Bandura, 1994). Learners with high self-efficacy, for example, may be more willing to set more difficult or less easily attainable goals that require working through difficult tasks. Self-efficacy is also related to persistence and better performance on learning tasks (Bandura, 1997).

[1] We here provide a brief summary of the explication of these non-cognitive factors found in Fancsali et al. (in press).

Self-report questionnaires, especially those given before or after a particular learning task, are frequently used measure goal orientation and self-efficacy. However, several authors note that factors like goal orientation can change as learners progress in a learning experience or over longer periods of instructional time (e.g., an academic year) (Richardson, 2004; Fryer & Elliot, 2007; Muis & Edwards, 2009). A propos, we consider work that takes such differences in measurement grain size into account. Later, we consider the possibility of “online” measures (e.g., log traces from ITSs) of such non-cognitive factors.

Grain size

Given the possibly dynamic nature of non-cognitive factors, including goal orientation, Bernacki, Nokes-Malach & Aleven (2013) raise concerns for the granularity at which factors like goal orientation are measured in conjunction with use of ITSs like CT. Beyond reliable changes in learners’ goal orientation from unit to unit of CT instruction found by Bernacki, Nokes-Malach & Aleven (2012), the same authors also found that domain-level measures of goal orientation (e.g., for the mathematics domain) versus unit-level measures of goal orientation (e.g., for CT units of instruction) have different levels of association with various behaviors, which provides for the possibility that different measurement grain sizes are providing different information in possibly important ways (Bernacki, Nokes-Malach & Aleven, 2013). We report recent work due to Fancsali et al. (submitted) that provides a linear structural equation model relating data collected on self-efficacy at the unit- and domain-levels and learners’ goal orientations. This work suggests that the move from self-report questionnaires to relatively simple log traces to measure achievement goals, including counts of hints and glossary access in the CT, is likely more nuanced than suggested by recent work of Otieno, Schwonke, Salden & Renkl (2013). Before describing this work, we briefly consider one facet of personalization based on non-cognitive factors that provides the basis of the second study we report.

Student preferences, interest area personalization, and grain size

Computer-based ITSs like CT can provide functionality that allows for context personalization, roughly aligning instructional content to interests of an individual learner (Anand & Ross, 1987; Cordova & Lepper, 1996; Walkington, 2013). For instance, Carnegie Learning’s MATHia product, an ITS for middle school mathematics based on CT, provides for mathematics word problems to be tailored to specific interest areas of learners for domains outside of the classroom (e.g., “sports & fitness”) as well as based on student-entered names of their favorite classmates or friends. Personalizing instruction based on both of these non-cognitive factors (i.e., problem content and names) has been demonstrated effective by previous experiments (Anand & Ross, 1987; Cordova & Lepper, 1996; Ku & Sullivan, 2002). With respect to CT, research using an experimental version of this ITS finds that personalizing mathematics word problems based on student interest areas improves learning outcomes (i.e., performance) at the problem-level, especially on problems with a relatively high reading level that involve difficult KCs (Walkington, 2013). One posited mechanism at the level of KCs is that personalization improves students’ ability to symbolically represent word problems at a higher reading level. However, as we describe later, at a higher level of granularity, when we have considered the aggregate association of “honoring” student preferences with learning outcomes, our results are less clear (Fancsali & Ritter, 2014; Ritter, Sinatra & Fancsali, in press).

Goal Orientation, Self-Efficacy, Behavior, and Learning

To better understand, among other things, the proposal of Otieno, Schwonke, Salden & Renkl (2013) that "online" measures from ITS log data, simple counts of hints requested and glossary use, be used to assess achievement goals or goal orientation (performance approach and mastery approach goals, respectively), Fancsali et al. (in press) learned qualitative causal structure, using data-driven procedures based on conditional independence tests and background knowledge (Spirtes, Glymour & Scheines, 2000), to specify a linear structural equation model (estimated and illustrated as Figure 2[2]) over features representing goal orientation, self-efficacy, math grades (pre/post), and process data from CT.

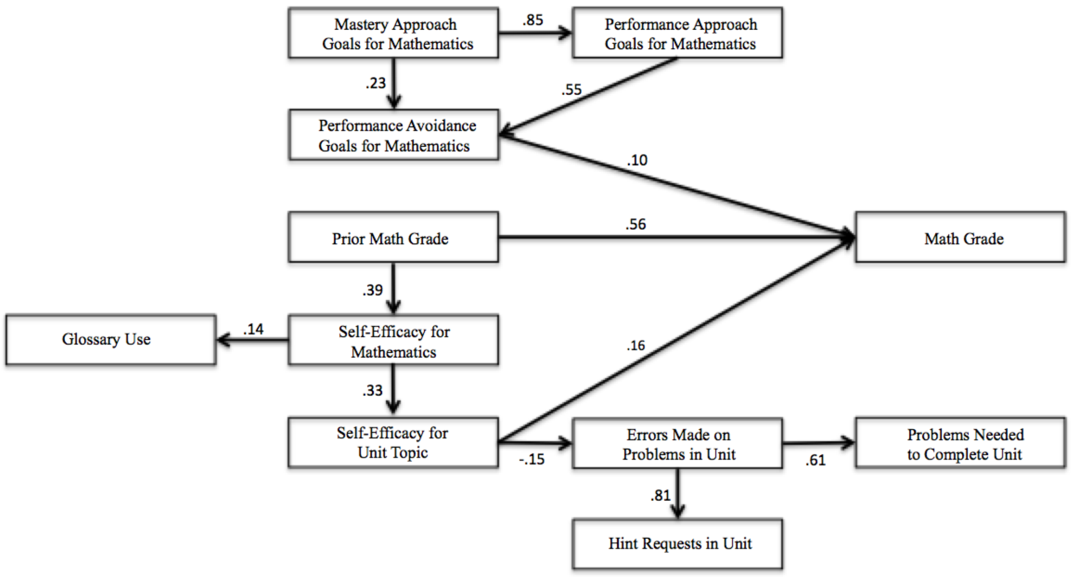

Mastery Approach Goals for Mathematics
.85
Performance Approach Goals for Mathematics
.23
.55
Performance Avoidance Goals for Mathematics
.10
Prior Math Grade
.56
Math Grade
.39
Glossary Use
.14
Self-Efficacy for Mathematics
.16
.33
Self-Efficacy for Unit Topic
-.15
Errors Made on Problems in Unit
.61
Problems Needed to Complete Unit
.81
Hint Requests in Unit

Figure 22. Estimated linear structural equation model reported by Fancsali, Bernacki, Nokes-Malach, Ritter & Yudelson (submitted), including standardized path coefficients.

Data were collected for 273 pre-algebra, algebra, and geometry students using the CT during an experiment run by Matthew Bernacki, Timothy Nokes-Malach, and Vincent Aleven in a school district in western Pennsylvania to investigate issues of grain-size and measurement of performance goals and self-efficacy. Self-report survey items[3] for achievement goals were original items from subscales for performance approach, performance avoidance, and mastery approach on the Achievement Goals Questionnaire – Revised (Elliot & Murayama, 2008). The study designers crafted self-efficacy survey items according to guidance provided by Bandura (2006). Domain-level survey items (i.e., with respect to learner's feeling about mathematics) were administered within the CT software at the beginning of the school year, while unit-level items (with language adapted to refer to the particular CT unit of instruction students had just completed) were presented after every other unit of instruction (alternating with unit-level self-efficacy items).

[2] Their reported model fits the data well according to the appropriate chi-square statistical test for such a model comparing its implied covariance matrix to the observed covariance matrix ($\chi^2(43) = 49.19$, $p = 0.239$) (Bollen, 1989).
[3] For complete details of survey items, see Fancsali et al. (in press).

While the proposal of Otieno, Schwonke, Salden & Renkl would have us count *Glossary Use* as a measure of mastery approach, we find that it is weakly linked only to domain-level learner self-efficacy (i.e., for mathematics); the model of Figure 2 implies that conditional on *Self-Efficacy for Mathematics*, *Glossary Use* is independent of all other variables included in the model. Similarly, *Hint Requests in Unit*, a proposed measure of performance approach goal orientation, is strongly correlated with *Errors Made on Problems in Unit*, which in turn is only weakly linked to *Self-Efficacy for Unit Topic* (i.e., unit-level self-efficacy). The model is inconsistent with any strong link between hint use and achievement goals.

The model posits that three variables are directly linked to *Math Grade*: *Prior Math Grade*, *Self-Efficacy for Unit Topic*, and *Performance Avoidance Goals for Mathematics*. Notably, this set of variables includes both a unit-level and domain-level variable, and we see that these different levels of granularity do provide different information about behavior in CT. Direct links in this model, and especially the weak correlation of *Performance Avoidance Goals for Mathematics* with *Math Grade* should be interpreted cautiously, as the algorithm used to infer the structure of the model in Figure 2 assumes there are no unmeasured common causes of measured variables, and this assumption is unlikely to obtain in this application domain (i.e., the adage to not confuse correlation and causation is operative). Nevertheless, the model is useful for providing basic structural relationships from correlational data and providing information about what relationships *are not consistent* with observed data. Further, the pattern of high correlations among goal orientation variables is consistent with previous literature positing a "goal complex" (Barron & Harackiewicz, 2001; Senko, Hulleman & Harackiewicz, 2011); one explanation of such a "complex" would include one or more latent variables representing some more complex factor(s). We return to the problem of developing online measures for important non-cognitive factors like goal orientation and self-efficacy as well as issues of grain size and implications for GIFT in the discussion section.

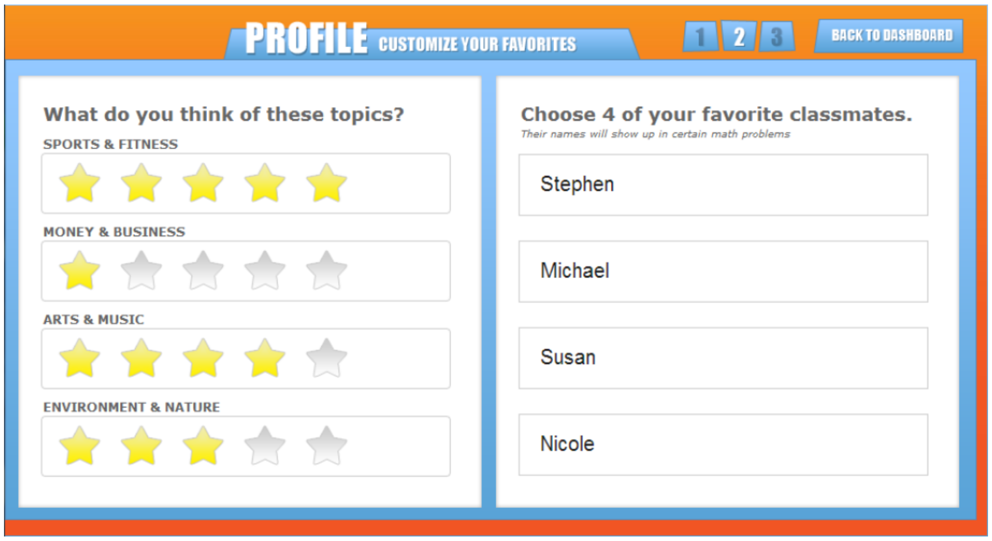
"Honoring" Student Preferences and Learning

While previous CT experiments had demonstrated improved problem-level learning outcomes from personalizing mathematics word problems based on student interest areas, less clear was whether "honoring" interest area preferences, in the aggregate, is associated with better performance in an ITS like CT. To begin considering this question, we analyzed observational data from Carnegie Learning's MATHia ITS, which is based on CT (Fancsali & Ritter, 2014).[4] While MATHia allows students to (optionally) rate four interest areas with one to five "stars," (cf. Figure 3) we found, in a sample of 104,197 learners, 62,168 learners (59.7%) set interest areas, but 16,003 learners (25.7% of those who set interest areas) provided the same rating to all interest areas.

Consequently, we employ a notion of strong student preferences according to which a student has strong preferences if (and only if) they rate at least one interest area with five stars and at least one area with one star (or leave at least one area un-rated). From a smaller sample of 1,230 learners (from eight randomly selected schools that completed at least five instructional sections that contain preference-tailored problems and upon whom our detailed analysis focused), we found that 518 students had strong preferences

[4] Results reported in this section are described in detail by Fancsali & Ritter (2014).

according to this definition. We then calculated the proportion of problems presented to each of these students (in sections that contained preference-tailored word problems) that corresponded to their highly rated interest area as a measure of the extent to which their preferences were "honored."



What do you think of these topics?

SPORTS & FITNESS ★★★★★

MONEY & BUSINESS ★☆☆☆☆

ARTS & MUSIC ★★★★☆

ENVIRONMENT & NATURE ★★★☆☆

Choose 4 of your favorite classmates.

Their names will show up in certain math problems

Stephen

Michael

Susan

Nicole

Figure 23. MATHia's preference dashboard

We found that the probabilistic provision of preference-tailored problems led to a relatively restricted range of proportions of preference-honoring per student, with the majority of students seeing somewhere between 10%–40% of problems in preference-tailoring sections that corresponded to their interests. Given that MATHia does not explicitly signal to students that problems have been provisioned according to their preferences, there is some question as to whether students generally realize that the software is adapting to their interests. Further, we considered correlations of proportion of preference-honoring problems to four process variables found in past work to be predictive of standardized test scores (Ritter, Joshi, Fancsali & Nixon, 2013): assistance per problem (i.e., number of hints requested & errors committed per problem), number of MATHia sections encountered, number of sections completed per hour (i.e., roughly efficiency), and amount of time logged into MATHia.

While three of four correlations with proportion of preference-honoring problems were significant (at the $\alpha = 0.05$ level), no correlation had an absolute value greater than 0.14, and it is unclear that there is any substantive use to be made of such weak, aggregate correlations. Further, the directions of these correlations imply a slight negative relationship between preference honoring and MATHia performance, but it is unclear whether this also has any substantive significance. Interestingly, we found that there were statistically significant differences (at the $\alpha = 0.05$ level) between strong and weak preference students, as strong preference students worked through MATHia material more efficiently (i.e., greater sections completed per hour) (Welch two-sided, two sample $p = 0.01$, Cohen's $d = 0.2$)[5], and students who merely set preferences require (statistically) significantly less assistance per problem than those who do not set

[5] We omit mean values since Fancsali & Ritter (2014) normalize and transform process variables, following Ritter, Joshi, Fancsali & Nixon (2013) in such a way that mean values would be relatively un-interpretable without more significant explanation, which we omit for brevity.

preferences (p = 0.03, d = -0.12). Perhaps even more interestingly, students who set names of friends/classmates in MATHia outperformed those who did not set such names on all four process variable measures, requiring less assistance per problem (p < 0.001, d = -0.27) and encountering more MATHia sections (p =0.009, d = 0.16) while working more efficiently (p < 0.001, d = 0.12) in less time (p < 0.001, d = -0.23).

These results raise questions about the nature of the factor(s) (either cognitive or non-cognitive) for which use of the preference dashboard is serving as a proxy; possibilities include conscientiousness or attentiveness to the MATHia environment, but since total login time is not significantly different in two out of the three comparisons, possible notions of engagement for which dashboard use might serve as a proxy may be constrained. It is also possible that students may simply appreciate the opportunity to set preferences to which the ITS responds and adapts. Future versions of MATHia should make more explicit the prevision of problems to students based on their interests, while also honoring preferences overall more frequently and with greater variability to avoid the restricted range pitfall that we have discovered in our current implementation.

Discussion and Implications for GIFT

One immediate implication, if our hypothesis about learners' appreciation of an instructional system's personalization to their interest areas or adaptation based on other non-cognitive factors is true, is that GIFT might also benefit from providing means by which training applications can not only provide adaptive instruction but also make explicit (at least aspects of) the decision making process that led GIFT to "choose" a particular adaptive, instructional strategy. We now discuss several other implications derived from the above presentation of recent research on specific non-cognitive factors in practice.

Online measures

We agree with the argument of Otieno, Schwonke, Salden & Renkl (2013) that developing appropriate online measures of non-cognitive and other factors is important to improving the state of the art of ITS research because surveys tend to be (at least) time-consuming while providing only noisy measures of intended phenomena. One implication for an intelligent tutoring architecture, then, is that the tutor may, in addition to its role as a source of instruction, play a role as a "sensor" for various characteristics and states. As noted by Fancsali, Ritter, Stamper & Nixon (2013), developments in sensor-free, data-driven "detectors" (e.g., Baker 2007; Baker & de Carvalho 2008) of both relatively complex behavior and affective states might be extended to factors like achievement goals, providing important avenues for future research into data-driven, online methods for making inferences about such phenomena.

Transferable inferences and grain size

We have provided concrete examples of (still unresolved) issues raised by Fancsali, Ritter, Stamper & Nixon (2013) in the ever-growing body of research on ITSs. Differing levels of granularity in measurement and analysis provide disparate but possibly important information about learners" interactions with ITSs like CT, and the effects of some types of personalization (e.g., preference-tailored word problems) may only be seen at the fine-grained level of individual problems, based upon their composition in terms of KCs.

While much of the work we have presented and our previous recommendations describing the integration of HPIT and GIFT (Fancsali, Ritter, Stamper & Nixon, 2013) has focused on relatively fine-grained measurements and inferences necessary for a particular ITS or instructional system to provide appropriately adaptive educational experiences, relatively coarse-grained assessments and inferences may be most appropriate for GIFT to provide rich learner models for multiple instructional systems or training applications. However, it is not, at this stage, clear how, for example, non-cognitive (and meta-cognitive) factors like achievement goals or affective states like boredom in an ITS for mathematics might transfer to other training applications (when even factors like achievement goals, as noted above, have variability at different grain sizes during instruction that can provide important information about learning outcomes). It isn't necessarily true that, for example, detection of a student as bored in one tutor has any relevance to another tutor; perhaps the second tutor is more exciting to the student. For this reason, it makes sense to consider communicating not just conclusions (e.g., "boredom") to the *Learner module* but also some of the evidence leading to that conclusion. A subsequent system or training application, coordinating with the *Pedagogical module* and *Domain module*, could then judge whether the conclusion is likely to obtain, given evidence.

Contrast these factors with what could plausibly be transferable assessments of other factors like learner interest areas, assuming such factors are relatively stable over time. Nevertheless, a great deal of factor-specific research across ITS platform, educational games, simulation-based training environments, and others is required to determine those factors (and at what level of granularity) that will allow for transferable inferences across training applications and at what level of granularity these factors are sufficiently "domain-independent" to be tracked by GIFT's *Learner module*. The type of factor-specific research described above (though only in the CT and MATHia platforms) and also pursued by a wide variety of researchers in the educational data mining community and elsewhere is a step in the right direction (cf. the brief review for ITSs like CT in Fancsali, Ritter, Stamper & Nixon, 2013), but determining how findings transfer across domains and training applications is vital.

This problem is not limited to non-cognitive factors: Even certain cognitive factors like low-level skill models may not transfer from one ITS for mathematics to another. For example, the types of errors that students can make in a desktop version of the CT's equation solver units are different than those that can occur in a tablet-based version of these units we are currently implementing as a part of the HPIT project that will use handwriting recognition. This has important implications for the types of competencies and/or skills that can be usefully tracked by the *Learner module* and the *Persistent Learner Model* in GIFT. Finding appropriate levels of granularity (i.e., grain sizes) for various factors that must be measured, assessed, and tracked by systems with ambitious goals like HPIT and GIFT is paramount to delivering intelligent tutoring in a variety of educational and training contexts.

References

Ames, C. (1992). Classrooms: goals, structures, and student motivation. *Journal of Educational Psychology, 84*(3), 261 – 271.

Anand, P.G. & Ross, S.M. (1987). Using computer-assisted instruction to personalize arithmetic for elementary school children. *Journal of Educational Psychology, 79*(1), 72 – 78.

Baker, R.S.J.d. (2007). Modeling and understanding students' off-task behavior in intelligent tutoring systems. In *Proceedings of the 2007 Conference on Human Factors in Computing Systems* (pp.1059 – 1068). ACM: New York.

Baker, R.S.J.d., de Carvalho, A. M. J. A. (2008). Labeling student behavior faster and more precisely with text replays. In R.S.J.d. Baker, T. Barnes & J.E. Beck (Eds.), *Proceedings of the 1st International Conference on Educational Data Mining* (pp. 38 – 47). International Educational Data Mining Society.

Bandura, A. (1994). Regulative function of perceived self-efficacy. In M. G. Rumsey, C. B. Walker & J. H. Harris (Eds.), *Personal selection and classification* (pp. 261 – 271). Hillsdale, NJ: Erlbaum.

Bandura, A. (1997). *Self-efficacy: The exercise of control*. New York: Freeman.

Bandura, A. (2006). Guide for constructing self-efficacy scales. In F. Pajares & T. Urdan (Eds.), *Self-Efficacy Beliefs for Adolescents* (pp. 307 - 377). IAP.

Barron, K. E. & Harackiewicz, J. M. (2001). Achievement goals and optimal motivation: testing multiple goal models. *Journal of Personality and Social Psychology, 80*, 706 – 722.

Bernacki, M.L., Nokes-Malach, T.J. & Aleven, V. (2012, April). Investigating stability and change in unit-level achievement goals and their effects on math learning with intelligent tutors. Paper presented at Annual Meeting of the American Educational Research Association, Vancouver, Canada.

Bernacki, M.L., Nokes-Malach, T.J. & Aleven, V. (2013). Fine-grained assessment of motivation over long periods of learning with an intelligent tutoring system: methodology, advantages, and preliminary results. In R. Azevedo & V. Aleven (Eds.), *International handbook of metacognition and learning technologies* (pp. 629 – 644). New York: Springer.

Bollen, K. (1989). *Structural Equations with Latent Variables*. Wiley.

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.

Cordova, D.I. & Lepper, M.R. (1996). Intrinsic motivation and the process of learning: Beneficial effects of contextualization, personalization, and choice. *Journal of Educational Psychology, 88*(4), 715 – 730.

Dweck, C.S. (1986). Motivational processes affecting learning. *American Psychologist, 41*(10), 1040 – 1048.

Elliot, A.J. (1999). Approach and avoidance motivation and achievement goals. *Educational Psychologist, 34*(3), 169 – 189.

Elliot, A.J. & McGregor, H.A. (2001). A 2X2 achievement goal framework. *Journal of Personality and Social Psychology, 80*(3), 501 – 519.

Elliot, A.J. & Murayama, K. (2008). On the measurement of achievement goals: critique, illustration, and application. *Journal of Educational Psychology, 100*(3), 613 – 628.

Fancsali, S.E., Bernacki, M.L., Nokes-Malach, T.J., Ritter, S. & Yudelson, M. (in press). Goal orientation, self-efficacy, and "online measures" in intelligent tutoring systems. In *Proceedings of the 36th Annual Meeting of the Cognitive Science Society*. Austin, TX: Cognitive Science Society.

Fancsali, S.E., Ritter, S. (2014). Context personalization, preferences, and performance in an intelligent tutoring system for middle school mathematics. In S. Teasley & A. Pardo (Eds.), *Proceedings of the 4th International Conference on Learning Analytics and Knowledge* (pp. 73 – 77). ACM: New York.

Fancsali, S.E., Ritter, S., Stamper, J., & Nixon T. (2013). Toward “hyper-personalized” Cognitive Tutors. In R.A. Sottilare & H.K. Holden (Eds.), *AIED 2013 Workshops Proceedings Volume 7: Recommendations for Authoring, Instructional Strategies and Analysis for Intelligent Tutoring Systems (ITS): Towards the Development of a Generalized Intelligent Framework for Tutoring (GIFT)* (pp. 71 – 79). Sun SITE Central Europe (CEUR).

Fryer, J.W. & Elliot, A.J. (2007). Stability and change in achievement goals. *Journal of Educational Psychology, 99*(4), 700 – 714.

Ku, H.Y. & Sullivan, H.J. (2002). Student performance and attitudes using personalized mathematics instruction. *Educational Technology Research and Development, 50*(1), 21 – 34.

Muis, K.R. & Edwards, O. (2009). Examining the stability of achievement goal orientation. *Contemporary Educational Psychology, 34*(4), 265 – 277.

Otieno, C., Schwonke, R., Salden, R. & Renkl, A. (2013). Can help seeking behavior in intelligent tutoring systems be used as an online measure for goal orientation? In M. Knauff, M. Pauen, N. Sebanz & I. Wachsmuth (Eds.), *Proceedings of the 35th Annual Meeting of the Cognitive Science Society* (pp. 1103 – 1108). Austin, TX: Cognitive Science Society.

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*. Advance online publication. doi: 10.3102/0162373713507480.

Richardson, J.T.E., (2004). Methodological issues in questionnaire-based research on student learning in higher education. *Educational Psychology Review, 16*(4), 347 – 358.

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

Ritter, S., Joshi, A., Fancsali, S.E. & Nixon, T. (2013). Predicting standardized test scores from Cognitive Tutor interactions. In S. K. D’Mello, R. A. Calvo & A. Olney (Eds.), *Proceedings of the 6th International Conference on Educational Data Mining* (pp. 169 – 176). International Educational Data Mining Society.

Ritter, S., Sinatra, A.M. & Fancsali, S.E. (in press). Personalized content in intelligent tutoring systems. In *Design recommendations for adaptive intelligent tutoring systems: Volume 2 - adaptive instructional strategies*. Orlando, FL: U.S. Army Research Laboratory.

Senko, C., Hulleman, C.S. & Harackiewicz, J.M. (2011). Achievement goal theory at the crossroads: Old controversies, current challenges, and new directions. *Educational Psychologist, 46*(1), 26 – 47.

Spirtes, P., Glymour, C. & Scheines, R. (2000). *Causation, Prediction, and Search*. 2nd Edition. Cambridge, MA: MIT.

Walkington, C.A. (2013). Using adaptive learning technologies to personalize instruction to student interests: The impact of relevant contexts on performance learning outcomes. *Journal of Educational Psychology*, *105*(4), 932 – 945.

It’s All About the Process: Building Sensor-Driven Emotion Detectors with GIFT

Jonathan P. Rowe, Bradford W. Mott, James C. Lester
North Carolina State University

Introduction

Over the past decade, there have been major advances in research on affective computing (Calvo & D’Mello, 2010). In education and training, computational models of affect are widely recognized as integral parts of adaptive learning technologies, and their promise has been demonstrated in intelligent tutoring systems (ITSs) that model students’ affective states (Conati & Maclaren, 2009), model virtual agents’ affective behaviors (Marsella & Gratch, 2009), and detect student motivation and engagement (Forbes-Riley & Litman, 2013). Considerable work on affective computing has sought to increase the fidelity with which affective processes are understood and utilized in learning environments. Equally important is work on generalized models of affect to be used across a range of educational settings, populations, and subject areas.

In this paper, we describe operational details of an ongoing project to create multi-channel computational models of affect and engagement with the Generalized Intelligent Framework for Tutoring (GIFT). We describe the process of collecting and analyzing a corpus of affect sensor data for inducing emotion detector models in a serious game for tactical combat casualty care. We provide details about the research study conducted to collect a corpus of affect and learning data during the first phase of the project. Further, we describe the dataset produced by the study, as well as the data analysis pipeline that has been implemented for building and evaluating sensor-based emotion detection models. As the work is still in-progress, we conclude by discussing lessons learned from the data collection and analysis conducted thus far, as well as recommendations for enhancements to GIFT, to support members of the user community interested in computational models of affect.

Affect Corpus Collection

To illustrate how GIFT is used to collect affect sensor data, we describe a study conducted with a population of U.S. Army trainees. The study was part of a collaborative project between North Carolina State University, Teachers College Columbia University and the US Army Research Laboratory to create computational models of affect and engagement. The collected data serves as a corpus for machine learning emotion detection models that will ultimately be integrated with GIFT.

During the study, all participants completed the same training module. The training module focused on a subset of skills for tactical combat casualty care: care under fire, hemorrhage control, and tactical field care. The study materials, including pre-tests, training materials, and post-tests, were administered through GIFT. At the onset of each study session, learners completed a brief demographic questionnaire and content pre-test on tactical combat casualty care. Afterward, participants were presented with a

PowerPoint presentation about tactical combat casualty care. After completing the PowerPoint, participants completed a series of training scenarios in the vMedic serious game environment where they applied skills and procedures presented in the PowerPoint. In vMedic, the learner adopts the role of a combat medic faced with a situation where one (or several) of her fellow soldiers has been seriously injured. The learner is responsible for properly treating and evacuating the casualty. After the vMedic training scenarios, participants completed a post-test, which included the same content assessment items as the pre-test.

During the study, ten separate research stations were configured to collect data from multiple participants simultaneously (Figure 1). Each station consisted of an Alienware laptop, a Microsoft Kinect for Windows sensor, and an Affectiva Q-Sensor, as well as a mouse and pair of headphones. The study room layout is shown in Figure 1. Participant stations are denoted as ovals with assigned station letters. Red cones show the locations of Microsoft Kinect sensors, as well as the sensors' approximate fields-of-view. The dashed line denotes the walking path for the field observers. Participants' learning behaviors in both PowerPoint and vMedic were logged by GIFT, writing all events to text files stored on the Alienware laptops' hard disks. Participants' pre- and post-test responses were also logged to these files.

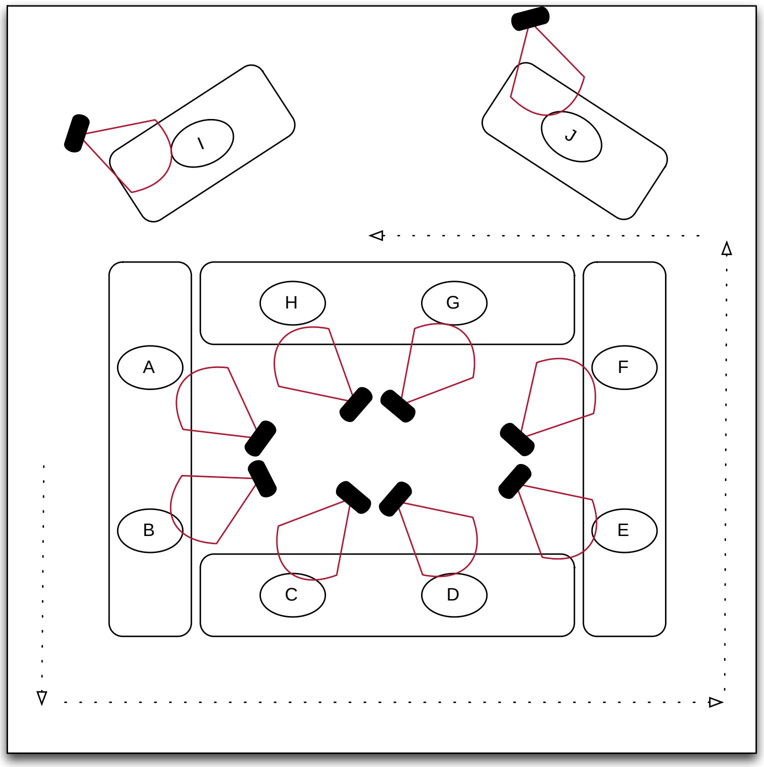


Figure 1. Study room layout.

As participants completed the study materials, a pair of field observers regularly recorded participants' physical displays of emotion using hand-held Android devices running the HART field observation software. The field observers followed an observation protocol developed by Baker, D'Mello, Rodrigo, and Graesser (2010), in which observers walked around the perimeter of the study room, discreetly recording observations of each participant's affect in a round robin sequence. The field observers' walking path is shown with a dashed line in Figure 1. The HART software application enabled the field observers to select an emotional state that most closely corresponded to the participant's displayed state at that time. The seven candidate emotional states included concentration, confusion, boredom, surprise, frustration, contempt, and other. In addition to emotional states, field observers recorded participants' learning behaviors during every observation. Categories of learning behavior included on-task, off-task, and without thinking fastidiously. Each observation was marked with a timestamp and written to a text file residing on the field observer's Android device. In order to ensure synchronization between the field observations' timestamps and the GIFT timestamps assigned to the sensor and log data, all devices were synced to the same time-server prior to each session.

Kinect sensors recorded participants' physical behavior during the study, including facial expressions, posture shifts and hand gestures. Each Kinect sensor was mounted on a tripod and positioned in front of a participant (Figure 1). The Kinect integration with GIFT provided four data channels: skeleton tracking, face tracking, RGB (i.e., color), and depth data. The first two channels leveraged built-in tracking algorithms (which are included with the Microsoft Kinect for Windows SDK) for recognizing a user's skeleton and face, each represented as a collection of 3D vertex coordinates. The RGB channel is a 640x480 color image stream comparable to a standard web camera. The depth channel is a 640x480 IR-based image stream depicting distances between objects and the sensor.

Q-Sensors recorded participants' physiological responses to events during the study. The Q-Sensor is a wearable arm bracelet that measures participants' electrodermal activity (i.e., skin conductance), skin temperature, and its orientation through a built-in 3-axis accelerometer. The wireless sensor collected data at 10Hz, and was primarily used for real-time arousal detection. A Q-sensor was worn on each participant's left wrist. Sensors were calibrated at the beginning of each study session, immediately before participants completed the study's pre-test measures. The calibration process took no more than a few minutes for each session.

Affect-Sensor Data Analysis Pipeline

To produce an automated, reproducible data analysis pipeline for machine learning emotion detection models from raw study data, a collection of Python scripts were written to automate data cleaning, filtering, integration and distillation phases of analysis. This pipeline is implemented separately from GIFT. With this infrastructure in place, all stages of data analysis can be reproduced, inspected, and validated by members of the research team, as well as members of the outside research community. Currently, the implemented data analysis infrastructure has focused on integrating Kinect and HART data, with a particular emphasis on encoding shifts in posture and their relation to observed emotional states.

Affect-Sensor Dataset

Different data channels in the corpus were encoded in different forms. GIFT log data from PowerPoint and TC3Sim was encoded as a series of text-based JSON messages distributed across multiple text files marked with timestamps. These text files also contained participants' pre-test and post-test responses, which were encoded as JSON messages as well. In general, there was one text log file for each participant present during the study. However, if the software crashed or required a restart, then an additional log file was generated.

Kinect sensor data was stored in four different types of files. Face and skeleton mesh data were recorded in csv format. In these files, each row corresponded to a single observation from the Kinect sensor. The first column contained timestamps for the observations. Subsequent columns contained the 3D coordinates for all vertices tracked in the skeleton and face meshes. In total, 91 vertices were tracked, with the majority of vertices originating from participants' faces. Kinect data was recorded at 10-12 Hz, or 10-12 rows per second. Two types of Kinect csv files were generated by GIFT: unfilteredKinect and filteredKinect files. UnfilteredKinect files included position data for every vertex tracked by the Kinect. FilteredKinect files contained position data for a select subset of tracked vertices.[1] The Kinect csv files did not include explicit information about participants' identifiers, station identifiers, or session numbers. However, the start time for each session (i.e., when GIFT was launched on the participant's laptop) was included in the filename for each csv file. This start time can be used to recover the session number for each file.

In addition to face and skeleton mesh data, two types of dat files were generated by GIFT's Kinect module: binary-encoded files that contain raw RGB channel data and depth channel data. These two types of files were separate from one another, and they quickly grew very large: a single hour-long session produced tens of gigabytes of compressed RGB and depth data. Both types of data were compressed using LZ4 compression, and they were written directly to external hard disks in order to save the study laptops' hard drive storage space. The GiftKinectDecoder, included in all versions of GIFT since v3.0, is necessary to decompress and transform this data for analysis. At present, analysis of raw RGB and depth channel data is not part of our automated data analysis pipeline.

Q-Sensor data recorded by GIFT was stored in csv files. Observations were recorded at approximately 10 Hz for the duration of each session. Among the columns, the first held timestamp information. Subsequent columns stored information about skin temperature, electrodermal activity, and the sensor's 3D coordinates measured by the built-in accelerometer. The QSensor data files also did not include explicit information about participants' identifiers, station identifiers, or session numbers. However, the start time for each session (i.e., the time that GIFT was launched on the participant's laptop computer) was included in the filename for each csv file.

Field observation data from the HART app was stored in csv format. One file was generated for each study session, and each file encompassed field observation data for every participant in the session. The files included a multi-line preamble denoting the coder's name, start time, emotion set, behavior set, and related meta-information. After the preamble, each row corresponded to a single observation of a participant. The columns contained information about participants' identifiers, observation timestamps, ob-

[1] An error in the version of GIFT software used during the study introduced column-alignment errors in the filteredKinect files. Consequently, the project's data analysis pipeline is based on unfilteredKinect files.

served learning behaviors, and observed emotional states, respectively. Separate sets of files were produced for each field observer.

Data Cleaning

The first stage of data analysis was cleaning: removing unnecessary files and header information from the dataset that did not comprise actual participant data. Empty log files corresponding to test runs and false starts were removed. Header information was parsed to determine the contents of each file for use in subsequent stages of the pipeline, and this information was separated from the participant data.

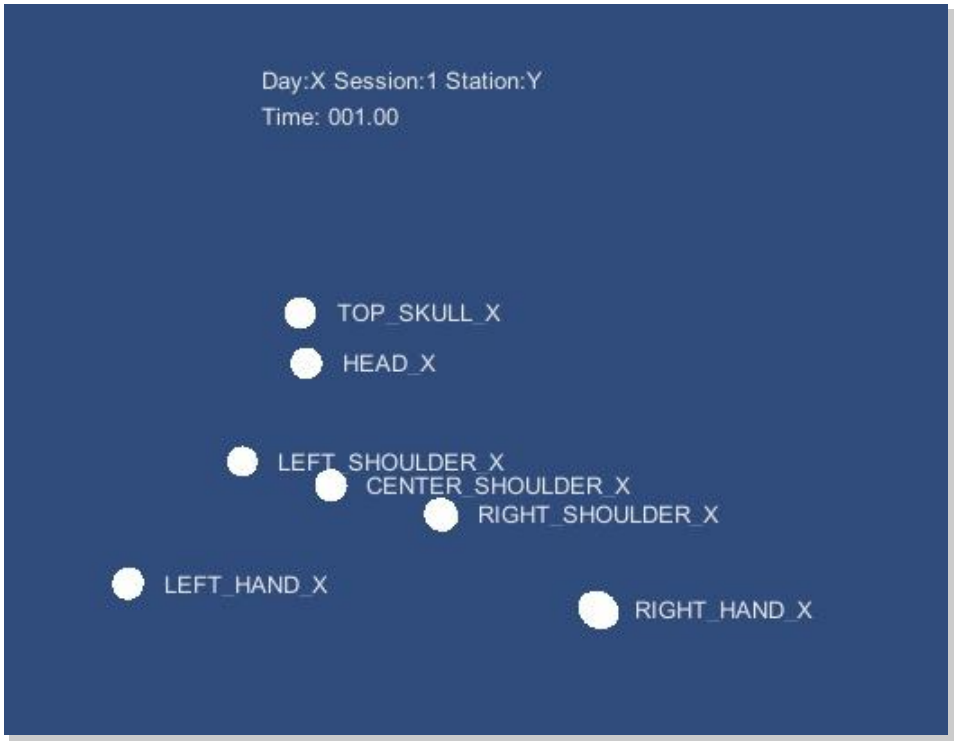


Figure 2. Screenshot from Kinect visualization tool.

Anomalies in the dataset were also removed during cleaning. Upon closely inspecting our corpus, it was apparent that sudden jumps in the coordinates of posture-related vertices occasionally appeared across adjacent frames. Notably, these jumps were much larger than typically observed across frames, and they occurred due to an issue with the way GIFT logged tracked skeletons: recording the most recent detected skeleton, rather than the nearest detected skeleton. If a bystander comes into the Kinect's field-of-view, and her skeleton is recognized by the Kinect, GIFT will log the bystander's skeleton rather than the participant's skeleton. This leads to a sudden jump in the coordinates of nearly every vertex in the tracked skeleton. In our case, this occurred whenever the Kinect detected a field observer behind the participant. Unfortunately, this issue was a symptom of the room configuration (Figure 1) employed for the data collection.

In order to investigate the skeleton mistracking issue, a prototype Kinect visualization tool was implemented in Unity3D (Figure 2). The tool reads and parses a filtered Kinect data file and then replays the session by rendering small white spheres corresponding to skeletal vertices at their appropriate locations in 3D space over time. Each sphere was labeled with its associated vertex name, enabling the posture and behavior of a participant to be visually inspected using only the Kinect mesh data.

To identify observations that corresponded to field observers rather than participants, Euclidean distances between subsequent observations of a central vertex were calculated. The distribution of Euclidean distances was plotted to inspect the distribution of between- frame movements of the vertex. If the Kinect tracked field observers, who were physically located several feet behind participants, the distribution was likely to be bimodal. In this case, one cluster would correspond to regular posture shifts of a participant between frames, and the other cluster corresponded to shifts between tracking participants and field observers. This distribution could be used to identify a distance threshold for determining which observations should be thrown out, as they were likely due to tracking field observers rather than participants.

Data Filtering

After cleaning, a filtering process was performed to remove data that were unnecessary for planned analyses. For example, in analyses of Kinect data investigating participants' posture shifts, a majority of face mesh vertices recorded by the Kinect were not necessary. Among the 91 vertices recorded in the unfilteredKinect log files, we preserved only seven for posture analysis: top_skull, head, center_shoulder, left_shoulder, right_shoulder, left_hand, and right_hand. These vertices were selected based on prior findings about postural indicators of emotion measured by Kinect sensors (Grafsgaard, Boyer, Wiebe, & Lester, 2012). Although the face mesh vertices were removed from the perspective of downstream analysis, they remained in the raw dataset; they can easily be re-added to the pipeline at a later time if analyses of participant hand gestures and facial behavior are undertaken.

Data Integration

GIFT's Kinect data files do not include explicit records of participant identifiers. However, determining the unique identifier for each log is critical for integrating the Kinect data with other data channels, such as field observation data or Q-Sensor data. By contrast, HART data files do include unique identifier information. In our dataset, every field observation included an explicit record of the participant's unique identifier as the first attribute in the row. Participants' unique identifiers consisted of a combination of study day, session number, and station letter, and fortunately this information could be recovered from start time information stored in each Kinect log's filename, along with station identifier information denoted in the directory structure of the raw dataset. For the most part, the participant identifier recovery process could be automated. However, some sessions started several minutes early due to early arrivals of participants, making it more challenging to detect session boundaries automatically. These exceptional cases were handled case-by-case.

After both the Kinect and HART data were cleaned and filtered, and unique participant identifiers had been assigned to each observation in the dataset, the two data sources were integrated. To perform this integration, the timestamps for each observation across the two data sources were aligned. Even though both HART and GIFT were synchronized to the same time-server during data collection, they were generally launched at different times. Consequently, the timestamps for each observation could not immediately be aligned, because they denoted relative time elapsed since application start, not clock time. Therefore, an offset had to be calculated between the two data sources' start times in order to establish a shared time scale. After a shared time scale was established, and the timestamps for each observation were adjusted, the two data channels could be integrated.

To perform the integration, the HART dataset's columns were appended to the Kinect dataset's columns, and field observations were interspersed with the Kinect data at the appropriate timestamps. The HART columns included observations from both field observers involved in the study. This integration produced a dataset that consisted of long series of Kinect observations, which include blank values for the HART columns, and occasional instances of field observations that included blank values for the Kinect columns. In rare cases where both a Kinect observation and HART observation occurred during the same shared timestamp, both Kinect and HART values were contained in the data vector.

Feature Distillation

After the data sources were integrated, predictor features were derived from the Kinect data for association with the field observations. Long series of Kinect observations were distilled into one-dimensional vectors of Kinect summary statistics, which were then prepended to vectors of field observation data. Afterward, rather than having many instances of Kinect observations for each field observation, the distilled dataset consisted of a series of one-dimensional vectors with Kinect summary statistics and field observation attributes. In machine learning terms, the Kinect features served as predictor features, and the field observations served as class labels.

Currently, the engineered predictor features are inspired by related work in the affective computing research literature (Dragon et al., 2008; D'Mello & Graesser, 2010; Grafsgaard et al., 2012). To date, much of the work investigating posture as a predictor of learners' emotions has leveraged pressure-sensitive chairs (Dragon et al., 2008; D'Mello & Graesser, 2010). More recently, Grafsgaard and colleagues have used Kinect to investigate posture and hand gestures as predictors of self-reported engagement during tutorial dialogues (2012). Several research groups have converged on common sets of postural indicators of emotional states. For example, in several cases boredom has been found to be associated with leaning backward, as well as increases in posture variance (Grafsgaard et al., 2012; D'Mello & Graesser, 2010). Conversely, confusion and flow have been found to be associated with forward-leaning behavior (Grafsgaard et al., 2012; D'Mello & Graesser, 2010). Given these findings, we have begun to identify analogous predictor features that can be generated from our dataset. Specifically, we calculate depth statistics characterizing each skeleton vertex's distance from the camera. Distilled features for each of the vertices include the following:

- Most recently observed distance,
- Minimum observed distance observed thus far,

- Maximum observed distance observed thus far,
- Median observed distance observed thus far,
- Variance in distance observed thus far,
- Median observed distance during the past five seconds,
- Variance in distance during the past five seconds,
- Median observed distance during the past twenty seconds,
- Variance in distance during the past twenty seconds,
- Median observed distance during the past forty seconds,
- Variance in distance during the past forty seconds,
- Median observed distance since the last emotion observation
- Variance in distance since the last emotion observation.

From just these features, 91 predictors are available for training emotion detection models. Of course, additional predictor features can be engineered as well. Afterward, the dataset is written to disk to serve as an intermediate representation for analysis and modeling. Alternatively, the dataset can be provided directly to a machine-learning toolkit for model training and validation.

Emotion Detector Modeling

Rather than attempt to train a classifier capable of discriminating among all seven emotion states, we instead train seven binary classifiers that discriminated between the occurrence and non-occurrence of each emotion. In order to train and validate these classifiers, we produce seven separate datasets, one for each emotion in the corpus. The emotion labels from both field observers are used as separate class labels for these datasets. The emotion labels are recoded into binary variables that indicate whether the target emotion is observed or whether some other emotional state is observed. This approach mirrors techniques used in related work with HART-generated affect data (Baker et al., 2012). Given the project's goal of eventually integrating the emotion detectors with GIFT, a hierarchical classification scheme leveraging these binary classifiers at the base level will likely be employed (D'Mello & Graesser, 2009).

In order to train and validate emotion detection models from the corpus, we are using the Waikato Environment for Knowledge Analysis (WEKA), a popular open-source toolkit that provides a collection of machine-learning algorithms and analysis tools (Hall et al., 2009). Prior to machine learning, our distilled datasets are reformatted and written in ARFF format, the standard data format for WEKA. Next, the dataset is processed by a short Java program that divides the corpus into 10 independent folds for cross-validation. These folds are used to train and test several emotion detection models using a range of classification algorithms, including J48 decision trees, naïve Bayes, support vector machines, logistic regression, JRIP, k-nearest neighbor, and neural networks. Feature selection is performed using forward selection techniques. The detector models' performance is compared using Cohen's kappa and A' (area under the ROC curve). Different combinations of classification techniques, parameter settings, predictor features, and field observer emotion labels are compared to identify the most effective emotion detection models for integration with GIFT. At the time of writing this process is ongoing, as we are currently investigating which predictor features and modeling techniques are most effective.

Recommendations and Lessons Learned

We have described a collaborative project between North Carolina State University, Teachers College Columbia University, and the US Army Research Lab to investigate computational models of affect and engagement for a serious game on tactical combat casualty care. Although the project is ongoing, we have described operational details of a study to collect a corpus of affect sensor data for inducing emotion detection models, as well as the data analysis pipeline to clean, filter, integrate, distill, analyze, and model the dataset. It is our hope that by sharing details of the study setup, dataset, and data analysis process, we have provided a useful account for other GIFT users interested in investigating generalizable models of affect and engagement.

Through our experiences collecting and analyzing affect sensor data with GIFT, we have learned several lessons that will inform our subsequent data collections. First, it is clear that seemingly innocuous details of a research study, such as the physical configuration of a room, can have significant implications for the dataset that is collected, as well as the data analysis process that will follow. In our case, considerable time and effort has been dedicated to understanding and resolving the Kinect mistracking issue, which was a consequence of our particular study room configuration and idiosyncrasies in the way GIFT records Kinect data. Furthermore, we have been reminded that any shortcoming in pilot testing in advance of a research study creates openings for surprises and headaches later on. Although our team ran several pilot tests with individual participants before the study, we overlooked running pilot tests with bystanders walking behind participants to simulate actual study conditions, a factor that contributed to the skeleton mistracking issue.

Fortunately, these oversights have not adversely affected the corpus that we collected in any major capacity, and most of the challenges we encountered have been addressable through automated techniques implemented in the data analysis pipeline. As the project progresses, we will continue to investigate alternate combinations of predictor features, machine learning algorithms, parameter values, and field observations to devise effective emotion detection models for integration with GIFT, and we will report these findings to the research community as they emerge. In addition, we should note several promising areas for implementing future enhancements to GIFT to support affective computing research. At a low level, logging participants' identifier information in sensor data files would significantly reduce the burden of recovering identifier information from logs' timestamp information. At a higher level, providing tools for automatically integrating data channels and exploring sensor data would significantly streamline research on affective computing with GIFT. Currently, these types of functionality must be implemented separately from GIFT on a project-by-project basis. Although cleaning, filtering, distilling, and modeling data will likely best be performed outside of GIFT, phases that can be standardized across datasets—such as data integration and exploration—are likely to prove useful for the GIFT user community.

Acknowledgements

The authors wish to thank the Army Research Lab for supporting this research. Additionally, we wish to thank our collaborators, Ryan Baker and Jeanine DeFalco at Teacher's College Columbia University, and Bob Sottilare, Keith Brawner, Benjamin Goldberg, and Heather Holden from ARL.

References

Baker, R., D'Mello, S., Rodrigo, M. 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*(4), 223–241.

Baker, R., Gowda, S., Wixon, M., Kalka, J., Wagner, A. Z., Salvi, A., Aleven, V., Kusbit, G., Ocumpaugh, J., & Rossi, L. (2012). Towards Sensor-Free Affect Detection in Cognitive Tutor Algebra. *Proceedings of the 5th International Conference on Educational Data Mining*, Chania, Greece, 126–133.

Calvo, R. A., D'Mello, S. K. (2010). Affect Detection: An Interdisciplinary Review of Models, Methods, and their Applications. *IEEE Transactions on Affective Computing*, *1*(1), 18–37.

Conati, C., & Maclaren, H. (2009). Empirically Building and Evaluating a Probabilistic Model of User Affect. *User Modeling and User-Adapted Interaction*, *19*(3), 267–303.

D'Mello, S., Graesser, A. (2010). Mining Bodily Patterns of Affective Experience during Learning. *Proceedings of the 3rd International Conference on Educational Data Mining*, Pittsburgh, Pennsylvania, 31–40.

D'Mello, S., & Graesser, A. (2009). Automatic detection of learner's affect from gross body language. *Applied Artificial Intelligence*. *23*(2), 123-150.

Dragon, T., Arroyo, I., Woolf, B., Burleson, W., El Kaliouby, R., & Eydgahi, H. (2008). Viewing student affect and learning through classroom observation and physical sensors. *Proceedings of the 20th International Conference on Intelligent Tutoring Systems*, Montreal, QC, Canada, 29–39.

Forbes-Riley, K., & Litman, D. (2013). When Does Disengagement Correlate with Performance in Spoken Dialog Computer Tutoring? *International Journal of Artificial Intelligence in Education*, *22*(1-2), 39-58.

Grafsgaard, J., Boyer, K., Wiebe, E., & Lester, J. (2012). Analyzing Posture and Affect in Task-Oriented Tutoring. *Proceedings of the Twenty-Fifth Florida Artificial Intelligence Research Society Conference*, Marco Island, Florida, 438–443.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten , I. (2009). The WEKA Data Mining Software: An Update. *SIGKDD Explorations*. *11*(1), 10–18.

Marsella, S. C., & Gratch, J. (2009). EMA: A Process Model of Appraisal Dynamics. *Cognitive Systems Research*, *10*(1), 70–90.

Theme V: Teachable Agents

Design Challenges And Recommendations For Multi-Agent Learning Systems Featuring Teachable Agents

Natalie Parde and Rodney D. Nielsen
University of North Texas

Introduction

As we continue to progress in the development of effective Intelligent Tutoring Systems (ITSs), it is important to design systems that facilitate more comprehensive, supportive, and natural learning environments (Sottilare & Holden, 2013). One way to do this is to incorporate multiple intelligent agent roles into the system, thus offering the opportunity for greater and more varied learning support. Of particular and novel interest are systems that include both a teachable agent and a virtual tutor. Teachable agents are pedagogical agents in which the human learner assumes the role of the teacher and the agent assumes the role of the student. Systems using these agents have been shown to increase both learning gains and motivation in students, leading to greater reasoning skills even when the student is no longer using the system (Biswas et al., 2005). However, evidence indicates that students who are insufficiently prepared to teach a subject might benefit more from tutored problem solving (Matsuda et al., 2011), such as seen in traditional ITSs like AutoTutor (D'Mello & Graesser, 2012).

Therefore, merging teachable agents with traditional ITSs to form multi-agent systems featuring both virtual tutor and teachable agent roles poses an ideal way to build a medium between the two systems, in turn, realizing the benefits of both. Such systems can offer a more comprehensive mode of initially teaching students than is currently seen in teachable agent systems, while still capitalizing on their motivational benefits and increased learning gains, ultimately creating a stronger overall tutoring system.

Here the expected advantages of multi-agent learning systems of this nature are detailed, as well as some of the design challenges that arise in creating these systems. Recommendations for accommodating these design challenges are provided, and additionally some example instructional strategies that a teachable agent in such a multi-agent system could use to exact higher, longer-lasting learning gains are proposed. Finally, the conclusion discusses how the Generalized Intelligent Framework for Tutoring (GIFT) could be used as a framework for this type of multi-agent system.

Expected Advantages of the Proposed Learning System

ITSs in general have already proven to be an invaluable learning aid in a diverse array of learning contexts, including academic, professional, and military domains. In fact, they have been shown to perform nearly as effectively as human one-on-one tutoring (VanLehn, 2011) and have been successfully used as classroom aids for more than fifteen years (Koedinger et al., 1997). Many existing ITSs have featured a single virtual tutor agent, but with growing frequency developers have demonstrated an interest in moving beyond this model, to the increased potential for personalization and realistic classroom scenarios offered by multi-agent systems. This trend indicates that to most effectively enable the system to evolve and adapt to a wide variety of learner requirements, an ideal cast of pedagogical agent roles should be designed to realistically simulate the multiples roles present in a physical learning environment.

Many other agent roles besides the virtual tutor have been explored already in pedagogical systems, particularly in Serious Games (Prensky, 2005) and story-based learning environments. Agent roles have included the learning companion (Chou et al., 2003), teachable agent (Blair et al., 2007), collaborator agent (Goodman et al., 1997), and critic agent (Wißner et al., 2012), among others. The potential benefits offered by these different pedagogical agents are vast; they can be used to promote deeper learning (Moreno et al., 2001), increase student motivation (Lester et al., 1997), and even improve metacognitive skills (McNamara et al., 2007). One way to harness both the documented benefits of ITSs and the motivational advantages offered by weaker pedagogical agents while also offering the potential to sustain the learning environments currently supported by either is to pair an intelligent tutor with a teachable agent in a multi-agent learning system, either as two externally separate agents or as two agents embodied as a single peer agent. Some specific advantages to doing this are outlined in the following sub-sections.

The advantages of varied pedagogical agent knowledge levels

Before discussing the advantages of using pedagogical roles of varying knowledge levels in learning systems, it is beneficial to define the terms "strong agent" and "weak agent" as they are used in this work. A strong agent here is defined as an agent with a high externally evident knowledge level that can be relied upon for correctness—thus, a strong agent makes no errors. A weak agent, on the other hand, is prone to error and likewise possesses a low externally evident knowledge level. Thus, a virtual tutor can be seen as a strong agent and a teachable agent can be seen as a weak agent. A major expected advantage from using both strong and weak agents is that the system is likely to appeal to a broader range of learners. In fact, previous studies have indicated that different types of students prefer different types of pedagogical agents. For instance, students who are introverted or more capable have empirically been shown to prefer strong agents, whereas students who are extraverted or less capable have in contrast been shown to prefer weak agents (Hietala & Niemirepo, 1998). Furthermore, when examining the relationship between learning competency and agent competency, Kim (2007) found that strong students learned better with strong agents and weak students learned better with weak agents.

In addition to satisfying a larger number of learner preferences and needs, research has indicated that students simply find learning easier when they use systems featuring multiple agent roles and knowledge levels (Baylor & Ebbers, 2003). One explanation for this is that the ability for students to attribute different learning tasks (such as teaching or learning information for the first time) to different agent roles may help them to reduce their cognitive load. Regardless of whether or not this is true, the existence of both strong and weak agent roles in learning systems serves to satisfy a greater amount of students' learning needs by allowing for more diverse pedagogical interaction.

The expected advantages offered by teachable agents

Including teachable agents in learning systems not only offers the advantages correlated with weaker agents in general, but also can be expected to increase student motivation and learning gains. One reason why this can be expected is that by their nature, teachable agents require active rather than passive learning—the act of teaching requires a concerted effort on the part of the user to organize learned knowledge and understand it more deeply in order to be able to teach it themselves. The advantages of this have been empirically demonstrated: in a study comparing students using a teachable agent system and those using an intelligent tutoring version of the same system, students in the teachable agent condition experienced higher motivation and learning gains than their counterparts (Leelawong & Biswas, 2008). Teachable agents have also been shown to be particularly beneficial with lower-achieving students (Chase et al., 2009). In fact, the use of teachable agents has even shown promise for affecting long-term metacognitive change – students using the agents have demonstrated heightened reasoning skills even when they are no longer using the agents (Biswas et al., 2005).

The expected advantages of pairing teachable agents with virtual tutors

Teachable agents offer numerous expected advantages when included in learning systems, but that does not mean that virtual tutors should not play an important role as well. Many students prefer strong agent roles, as shown by Hietala and Niemirepo (1998), and moreover, virtual tutors should be present in learning systems to initially teach the requisite domain knowledge. Without first having a sufficient level of prior knowledge regarding the domain material, learning gains from using teachable agents could be limited, leading students to instead benefit more from traditional tutored problem solving (Matsuda et al., 2011). Thus, the virtual tutor should actually be a key role in the overall learning system, helping to make the system more comprehensive and effective. Without their help, students may fail to take full advantage of a teachable agent system and instead simply "game the system" to achieve correct answers. Shallow strategies such as these have been found to cause the long-term benefits of teachable agents to dwindle (Roscoe et al., 2013). Therefore, the greatest learning advantages will be seen when the system can both initially teach students using a virtual tutor and then later reinforce their learning by asking them to teach the information that they learned previously to a teachable agent.

Existing work

It is already common for multi-agent systems to include either a virtual tutor or a teachable agent. Multi-agent systems featuring at least one virtual tutor agent include MetaTutor (Azevedo et al., 2009), iStart (McNamara et al., 2007), Geometry Cognitive Tutor (Roll et al., 2011), and Wayang Outpost (Arroyo et al., 2011). The additional agent roles in these systems are primarily used to provide metacognitive support to students. Betty's Brain (Leelawong and Biswas, 2008) is a multi-agent system featuring a teachable agent, and the supplementary agent role in this system, Mr. Davis, provides metacognitive support. However, few systems to date include both virtual tutor roles and teachable agent roles. One system that does include both is DynaLearn, which provides separate agents playing the roles of a mechanic (to diagnose the learner's model), a critic (to provide feedback), and a quizmaster (who can quiz both the user and the teachable agent) in addition to a virtual tutor agent and teachable agent (Wißner et al., 2012). Another system that features both virtual tutor and teachable agent roles is Operation Aries! (Millis et al., 2011), which tailors a peer agent role according to learner strength. In Operation Aries!, learners engage in "trialogs" with Dr. Quinn (a teacher agent) and Glass (another student). When students are performing well, they teach Glass as Dr. Quinn observes, whereas when they are performing poorly they observe Dr. Quinn teaching Glass. Otherwise, Dr. Quinn teaches the student as Glass observes.

Design Challenges

Creating systems capable of accommodating teachable agents poses several unique design challenges, and even more challenges arise when the design of multi-agent systems including teachable agents is considered. This section first discusses the challenges unique to designing teachable agents, followed by a broader discussion of challenges that may materialize when designing multi-agent systems with one or more teachable agents.

Design challenges in creating teachable agents

One of the primary design differences between virtual tutors and teachable agents lies in their domain knowledge base. While a virtual tutor can easily function using only a static internal domain knowledge source, a teachable agent must also contain a dynamic knowledge base in order to accommodate for its growing externally evident "brain" of information. Depending upon the developer's needs and preferences, the amount of initial information that the teachable agent's dynamic knowledge base includes can vary greatly. One design challenge arises in determining what, exactly, the teachable agent should and should not know. To most closely mirror a human tutoring scenario, it should be assumed that the teachable agent has an adequate amount of prerequisite background knowledge, which is defined herein as knowledge not specifically within the agent's learning domain but that is still required to achieve a full understanding of the domain material. An example of this would be basic arithmetic knowledge when learning middle-school science – such a domain is almost certain to require at least some level of mathematical problem solving, but it does not specifically seek to assess or improve upon a student's knowledge of math.

Aside from prerequisite background knowledge, considerations must also be taken to determine how much prerequisite domain knowledge the teachable agent should have. An early teachable agent prototype, DENISE, proposed by (Nichols, 1994) featured an initial domain knowledge base that was entirely empty with no accompanying domain model. The complete absence of these components made it impossible for the designer to control the direction of the system's interactions, since the system could not determine whether or not it was being taught the proper subject material. A natural human tutoring environment typically operates under two assumptions:

- Tutoring is limited to a predefined domain
- The tutee has at least enough prerequisite knowledge to be able to formulate appropriate questions regarding the subject matter

To fulfill these requirements and to facilitate the teachable agent in a simulated tutoring environment, the teachable agent should have an expert domain model with which to compare its dynamic student-taught knowledge base. However, the teachable agent's student-taught knowledge base should be the only set of knowledge that is externally evident to the student.

Another design challenge with teachable agents lies in instructional strategy. Many previous strategies are incompatible with teachable agents, due to the agents' intentional naïveté. A teachable agent cannot offer didactic instruction, but must instead formulate dialogue to invisibly guide students toward reaching correct conclusions. In a later section, instructional methods suitable for teachable agents are discussed in detail.

Additional design challenges for multi-agent systems

One challenge of a standalone teachable agent system is that it requires students to gain an initial domain understanding elsewhere before they can teach material to their agents. Thus, the system must serve as a supplementary, rather than comprehensive, learning aid. Multi-agent systems can remedy this problem, but present some additional design challenges. For example, a system featuring both a teachable agent and a virtual tutor, and possibly even more agents, must allow for interactions between the various agents as well as with the user. This requires a centrally accessible dialogue context containing not only the information obtained from user interactions, but also from interactions between agents. Furthermore, each separate agent must have access to the appropriate pedagogical strategies for its respective role, and it is also beneficial for agents to have the ability to access one another's externally evident domain knowledge. For example, if the virtual tutor is asking the teachable agent to explain its reasoning regarding specific subject material, it should first ensure that the teachable agent has already been taught that information by the student to prohibit detracting from the realism of the learning environment.

One way to capitalize upon the benefits of a virtual tutor-teachable agent pairing without having to create the capability for external interaction between multiple agents, thus minimizing design challenges, is to internalize the pairing and encapsulate both roles within one greater role as a peer learning agent. A peer learning agent is defined herein as a non-authoritative pedagogical role designed to mimic a learner at a similar overall skill and knowledge level as the student, and thus having the ability to occasionally scaffold the student's knowledge by offering direct suggestions in addition to asking questions. A peer agent including both the virtual tutor and the teachable agent roles described in this paper could then alternate between tutor and tutee depending on learner preferences and perceived knowledge strength to select the best strategy for furthering each individual student's learning progress.

In a hypothetical interaction, the peer agent could function initially as a teachable agent, asking the student for help with a given subject, after a separate agent serving as an authoritative virtual tutor to the student has taught an adequate amount of background knowledge. In the event that the student began to struggle with the topic to the point that he or she no longer possessed sufficient knowledge to effectively teach it, the peer agent could take on aspects of a virtual tutor role to provide scaffolding hints, thereby guiding the student toward a deeper understanding of the topic or toward a solution to his or her problem. Once the student again demonstrated sufficient command of the subject, the peer agent could transition back to its original role as a teachable agent. The main design challenge of encapsulating the virtual tutor and teachable agent within a peer agent would lie in restructuring the pedagogical module to accommodate alternating roles.

Recommendations for Accommodating Design Challenges

The design challenges discussed in the previous section can be accommodated by adhering to several recommendations:

- Create a dynamic knowledge base for teachable agents to store their externally evident knowledge, as taught by the user

- Develop new instructional strategies meant to specifically take advantage of the unique features of teachable agent systems, as discussed in the following sections

- Allow cross-accessibility between all agents and the internal information stored in the system

This section shows how these recommendations can be accommodated in GIFT (Sottilare et al., 2012) by making some changes to its local tutoring processes. The local tutoring processes existing in GIFT include a sensor module, user module, pedagogical module, domain module, tutor-user interface, training application client, and sensors. Of these processes, the recommended changes would focus on the user module, pedagogical module, domain module, and tutor-user interface. Currently, the user module determines learners' states and tracks relevant learner trait and lesson data, and sends this information to the pedagogical module. The pedagogical module determines which instructional strategies should be used and sends this information to the domain module. The domain module defines and structures domain knowledge and chooses which specific content should be used in the specified instructional strategy. It then can return this information to the user module or send it onward to the tutor-user interface, with which the domain module has a bidirectional connection.

To store the externally evident, student-taught version of the teachable agent's knowledge base, a secondary domain module can be inserted in the pipeline at the same level as the original domain module. This new module, rather than the original domain module, would possess a bidirectional connection with the tutor-user interface. Information from both this module and the original domain module would feed into the user module. The data passed between this module and the other modules would primarily consist of triples of two concepts and a label describing the relationship between them. An example of this type of data could be a triple indicating the directional relationship between nucleotides and phosphate molecules, e.g., nucleotides (Concept A) contain (relationship label) phosphate molecules (Concept B). These triples would be sent to the user module to update its learner model, and to the tutor-user interface to create the content for the next dialogue act. Data passed to this new module from the pedagogical module would indicate the type of information (i.e., a triple containing a specific relationship role or target concept) that the module should then send to the tutor-user interface, and data passed to this module from the tutor-user interface following input from the human learner would be used to grow the module's externally evident knowledge base. An illustration of the modified architecture can be seen in Figure 1. The figure representing the original version of this GIFT architecture can be found in (Sottilare et al., 2012).

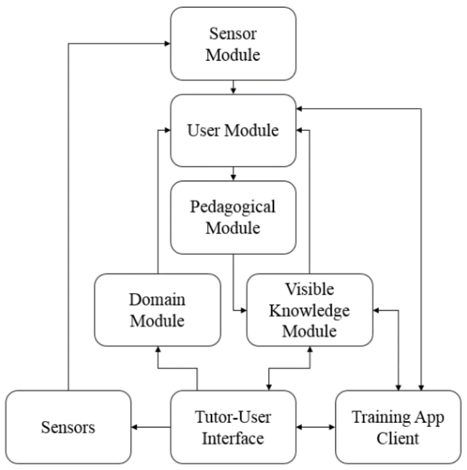


Figure 1: Local processes in the GIFT framework can be modified to accommodate teachable agents by inserting a visible knowledge module to show what has been taught by the student.

To additionally accommodate the integration of multiple agent roles, such as both a teachable agent and a virtual tutor agent, the pedagogical module can be used to determine which agent should interact with the user based on input from the user module indicating how closely the user's knowledge compares with the expert knowledge contained in the system's domain module. Students with a perceived domain knowledge level falling below a threshold set by the system's authors would receive additional prerequisite tutoring on the subject by a virtual tutor agent, whereas students perceived to have a higher aptitude for the subject would interact with a teachable agent instead to deepen their existing comprehension of the subject. The pedagogical module would additionally consider the differing anticipated learning and cognitive benefits of each of the strategies available to the agents in order to choose an optimal agent-strategy pairing. Ideally, the dual processes of agent selection and strategy selection would coincide, since a strategy's specific anticipated benefits could serve as a factor in selecting which agent to use, particularly for students with perceived domain knowledge levels near the system's default threshold. Alternatively, it may be that in some cases the user directly chooses to interact with a specific agent, in which case the pedagogical module could launch immediately to selecting a strategy available to that agent. Thus, the pedagogical module in a multi-agent system should include two integrated processes in order to determine the most beneficial agent-strategy pairing to be presented to the learner. A sample of this is seen in Figure 2. To support the two-process approach, the pedagogical module should implement policies to assess the user's current state and determine how his or her knowledge compares with the expert knowledge contained in the domain module, as well as his or her perceived learning rate, based on the number of misconceptions versus correct conceptions observed by the system. In more advanced implementations, the pedagogical module could also consider affective features such as positive or negative sentiment, and optimistic or pessimistic explanatory style in determining the appropriate agents and respective strategies to use.

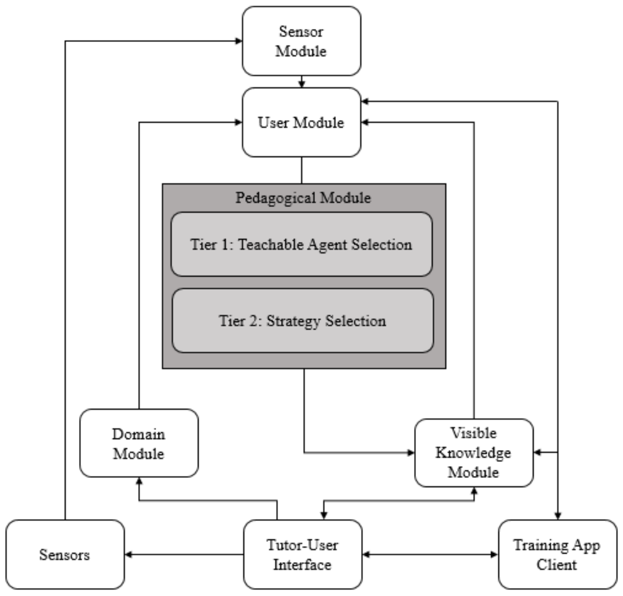


Figure 2: The pedagogical module can be modified to accommodate multiple agents by organizing its functions into two processes including both agent selection and strategy selection.

Instructional Strategies for Teachable Agents

Since a teachable agent offers an entirely different role from the teacher agent seen in traditional ITSs, it also presents the opportunity for different pedagogical strategies to be implemented and tested. In fact, many traditional pedagogical strategies are incompatible with the concept of a teachable agent, since they are used in ITSs to disseminate expertise presumably known to the agent but not the student. Teachable agents work on the opposite premise: they are used to draw out knowledge presumably known by the student (but possibly inaccessible until appropriate pedagogical strategies are used) that must be perceived as being "unknown" by the agent. To date, most teachable agent systems have relied upon asking the user to create a visual knowledge representation to teach the agent, and then having the teachable agent take a quiz so that its present knowledge of the subject can be evaluated. Although systems using this technique have proven functional, we believe the full potential for teachable agents to improve learning gains can be better realized with more interactive, dialogue-based strategies.

One obvious instructional strategy that fits perfectly within the dialogue capacity of teachable agents is Socratic questioning. Although Socratic questioning has previously been a strategy used with traditional ITSs (Graesser et al., 2001), teachable agents present the opportunity to create a new twist on the traditional Socratic question – the agents can ask the question from a convincingly naïve perspective. In traditional ITSs, although Socratic questions are open-ended, they are still posed by an authoritative teacher agent, and thus students retain an awareness that the question is directed and that their knowledge is being tested. On the other hand, the teachable agent is viewed by students as a peer without a predefined set of correct answers. As such, the prospect of answering believably authentic questions (cf. Otero & Graesser, 2001) is likely to serve as a better motivator, since students will feel less responsible for failure and less like they are being tested, as well as more responsible for their own role in the improvement of their agent's knowledge base.

To effect this strategy, questions can be generated based on the misconception types observed in students' concept maps. A taxonomy of different question types for use in question generation is described in (Nielsen et al., 2008) and in particular the question types found in (Collins, 1985) can be used to promote deeper reasoning in students who have demonstrated misconceptions in their knowledge of a subject. To create a training dataset for the system, a set of questions regarding concepts and their relations can be annotated according to which question type they fall under, and pattern-based question templates relevant to individual question types can then be extracted from this dataset via machine learning. Following this, the misconception type identified by the system can be used to determine which question templates to use, and the chosen question template can be filled with the appropriate concepts and relationships accordingly to pose a relevant question to the user. An example of how this might be done with two of Collins' question strategies, "overgeneralizing the misconception" and "testing the hypothesis," is shown in Figure 3.

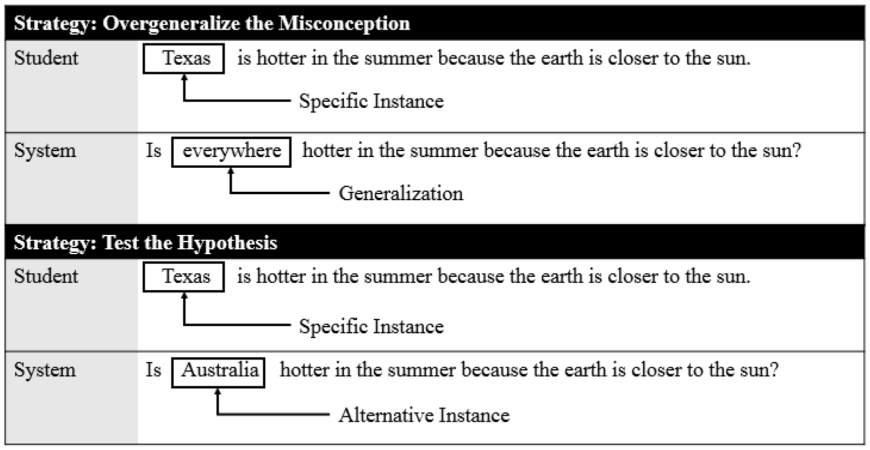

Strategy: Overgeneralize the Misconception
Student
Texas
is hotter in the summer because the earth is closer to the sun.
Specific Instance
System
Is
everywhere
hotter in the summer because the earth is closer to the sun?
Generalization
Strategy: Test the Hypothesis
Student
Texas
is hotter in the summer because the earth is closer to the sun.
Specific Instance
System
Is
Australia
hotter in the summer because the earth is closer to the sun?
Alternative Instance

Figure 3: Different types of question strategies can be used to guide students toward recognizing their misconceptions.

Another more complex strategy that can be used by teachable agents involves exploring the use of metaphor as a means of guiding students toward recognizing misconception. A visual representation of the matching process between expert and student-taught knowledge can be seen in Figure 4. In the case of this figure, the solid connections and concepts represent information taught by the student, and the dashed connection represents conflicting knowledge from the domain model. Thus, it is determined that the two concepts, "Separate Strands" and "Nucleotides," do exist in the domain model but that the student has reversed their relation. Once a misconception type has been identified, the system can search for analogous concepts and relationships to the set that has been misconceived, and these sets can be presented to the user in a peer-like, interactive manner, as described in the following paragraphs and figures. The hypothesis is that this will often cause the student to draw his or her own parallels between the analogous correct conception and the misconception, hopefully recognizing the contradiction between the two without the use of any didactic correction from the virtual tutor.

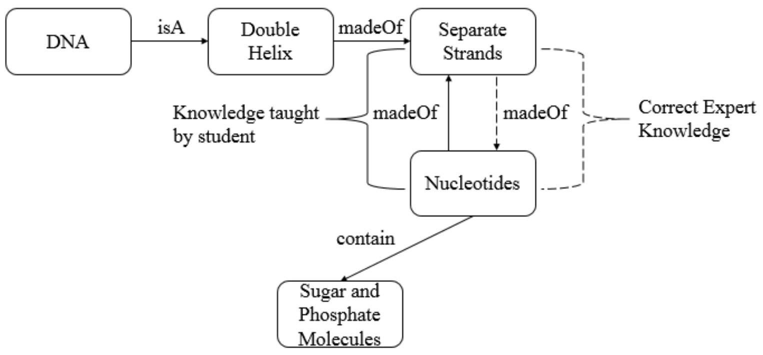
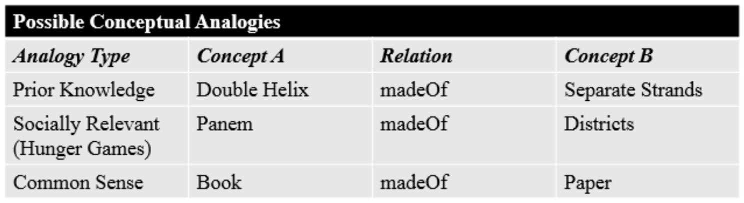


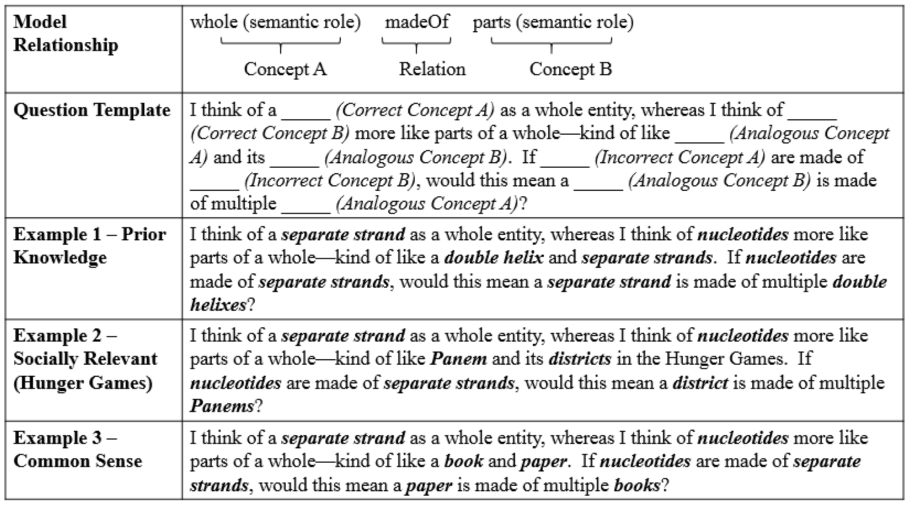
Figure 4: Teachable agent systems can detect misconception types by comparing user-taught knowledge to that of an expert concept map. One misconception type that could be detected, as seen in this example, is the reversal of the relationship between concepts.

Of course, a matter that remains to be determined is of what type of analogy will be most effective in promoting learning gains. Three possible conceptual analogies for the misconception in Figure 4 are presented in Figure 5. These are followed in Figure 6 with an example of how each would look when placed into the same question template. One focuses on prior correct knowledge that the student has already taught, one presents a socially relevant analogy that could increase student interest by making the material more relatable, and one features common sense knowledge likely to be readily understood by anyone.



Possible Conceptual Analogies			
Analogy Type	***Concept A***	***Relation***	***Concept B***
Prior Knowledge	Double Helix	madeOf	Separate Strands
Socially Relevant (Hunger Games)	Panem	madeOf	Districts
Common Sense	Book	madeOf	Paper

Figure 5: A number of different types of conceptual analogies may be presented to students to help them understand the correct relationship between misconceived concepts. Possible conceptual analogy domains include prior knowledge, socially relevant knowledge, and common sense knowledge.



Model Relationship	$\underbrace{\text{whole (semantic role)}}_{\text{Concept A}}$ $\underbrace{\text{madeOf}}_{\text{Relation}}$ $\underbrace{\text{parts (semantic role)}}_{\text{Concept B}}$
Question Template	I think of a _____ *(Correct Concept A)* as a whole entity, whereas I think of _____ *(Correct Concept B)* more like parts of a whole—kind of like _____ *(Analogous Concept A)* and its _____ *(Analogous Concept B)*. If _____ *(Incorrect Concept A)* are made of _____ *(Incorrect Concept B)*, would this mean a _____ *(Analogous Concept B)* is made of multiple _____ *(Analogous Concept A)*?
Example 1 – Prior Knowledge	I think of a ***separate strand*** as a whole entity, whereas I think of ***nucleotides*** more like parts of a whole—kind of like a ***double helix*** and ***separate strands***. If ***nucleotides*** are made of ***separate strands***, would this mean a ***separate strand*** is made of multiple ***double helixes***?
Example 2 – Socially Relevant (Hunger Games)	I think of a ***separate strand*** as a whole entity, whereas I think of ***nucleotides*** more like parts of a whole—kind of like ***Panem*** and its ***districts*** in the Hunger Games. If ***nucleotides*** are made of ***separate strands***, would this mean a ***district*** is made of multiple ***Panems***?
Example 3 – Common Sense	I think of a ***separate strand*** as a whole entity, whereas I think of ***nucleotides*** more like parts of a whole—kind of like a ***book*** and ***paper***. If ***nucleotides*** are made of ***separate strands***, would this mean a ***paper*** is made of multiple ***books***?

Figure 6: Analogous concepts and relations can be inserted into question templates created for specific conceptual relationship models. The resulting questions can then be presented to learners to prompt them to restructure their thinking.

There are advantages and disadvantages to each analogy type, and different types may be best suited for different learning contexts. An advantage of using pre-existing knowledge is that the system knows for sure that the student already understands the analogous conceptual relationship enough to teach it correctly. However, this can also pose risks, since bringing this otherwise resolved information into an analogy with a concept that the student has misconceived may cause the student to rethink the wrong concept, or at the very least introduce an unnecessary degree of confusion. Alternatively, an advantage of using socially relevant knowledge is that the correct relationship between the analogous concepts is likely to be immediately clear to the student, and moreover is not likely to be a relationship that could be second-guessed. The disadvantage to this strategy, of course, is that if the student is not familiar with the socially relevant analogy then he or she is unlikely to find much help from its use as a propositional metaphor. Care must be taken to ensure that if socially relevant analogies are used, they are known almost to the point of being common sense knowledge. To that end, an advantage of using common sense knowledge is that it is almost always immediately obvious to students. Its disadvantage, however, is that the analogy may not seem as relevant or exciting to students as the other analogy types. It also may be more difficult to introduce a common sense analogy as an authentic question, rather than as a means of pointing out an obvious misconception. As long as the strategy of using a propositional metaphor to effect conceptual change existed in the system, these analogy domains themselves could be easily interchanged according to the system author's preference.

Implications for GIFT

These recommendations lead to several main implications for GIFT as a whole. First, the proposed instructional strategies will allow GIFT to diversify its pedagogical agent roles by introducing the ability to directly promote more active learning characteristics, through acting as a naïve teachable agent rather than dispensing purely didactic dialogue. The only assumption associated with this approach, aside from what is already associated with any ITS or educational system, is that the student is able to teach. An architectural implication of the approach is that the domain module may need to be expanded to include more background knowledge, particularly if propositional metaphors from common sense or socially relevant domains are used to guide students toward diagnosing their misconceptions. Models required for implementation will include a learner model located in the user module, a learned teachable agent model located in the visible knowledge module, and a domain model containing a "gold standard" of the concepts and relations to be learned located in the domain module. The incorporation of multiple roles in this approach will allow students to experience a more realistic and thus effective learning environment, and this will broaden the educational scope of GIFT to include domains that may not have previously been feasible with the use of only a single pedagogical role. Of course, the structural implication to this is that the pedagogical module must be modified so that it can include the two related processes of (1) agent selection and (2) strategy selection.

With the addition of a teachable agent role to GIFT, lower-achieving learners in particular are likely to experience increased learning gains and motivation. Since teachable agents must simulate having a weaker knowledge base, an obvious interface implication has already been discussed—the teachable agent's visible knowledge base should not be the entire expert domain knowledge base, but rather the current knowledge base reflecting concepts and relationships already taught by the student. The exact approach to doing this should reflect the design impacts on the system as a whole, and careful considerations should be made to how they may affect overall flexibility and modularity.

Conclusions

In conclusion, GIFT can be adapted for use with teachable agents, as well as with a multi-agent system of any type, according to the recommendations within. In particular, doing so requires the addition of targeted, question-based strategies designed to promote deeper learning and resemble the queries of a peer learner rather than the didactic suggestions of an authoritative tutor. To accommodate multiple agents of any type, it requires a small set of structural modifications to the pedagogical module as well as additions to the domain module. Recommendations for these modifications include revising the pedagogical module so that it exhibits a structure capable of selecting both an interactive agent and a strategy, and adding an additional externally evident knowledge module to demonstrate the knowledge taught by the learner. The beneficial implications of these changes are manifold: through the ability to portray weaker agent roles as well as tutor roles, the system can be expected to see an increase in learner motivation and learning gains, and through the ability to portray multiple roles within the same system, learners will be presented with a more realistic learning environment that can be tailored to their individual pedagogical needs, likely multiplying the efficacy of the system even more.

Acknowledgements

This research was supported by the Institute of Education Sciences, U.S. Department of Education, Grant R305A120808 to the University of North Texas. The opinions expressed are those of the authors.

References

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

Azevedo, R., Witherspoon, A. M., Graesser, A. C., McNamara, D. S., Chauncey, A., Siler, E., ... & Lintean, M. C. (2009, July). MetaTutor: Analyzing Self-Regulated Learning in a Tutoring System for Biology. In *AIED* (pp. 635-637).

Baylor, A. & Ebbers, S. J. (2003). Evidence that multiple agents facilitate greater learning. *Artificial intelligence in education: Shaping the future of learning through intelligent technologies*, 377-379.

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.

Blair, K., Schwartz, D., Biswas, G. & Leelawong, K. (2007). Pedagogical agents for learning by teaching: Teachable agents. *Educational Technology*, 47(1), 56.

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.

Chou, C. Y., Chan, T. W. & Lin, C. J. (2003). Redefining the learning companion: the past, present, and future of educational agents. *Computers & Education*, 40(3), 255-269.

Collins, A. (1985). Teaching reasoning skills. In S. Chipman, J. Segal & R. Glaser (Eds.), *Thinking and learning skills: Current research and open questions* (Vol. 2). Hillsdale, NJ: Erlbaum, 579-586.

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

Goodman, B., Soller, A., Linton, F. & Gaimari, R. (1997). Encouraging student reflection and articulation using a learning companion. In *Proceedings of the AI-ED 97 World Conference on Artificial Intelligence in Education* (pp. 151-158).

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.

Hietala, P. & Niemirepo, T. (1998). The competence of learning companion agents. *International Journal of Artificial Intelligence in Education (IJAIED)*, 9, 178-192.

Kim, Y. (2007). Desirable characteristics of learning companions. *International Journal of Artificial Intelligence in Education*, 17(4), 371-388.

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.

Leelawong, K. & Biswas, G. (2008). Designing learning by teaching agents: The Betty's Brain system. *International Journal of Artificial Intelligence in Education*, 18(3), 181-208.

Lester, J. C., Converse, S. A., Kahler, S. E., Barlow, S. T., Stone, B. A. & Bhogal, R. S. (1997, March). The persona effect: affective impact of animated pedagogical agents. In *Proceedings of the ACM SIGCHI Conference on Human factors in computing systems* (pp. 359-366). ACM.

Matsuda, N., Yarzebinski, E., Keiser, V., Raizada, R., Stylianides, G. J., Cohen, W. W. & Koedinger, K. R. (2011, January). Learning by teaching SimStudent–An initial classroom baseline study comparing with Cognitive Tutor. In *Artificial Intelligence in Education* (pp. 213-221). Springer Berlin Heidelberg.

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. *Reading comprehension strategies: Theories, interventions, and technologies*, 397-421.

Millis, K., Forsyth, C., Butler, H., Wallace, P., Graesser, A. & Halpern, D. (2011). Operation ARIES!: A serious game for teaching scientific inquiry. In *Serious games and edutainment applications* (pp. 169-195). Springer London.

Moreno, R., Mayer, R. E., Spires, H. A. & Lester, J. C. (2001). The case for social agency in computer-based teaching: Do students learn more deeply when they interact with animated pedagogical agents?. *Cognition and Instruction*, 19(2), 177-213.

Nichols, D. (1994). Issues in designing learning by teaching systems.

Nielsen, R. D., Buckingham, J., Knoll, G., Marsh, B. & Palen, L. (2008). A taxonomy of questions for question generation. In *Proceedings of the Workshop on the Question Generation Shared Task and Evaluation Challenge.*

Otero, J. & Graesser, A. C. (2001). PREG: Elements of a model of question asking. *Cognition and instruction*, 19(2), 143-175.

Prensky, M. (2005). Computer games and learning: Digital game-based learning. *Handbook of computer game studies*, 18, 97-122.

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.

Roscoe, R. D., Segedy, J. R., Sulcer, B., Jeong, H. & Biswas, G. (2013). Shallow strategy development in a teachable agent environment designed to support self-regulated learning. *Computers & Education*, 62, 286-297.

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.A. & Holden, H.K. (2013). Motivations for a generalized intelligent framework for tutoring (gift) for authoring, instruction and analysis. In: *AIED 2013 Workshops Proceedings* Volume 7. (pp. 1-9).

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

Wißner, M., Beek, W., Lozano, E., Mehlmann, G., Linnebank, F., Liem, J., ... & André, E. (2012). Increasing Learners' Motivation through Pedagogical Agents: The Cast of Virtual Characters in the DynaLearn ILE. In *Agents for Educational Games and Simulations* (pp. 151-165). Springer Berlin Heidelberg.

THEME VI: DIALOGUE-BASED TUTORING

Using GIFT as an Adaptation Engine for a Dialogue-Based Tutor

Debbie Brown[1], Eric Martin[2], Fritz Ray[1], Robby Robson[1]
[1]Eduworks Corporation
[2]Engineering and Computer Simulations, Inc.

Introduction

The Generalized Intelligent Framework for Tutoring (GIFT) (Sottilare, Brawner, Goldberg et al., 2012; Sottilare & Holden, 2013b) is a flexible, open-source platform that can be used to implement a variety of Intelligent Tutoring Systems (ITSs). Developed by the U.S. Army Research Laboratory (ARL), GIFT uses a modular architecture in which various components communicate via a messaging bus. Major components include a domain module that defines domain knowledge, a pedagogical module that determines which instructional strategy to apply, a user or learner module that maintains a learner model, and a sensor module that GIFT uses to collect and process data from biometric and other sensors. As reported in the inaugural GIFT symposium (Sottilare & Holden, 2013a), GIFT has been integrated with game engines, used to create cognitive tutors, used to combine multiple forms of learning experiences, and even used to conduct surveys. In this paper we describe an application of GIFT arising from a Small Business Innovation Research (SBIR) project called Tools for the Rapid Development of Expert Models (TRADEM) in which GIFT is used as the adaptive engine for tutors that result from a semi-automated authoring process (or more accurately, transformation process) that is applicable to any domain of study.

The TRADEM Process

The goal of TRADEM is to reduce the time and expense required to produce ITS in arbitrary domains and to thereby create a means to develop training that is more effective than standard eLearning or Computer Based Training (Dodds & Fletcher, 2004; Fletcher, 2011; Graesser, Conley & Olney, 2012; K. VanLehn, 2011). To accomplish this, TRADEM starts with a corpus of traditional content representing instruction and knowledge in a domain. TRADEM then uses natural language processing (NLP), machine learning (ML), and statistical techniques to analyze the corpus and automatically generate topics and a prerequisite structure. TRADEM also associates a variety of content with each topic, including instructional material, media references, assessment questions, and prompts derived from the source corpus. Together, the data generated by TRADEM are an *expert model* (Murray, 2003; Woolf, 2009) that can be used to present and sequence learning, ask learners questions, and evaluate learner responses. The tools developed for TRADEM allow the topics, prerequisites, paragraph content and ordering, media references, instructional strategy preferences, and questions in the expert model to be edited and exported in a JSON file (referred to hereafter as the Export). The JSON representation of the expert model is further transformed by TRADEM to an intelligent tutor. Figure 1 shows the TRADEM process; further details may be found in (Ray, Brawner & Robson, 2014; Robson, Ray & Cai, 2013).

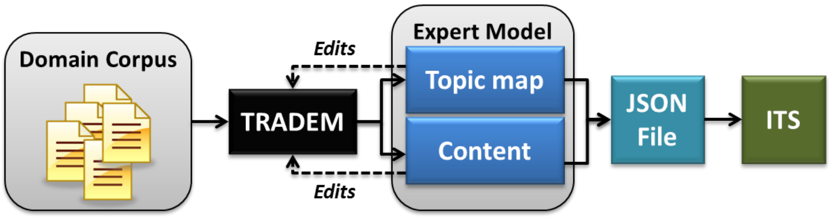
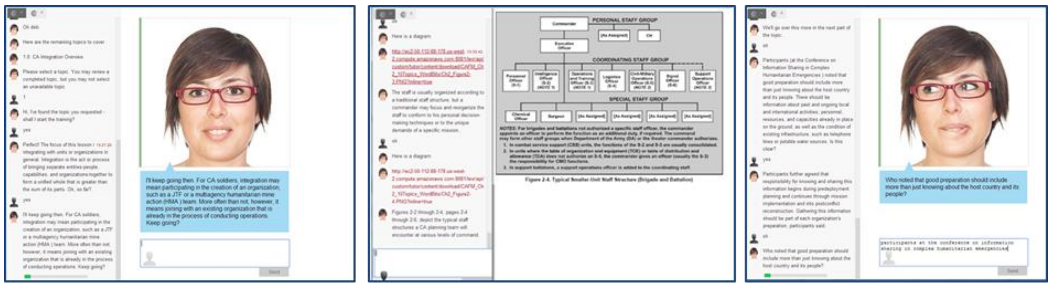


Figure 24: TRADEM Process

T-Tutors

The TRADEM process is designed to generate tutors that operate and interact with learners in multiple ways. To date it has been used to generate a family of *conversational tutors* (called *T-Tutors* for "TRADEM-Tutors") that interact with learners using a simple dialogue interface. T-Tutors combine features of the AutoTutor family of ITS (Graesser, Person, Lu et al.; 2005; Graesser, Wiemer-Hastings, Wiemer-Hastings et al., 1999) with mastery-based sequencing (Bloom, 1968; Stern & Woolf, 1998) and instructional strategies based on work of Gagne, Merrill and others (Gagne, 1985; Merrill, 1994, 2002). The following features of T-Tutor are planned, with the majority implemented at the time of this writing:

Conversational interface with 2D Avatar: Learners converse with T-Tutors using verbal input with written input as an alternative. Verbal exchange is supported by text-to-speech and speech-to-text technology. As shown in Figure 2, T-Tutor displays a) a 2D image of a face selected from a small set of expressions and b) contextually relevant learning resources during the dialogue. Resources can be anything from links or documents to videos and simulations. While research has demonstrated that nonverbal communication of pedagogical agents contributes to improved learner attention, engagement, social connectedness, and understanding (Doering, Veletsianos & Yerasimou, 2008; Veletsianos, 2009), some studies warn designers of the strongly negative "uncanny valley phenomenon" resulting from a learner's impression of not-quite-human likeness (Tinwell, Grimshaw, Williams et al., 2011; Yamada, Kawabe & Ihaya, 2013). Veletsianos (2009) demonstrated that animated pedagogical agents were often distracting to the learner, and (Baylor & Kim, 2008, 2009) found that simplified nonverbal communications such as facial expression were most effective for improving learning outcomes. The approach of using 2D avatars representing simple emotions was chosen to provide a pedagogically effective level of nonverbal expression while reducing overhead and bandwidth for smartphone delivery.


I'll keep going then. For CA soldiers, integration may mean participating in the creation of an organization, such as a JTF or a multiagency humanitarian mine action (HMA) team. More often than not, however, it means joining with an existing organization that is already in the process of conducting operations. Keep going?

Who noted that good preparation should include more than just knowing about the host country and its people?

Figure 25: T-Tutor Interface

Natural Dialogue: T-Tutor engages the learner in natural dialogue using the capabilities of ChatScript (Wilcox, 2013; Wilcox & Wilcox, 2013). ChatScript is a chatbot engine and scripting language that provides a rich vocabulary of rules and pattern matching and interprets semantics based on the WordNet ontology (Wordnet, 2012). It implements conversational strategies such as gambits and responders, and it has been used to construct chatbots that have won the Loebner Prize in artificial intelligence (Bradeško & Mladenić, 2013).

Instructional Phases: T-Tutor implements four basic instructional phases: Contextualization, Learning, Practice, and Assessment. T-Tutor uses different strategies within each phase, which are aligned with Gagne's Nine Events of Instruction (Doube, 1998; Gagne, 1985). During contextualization, T-Tutor introduces a topic and explores readiness, motivation and interest. During learning, T-Tutor conversation-ally presents information, displays resources, and provides prompts to retain interest and maintain interactivity. During practice T-Tutor asks questions about the information and provides contextual guidance. During assessment ,T-Tutor presents a series of questions with no feedback until the end. T-Tutor continuously tracks the learner state during each of the instructional phases.

Adaptations: T-Tutor adapts to the learner in multiple ways. The tutor follows a sequence of topics ordered by the prerequisite structure in the TRADEM expert model. A learner can only enter a topic if all of its prerequisites have been met. Similarly, T-Tutor skips topics that a learner has mastered and can return to pre-requisite topics if a learner does poorly. During the presentation of a topic, T-Tutor dynami-cally evaluates learner responses and adjusts the level of material along multiple dimensions, including speed, intrinsic difficulty (e.g., reading level), interactivity, and levels of knowledge in Bloom's taxono-my (Forehand, 2010; Milman, 2009; Ruskov, Ekblom & Sasse, 2013). During the learning and practice instructional phases, T-Tutor implements conversational moves that use pumps and hints when learner responses do not fully address a question (Wolfe, Widmer, Reyna et al., 2013; Zhou, Freedman, Glass et al., 1999) and adapts to off-topic responses by using conversational gambits to bring the learner back on task (Wilcox & Wilcox, 2013).

Granules

The data used to populate and instantiate a T-Tutor come from the TRADEM export and undergo a series of transformations, most of which are automated.

A single T-Tutor may cover one or more topics in the TRADEM expert model, and within each topic, TRADEM produces a set of *granules*. A granule is the smallest block of knowledge around which a conversation can be held. TRADEM uses NLP and machine learning to generate attributes, which, in turn, are used to map granules to instructional strategies. These attributes can be edited by the user. Although granules are generally equivalent to paragraphs, a single paragraph of text may be converted into multiple granules tagged with different instructional strategies:

- A learning granule covering the facts or concepts expressed in the paragraph.
- A practice granule that includes open-ended questions about the facts or concepts and guidance that is given to the learner if an answer appears to be incomplete, irrelevant or incorrect.

- An assessment granule that quizzes the user on the content of the paragraph. The assessment granule may include explicit answers used to score responses or text that is used to evaluate responses via semantic analysis as is done by AutoTutor (Graesser, Chipman, Haynes et al., 2005; Hu, Cai, Han et al., 2009; Wolfe, Widmer, Reyna et al., 2013).

Each granule goes through two more processes: dialogue transformation and annotation. The purpose of *dialogue transformation* is to convert the granule into content appropriate for use in ChatScript. This transformation uses templates, NLP, ontologies, dialogue moves, and other techniques. The purpose of *annotation* is to edit or add metadata that enables T-Tutor to operate on the granules as instructional elements and select responses based on changes in learner state. Each granule is annotated with a title; a level in Bloom's taxonomy; a difficulty level; an interactivity level; and its stages in three instructional strategies: the GIFT Engine for Macro/Micro-Adaptive Pedagogy (eM^{2}AP) (Goldberg, 2013), T-Tutor's four phases, and Gagne's nine events. Topics that were exported from the TRADEM expert model are also annotated with titles, prerequisites, and associated granules.

T-TUTOR User Interface (UI)

The T-Tutor Agent delivers instruction using the Extensible Messaging and Presence Protocol (XMPP) for client-server chat. Originally derived from Jabber™ (Ozturk, 2010), XMPP is used by many established networks such as Google™, Facebook™, and WebEx™ (Burt, 2011; Facebook, 2014; Warne, 2006). XMPP servers are often deployed in large enterprises to facilitate private conversation among employees in the office, in the field, and abroad.

The T-Tutor client interface is built using responsive design (Frain, 2012; Gardner, 2011; Marcotte, 2010). The version previously shown in Figure 25 is the full-screen browser version. Information is delivered to the UI as web content. This obviates the need for a learner to install additional software or viewers in most cases. At the same time, T-Tutor can gracefully degrade for Mobile devices and, in fact, can be delivered through other XMPP client applications such as Google Hangouts™ and may be easily integrated into simulations and other e-learning applications.

The T-Tutor client connects to the T-Tutor Agent web service and waits for conversations to start. The web service can handle any number of simultaneous conversations. During a conversation, the T-Tutor UI delays dialog by an appropriate amount of time to allow users to read or listen to what has been said.

T-Tutor Sessions

Within a tutoring session, adaptation and sequencing takes place at three levels:

- At the *macro* level, the learner is moved from topic to topic (macro-sequencing). Sequencing based on recorded completion and mastery of topics is considered to be macro-adaptation.

- At the *mezzo* level, different instructional strategies are selected within a topic. These are not true macro-adaptations because they are selected based on active learner state, nor are they micro-adaptations because they are above the level of conversational interactions and are not part of the inner loop of the tutor as described by (Aleven, Mclaren, Sewall et al., 2009; Durlach & Ray, 2011; Kurt Vanlehn, 2006). We call these mezzo-adaptations.

- At the *micro* level, the tutor interacts directly with the learner to present a particular instructional strategy and alters its behavior based on learner responses.

When a tutoring session is initiated, the T-Tutor Agent loads the files produced from the TRADEM export by the transformations discussed above. The learner interacts with and is guided through the session by the T-Tutor UI. At the macro level, learners may initiate a session and select topics from a menu that offers only those topics for which prerequisites have been met.

Once a topic is selected by the learner, mezzo-adaptation is guided by the GIFT framework using a common pattern sequence governed by learner states and rules. As is illustrated in Figure 3, for each topic selected by the learner, the T-Tutor Agent sequences through instructional strategy stages, displays the content matching each stage to the user one granule at a time, collects learner state information, and passes state information to GIFT. After each instructional strategy stage, GIFT examines the learner state, uses rules to detect state transitions, and tells the T-Tutor Agent which instructional strategy stage (or tactic) should be presented next. T-Tutor Agent then retrieves the granules that match the desired strategy stage and proceeds to deliver the granules to the T-Tutor UI. This mezzo-sequencing loop is repeated until the learner exits or there are no more instructional strategy stages for the topic. At this point, the T-Tutor resumes macro-level sequencing and allows the learner to select the next topic or tutor.

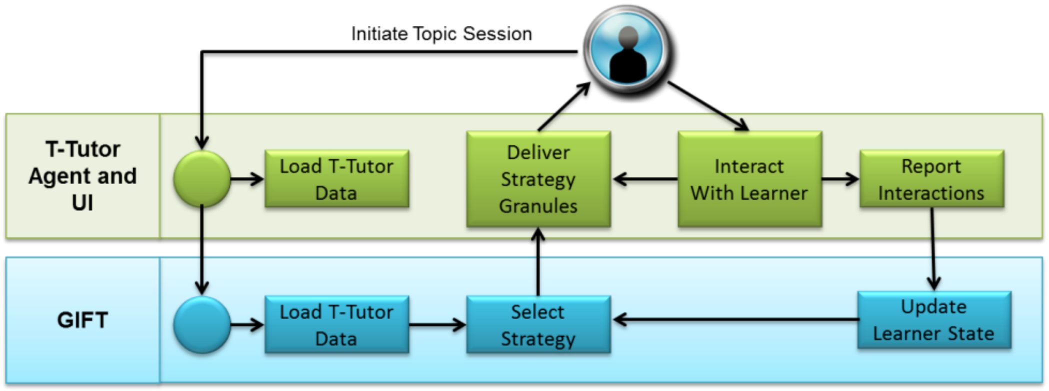


Figure 26: T-Tutor Session Mezzo Sequencing Loop

Micro-adaptation takes place within a granule, where a learner may interact in multiple ways with T-Tutor. For example, a practice granule might ask a question, evaluate a response, offer a hint, evaluate a second response, and offer remediation. If a learner wanders off topic, T-Tutor will respond with a conversational gambit that attempts to re-focus the conversation on the learning material. T-Tutor can also respond to queries about the meaning of terms and provide help when asked.

The next section provides detail about using the components of GIFT to implement T-Tutor sessions.

GIFT Integration

GIFT's architecture is described in detail in these references (Goldberg & Cannon-Bowers, 2013; Nye, 2013; Ragusa, Hoffman & Leonard, 2013), in other articles in the referenced workshop volume from the 2013 AI in Education (AIED) conference, and in documentation from (GIFT Project, 2014). In this section, we list the components of GIFT used by T-Tutor, explain how data for T-Tutor are consumed by these components, and describe how the components of GIFT and the T-Tutor UI and Web Service interact with each other.

As depicted in Figure 27, GIFT support files are generated from the export through the use of the Student Information Models for Integrated Learning Environment (SIMILE) Workbench during an authoring phase. The files required to inform adaptation within the GIFT portion of T-Tutor include the domain knowledge file (DKF), course file, and SIMILE IXS file. The TRADEM export is used to generate a course file (representing the topic structure in the expert model). The course file may be further refined using the GIFT Course Authoring Tool (CAT). The DKF and IXS files define rules, transitions, and tactics for T-Tutor mezzo-sequencing behavior, and this information may be edited and exported from SIMILE Workbench. After the automated T-Tutor data transformations are complete, the T-Tutor import contains the topic outline, prerequisites, and all of the granule content with mezzo- and micro-sequencing strategy and level-adaptation metadata.

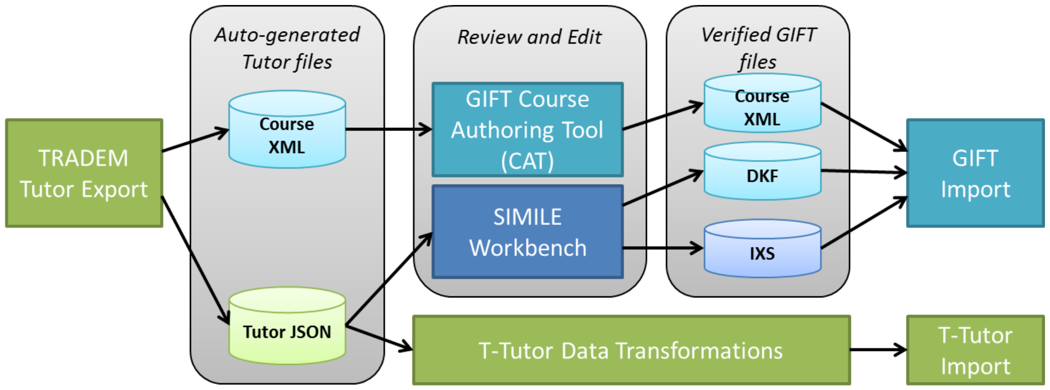


Figure 27: GIFT Authoring (as used by TRADEM)

SIMILE Workbench: SIMILE Workbench is a client application that enables a user to define tutor states and reasoning logic and export these in a format understood by GIFT. SIMILE Workbench is used to generate the DKF and IXS file from the TRADEM export, i.e., to effect the transformation from the TRADEM export to the data required by T-Tutor. Generally, the same set of DKF and IXS rules can be used across multiple tutors since they serve to define mezzo-state transitions for instructional strategies, which we assume to be independent of the topic and domain. However, SIMILE Workbench can be used to create different rules for different topics or domains if needed.

Domain Knowledge File: The DKF is an XML file that includes data used by the Domain and Pedagogical modules in GIFT. The data in the DKF include all of the information needed to execute a single tutor, including possible states, rules, and tactics that determine which pedagogical actions are triggered by changes in learner state. This is related to pedagogical metadata encoded in the export produced by the TRADEM process and defines the mezzo-sequencing rules in a topic-independent manner.

IXS File: The IXS (also known as the SIMILE knowledge base file) is the SIMILE runtime logic file, which also serves as the SIMILE Workbench project file. It contains objects, states, and reasoning logic (or rules) needed to detect state transitions and trigger the next action (or tactic) within GIFT. This file is generated or updated when a user saves a SIMILE Workbench project.

GIFT CAT: The CAT is included in the GIFT suite of XML authoring tools and is used to validate and edit a domain structure in the course file. The tool is not modified for use by TRADEM.

Course File: The GIFT course file represents the overarching sequence of topics within a tutor with integrated guidance to be presented to the learner. The TRADEM export process auto-generates the course file from its expert model, and the XML can be further edited using the CAT. The course file is used for macro-sequencing, but another architecture is possible in which the course file is not used and a mechanism such as SCORM is used for sequencing instead (Robson, Ray & Cai, 2013).

Figure 28 shows how T-Tutor uses GIFT services during the execution of a tutor to direct mezzo-sequencing. The following provides details for how TRADEM integrates with each component.

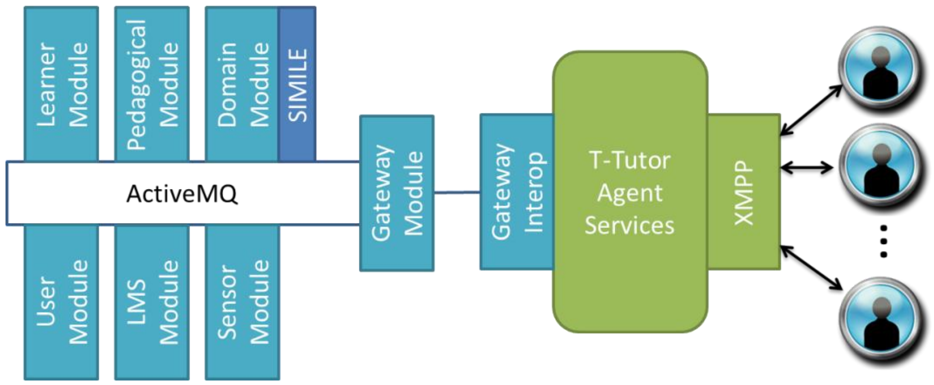


Figure 28: GIFT runtime architecture (as used by TRADEM)

Gateway Module: The Gateway module is the primary interface between GIFT and training applications, in this case, between GIFT and the T-Tutor Agent services. T-Tutor uses a Gateway Interop plugin to communicate with the GIFT Gateway module. The Gateway transports GIFT messages through an ActiveMQ™ message bus. This message bus is built into GIFT and distributes each message to the appropriate module. For TRADEM we developed a generic JSON message for the Gateway module that wraps JSON messages. This new message supports a data-driven approach to messaging that does not have to be re-programmed for each instance of a training application message – developers instead must define a JSON structure for each message payload, but that is the extent of the work that needs to be done.

Domain Module: The Domain module selects concepts from the DKF based on the current learner state and an instructional strategy recommended by the Pedagogical module. In T-Tutor, the learner state consists of topic, objective and granule completion and mastery data as well as the current stage in an instructional strategy. The Domain module adaptively selects instructional strategy stages while a learner sequences through a tutor and its topics. The instructional strategy stages are passed through the Gateway module to the T-Tutor Agent, which, in turn, launches the content into the learner's T-Tutor UI. We note that in developing T-Tutor we were somewhat limited by the types of messages that GIFT could process. Originally, we had hoped to pass rich recommendation data that included multiple characteristics of granules. In practice, we had to use workarounds to encode this data in "tactics" recommended from the DKF.

SIMILE: The SIMILE runtime engine is tightly integrated with the Domain module and assists with the tasks of interpreting the T-Tutor JSON messages and determining transitions and tactics for the learner's next phase of instruction. SIMILE uses the IXS file to reason on state transitions.

Pedagogical Module: The Pedagogical module recommends the next instructional strategy using the state transition rules represented in the DKF applied to the learner's current state data. In the TRADEM-GIFT integration the "instructional strategy" is a stage within a strategy, e.g., the "gain attention" stage of Gagne's nine events (Gagne, 1985) or the "recall" stage within the GIFT eM^{2}AP strategy (Goldberg, 2013).

Learner Module: The Learner module within GIFT maintains learner state data. As indicated above, this is currently mastery and completion data, but we plan to include other types of interaction data gathered from the T-Tutor UI, and GIFT has the ability to add affect data gathered through sensors.

LMS Module: The LMS module provides learners access to GIFT courses and works closely with the User module and Survey module to validate and collect learner information. After a tutor is imported into GIFT, it appears in the LMS course list. Since it is a requirement of the TRADEM project to integrate with a customer's existing infrastructure, TRADEM does not solely rely on the GIFT LMS to launch a specific T-Tutor.

Discussion

We encountered several challenges in designing and developing the GIFT integration approach described in this paper, but the chief issue was how to divvy up responsibilities for sequencing and adaptation and, correspondingly, whether to architect T-Tutor as a module contained within GIFT or implement GIFT as a service contained within T-Tutor. The former would have been possible. We could have treated each topic and granule as a separate executable to be referenced in the GIFT Course File and DKF and launched by GIFT. Instead, we integrated GIFT into the web services framework used to run T-Tutor and used GIFT as a mezzo-adaptation engine. We believe this is the more flexible approach. As implemented, the T-Tutor Agent can sequence topics and granules based on grounded instructional strategies without using GIFT but can also can leverage GIFT's features if GIFT is present. In the future, we expect that other GIFT tutors will use the T-Tutor Agent, whether or not it is used with a TRADEM-generated tutor.

Division of Adaptations

In addition to being technically more flexible, the approach we chose embodies a basic design principle:

- In using GIFT as an adaptation engine for a tutor, GIFT should be used for macro- and mezzo-adaptations that apply to any form of tutor. The portion of the tutor that interacts directly with the learner should be responsible for retrieving content based on GIFT recommendations and for any micro-adaptations based on learner interactions.

This design principle is reflected in T-Tutor and is hypothesized to be applicable in other contexts. It is significant because it separates the role of determining strategy (GIFT's job) from providing an engaging, unique and effective tutoring experience (the tutor's job). As pointed out in (Robson & Barr, 2013), the latter is an area where competition may lead to better tutoring systems, while the former is an area that can be standardized to lower the costs of production and barriers to adoption.

Anticipated Improvements

Both TRADEM and GIFT are ongoing projects, but the basic frameworks for each are established. Integration will continue as improvements are released. Currently, TRADEM effectively uses automation to identify topics and topic names, align content, and generate simple questions, and the edit tools allow a designer or SME to further modify and tag attributes as needed. The focus of ongoing TRADEM research is on improving the auto-generated expert model, improving the algorithms used to transform the TRADEM export to a T-Tutor, and on improved conversational tutor behavior.

Design-based Research

Consistent with design-based research (Amiel & Reeves, 2008; Barab & Squire, 2004; Hoadley, 2004), the key question raised by this paper is whether the design principle above is broadly applicable and how it should be modified. One way to explore this question is to apply the same design principle to multiple types of tutors. We have not used TRADEM to produce example-tracing (Aleven, Mclaren, Sewall et al., 2009) or constraint-based (Mitrovic, Martin, Suraweera et al., 2009; Ohlsson & Mitrović, 2006) tutors, but the data required to construct the mastery-based outer loop (Vanlehn, 2006) of such tutors is present in TRADEM expert models, and we believe that the data required for the inner loops of such tutors can be similarly derived by using similar transformation techniques. Developing an example-tracing or constraint-based export would provide an opportunity to test and refine both the technical approach and design principle.

Conclusion

In this paper, we have described a process (the TRADEM process) that applies NLP and text mining to analyze didactic content and produce data for developing and populating dialogue-based intelligent tutoring systems called T-Tutors. Each T-Tutor has an adaptive front end that interacts with the learner through natural language and uses GIFT as an adaptation engine. GIFT recommends instructional strategies and T-Tutor translates these recommendations into action. We have described technical aspects of this integration and derived a general principle for how such systems should be built.

Our work has demonstrated that the TRADEM process and the proposed approach to developing ITS are feasible. Our work also suggests further investigating the use of GIFT as an adaptation engine for different types of tutors and highlights the importance of GIFT's capabilities to pass messages to externally running systems and agents. We recommend that these capabilities be expanded. Currently, GIFT selects strategies using fixed rules applied to changes in a pre-defined set of learner states. By passing learner model parameter values between GIFT and an external system, GIFT could be modified to recommend both strategies *and interventions* based on combinations of states *and learner attributes*.

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 Intelligence in Education, 19(2), 105-154.

Amiel, T. & Reeves, T. C. (2008). Design-Based Research and Educational Technology: Rethinking Technology and the Research Agenda. *Journal of Educational Technology & Society, 11*(4).

Barab, S. & Squire, K. (2004). Design-based research: Putting a stake in the ground. *The journal of the learning sciences, 13*(1), 1-14.

Baylor, A. & Kim, S. (2008). The Effects of Agent Nonverbal Communication on Procedural and Attitudinal Learning Outcomes. In H. Prendinger, J. Lester & M. Ishizuka (Eds.), *Intelligent Virtual Agents* (Vol. 5208, pp. 208-214): Springer Berlin Heidelberg.

Baylor, A. & Kim, S. (2009). Designing nonverbal communication for pedagogical agents: When less is more. *Computers in Human Behavior, 25*(2), 450-457. doi: 10.1016/j.chb.2008.10.008

Bloom, B. S. (1968). Learning for mastery: Regional Education Laboratory for the Carolinas and Virginia Durham, NC.

Bradeško, L. & Mladenić, D. (2013). *A Survey of Chatbot Systems through a Loebner Prize Competition*. Ljubljana Slovenia: Artificial Intelligence laboratory, Jozef Stefan Institute.

Burt, J. (2011). Cisco Readies WebEx, Jabber for "Post-PC" Era. *eWeek*. http://www.eweek.com/c/a/Cloud-Computing/Cisco-Readies-WebEx-Jabber-for-PostPC-Era-689806/

Dodds, P. & Fletcher, J. D. (2004). Opportunities for New "Smart" Learning Environments Enabled by Next-Generation Web Capabilities. *Journal of Educational Multimedia and Hypermedia, 13*(4), 391-404.

Doering, A., Veletsianos, G. & Yerasimou, T. (2008). Conversational agents and their longitudinal affordances on communication and interaction. *Journal of Interactive Learning Research, 19*(2), 251-270.

Doube, W. (1998). Multimedia delivery of computer programming subjects: basing structure on instructional design (pp. 85): ACM - Association for Computing Machinery.

Durlach, P. J. & Ray, J. M. (2011). Designing adaptive instructional environments: Insights from empirical evidence *Army Research Institute Report*. Arlington, VA.

Facebook. (2014). Chat API. Retrieved April 2, 2014, from https://developers.facebook.com/docs/chat/

Fletcher, J. D. (2011). DARPA Education Dominance Program: April 2010 and November 2010 Digital Tutor Assessments. Alexandria, VA.

Forehand, M. (2010). Bloom's taxonomy. *Emerging perspectives on learning, teaching, and technology*, 41-47.

Frain, B. (2012). *Responsive web design with HTML5 and CSS3*: Packt Publishing.

Gagne, R. M. (1985). The conditions of learning and theory of instruction . New York: Holt, Rinehart and Winston (4th ed.). New York: Holt, Rinehart and Winston.

Gardner, B. S. (2011). Responsive Web Design: Enriching the User Experience. *Connectivity and the User Experience*, 13.

GIFT Project. (2014). GIFT Tutoring Documents. Retrieved March 31, 2014, from https://www.gifttutoring.org/projects/gift/documents

Goldberg, B. (2013). GIFT's Engine for Macro/Micro-Adaptive Pedagogy (eM2AP) and Micro-Adaptation Considerations Across a Domain-Independent Architecture: LITE Lab, Army Research Laboratory - Human Research & Engineering Directorate (HRED).

Goldberg, B. & Cannon-Bowers, J. (2013). *Experimentation with the Generalized Intelligent Framework for Tutoring (GIFT): A Testbed Use Case.* Paper presented at the AIED 2013 Workshops Proceedings Volume 7, Memphis, TN.

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

Graesser, A. C., Conley, M. W. & Olney, A. (2012). Intelligent tutoring systems. *APA handbook of educational psychology. Washington, DC: American Psychological Association*.

Graesser, A. C., Person, N., Lu, Z., Jeon, M. G. & McDaniel, B. (2005). Learning while holding a conversation with a computer. In L. PytlikZillig, M. Bodvarsson & R. Bruning (Eds.), *Technology-based education: Bringing researchers and practitioners together* (pp. 143 - 167): Information Age Publishing.

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

Hoadley, C. M. (2004). Methodological alignment in design-based research. *Educational Psychologist, 39*(4), 203-212.

Hu, X., Cai, Z., Han, L., Craig, S. D., Wang, T. & Graesser, A. C. (2009). *AutoTutor Lite*.

Marcotte, E. (2010). Responsive web design. *A list apart, 306*.

Merrill, M. D. (1994). *Instructional design theory*: Educational Technology.

Merrill, M. D. (2002). First principles of instruction. *Educational technology research and development, 50*(3), 43-59.

Milman, N. B. (2009). Crafting the "Right" Online Discussion Questions Using the Revised Bloom's Taxonomy as a Framework. *Distance Learning, 6*(4), 61.

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. (2003). An Overview of Intelligent Tutoring System Authoring Tools: Updated analysis of the state of the art. Chapter 17 in Murray, T., Blessing, S. & Ainsworth, S. *Artificial Intelligence*.

Nye, B. D. (2013). *Integrating GIFT and AutoTutor with Sharable Knowledge Objects (SKO).* Paper presented at the AIED 2013 Workshops Proceedings Volume 7, Memphis, TN.

Ohlsson, S. & Mitrović, A. (2006). Constraint-based knowledge representation for individualized instruction *Computer Science and Information Systems, 3*(1), 1-22. doi: 10.2298/CSIS0601001S

Ozturk, O. (2010, 2010 / 01 / 01 /). *Introduction to XMPP protocol and developing online collaboration applications using open source software and libraries.* Paper presented at the 2010 International Symposium on Collaborative Technologies and Systems (CTS), Chicago, IL.

Ragusa, C., Hoffman, M. & Leonard, J. (2013). *Unwrapping GIFT.* Paper presented at the AIED 2013 Workshops Proceedings Volume 7, Memphis, TN.

Ray, F., Brawner, K. & Robson, R. (2014). *Using Data Mining to Automate ADDIE*. Paper presented at the Submitted to Educational Data Mining 2014, London, UK.

Robson, R. & Barr, A. (2013). Lowering the Barrier to Adoption of Intelligent Tutoring Systems through Standardization. In R. A. Sottilare & H. K. Holden (Eds.), *Design Recommendations for Intelligent Tutoring Systems* (pp. 7). Orlando, FL: Army Research Lab.

Robson, R., Ray, F. & Cai, Z. (2013). *Transforming Content into Dialogue-based Intelligent Tutors.* Paper presented at The Interservice/Industry Training, Simulation & Education Conference Orlando, FL.

Ruskov, M., Ekblom, P. & Sasse, A. (2013). In Search for the Right Measure: Assessing Types of Developed Knowledge While Using a Gamified Web Toolkit. *Proceedings of the European Conference on Games Based Learning*, 722.

Sottilare, R. A., Brawner, K. W., Goldberg, B. S. & Holden, H. K. (2012). The Generalized Intelligent Framework for Tutoring (GIFT). Retrieved November 25, 2013, from https://www.gifttutoring.org/projects/gift/wiki/Overview

Sottilare, R. A. & Holden, H. K. (2013a). *Recommendations for Authoring, Instructional Strategies and Analysis for Intelligent Tutoring Systems (ITS): Towards the Development of a Generalized Intelligent Framework for Tutoring (GIFT)* Paper presented at the AI in Education 2013, Memphis, TN.

Sottilare, R. A. & Holden, H. K. (2013b). *Towards the Development of a Generalized Intelligent Framework for Tutoring (GIFT).* Paper presented at the Artificial Intelligence in Education.

Stern, M. K. & Woolf, B. P. (1998). *Curriculum sequencing in a web-based tutor.* Paper presented at the Intelligent Tutoring Systems.

Tinwell, A., Grimshaw, M., Williams, A. & Nabi, D. A. (2011). Facial expression of emotion and perception of the Uncanny Valley in virtual characters. *Computers in Human Behavior, 27*(2), 741-749. doi: 10.1016/j.chb.2010.10.018

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.

Veletsianos, G. (2009). The impact and implications of virtual character expressiveness on learning and agent–learner interactions. *Journal of Computer Assisted Learning, 25*(4), 345-357. doi: 10.1111/j.1365-2729.2009.00317.x

Warne, D. (2006). Google unshackles Google Talk. *APC, 26*(4), 14-14.

Wilcox, B. (2013). ChatScript (Version 3.81): SourceForge. Retrieved from http://chatscript.sourceforge.net/

Wilcox, B. & Wilcox, S. (2013). Making it Real: Loebner-winning Chatbot Design. *ARBOR Ciencia, Pensamiento y Cultura, 189*(764).

Wolfe, C., Widmer, C., Reyna, V., Hu, X., Cedillos, E., Fisher, C., . . . Weil, A. (2013). The development and analysis of tutorial dialogues in AutoTutor Lite. *Behavior Research Methods, 45*(3), 623-636. doi: 10.3758/s13428-013-0352-z

Woolf, B. P. (2009). *Building intelligent interactive tutors: Student-centered strategies for revolutionizing e-learning*. Burlington, MA: Morgan Kaufmann.

Wordnet. (2012). Wordnet: A lexical database for English., from http://wordnet.princeton.edu/

Yamada, Y., Kawabe, T. & Ihaya, K. (2013). Categorization difficulty is associated with negative evaluation in the "uncanny valley" phenomenon. *Japanese Psychological Research, 55*(1), 20-32.

Zhou, Y., Freedman, R., Glass, M., Michael, J. A., Rovick, A. A. & Evens, M. W. (1999, 1999 / 01 / 01 /). *Delivering hints in a dialogue-based intelligent tutoring system*.

Rapid, Form-Based Authoring of Natural Language Tutoring Trialogs

Benjamin D. Nye, Man Yang, Patrick Hays, Rodrigo Silva-Lugo,
Zhiqiang Cai, Mohammad Faisal Rahman, Xiangen Hu, Arthur C. Graesser
University of Memphis

Introduction

Time costs for authoring content have traditionally been a hurdle for Intelligent Tutoring Systems (ITSs). This paper introduces a novel form-based authoring tool for rapidly developing natural language intelligent tutoring dialogs and trialogs. This tool is designed to have a minimal learning curve and require no understanding of programming, formal logic, or rule-based systems. These tutoring scripts are based on the Shareable Knowledge Object (SKO) framework and the AutoTutor tutoring engine. Tutoring content is authored by instructors or content specialists in a step-by-step process through web-based forms, which are stored in cloud servers. These objects are processed to populate tutoring scripts that drive dynamic interactions with learners. Up to three trialogs can be authored to form a series (e.g., an introduction, main dialog, and exit dialog). These conversations can be either interactive tutoring sessions or vicarious dialogs (e.g., agents talking to each other). The SKO tutoring modules generated by this system can be delivered using AutoTutor Lite and AutoTutor Web services, which are integrated with recent versions of the GIFT framework. Challenges and opportunities for this authoring tool are discussed.

Authoring tools for ITSs are an important topic for reducing the cost and expanding the applications for ITSs. Considering that estimates for ITS content development can range up to 200 hours of development for a single hour of instruction, rapid authoring tools are vital for scalability (Koedinger et al., 2004). Prior work on authoring tools has found that simplifying the authoring process can result in tenfold increases in the speed of developing an hour of new content (Koedinger et al., 2004; Heffernan et al., 2006), though it could be argued that such tools do so at the expense of less powerful or less generalizable tutoring behavior. Reusability is a related issue: if the same content can be applied to multiple situations, then the ratio of hours of authoring to hours of instruction can be improved. This is particularly relevant for conceptual knowledge, where it may be important to emphasize the same skills or concepts across a variety of tasks for the learner to understand that they represent a general principle. So then, tutoring content could ideally be authored with different degrees of task-specificity: some ITS authoring may only be relevant to a specific problem, some might be relevant to all problems of one type, and some might even be relevant to all problems in a domain (e.g., basic numeracy for mathematics).

The goal of the authoring tool described in this paper is to rapidly develop tutoring conversations for natural language ITS content. A natural language tutoring system differs from a traditional problem-based ITS, in that users express all or part of their input in either speech and/or free text input and because the tutoring system responds back using speech and/or free text. In some respects, these requirements make natural language tutoring more complex: the correctness of student input is often less clear-cut in a verbal statement. These limitations are not limited to ITSs, but are also shared by human tutors, who may not understand a student's explanations or misconceptions (Graesser & Person, 1995).

On the other hand, these limitations have caused natural language tutors such as AutoTutor to focus on robust and fault-tolerant strategies for natural language understanding and feedback (Graesser et al., 2012). These include applying multiple semantic analysis techniques, such as latent semantic analysis (LSA) (Landauer et al., 1998), pattern matching, and frequency-weighted content word overlap (Cai et al., 2011). They also include discourse strategies such as trialogs, where the human talks with a computer tutor and a computer peer student (Millis et al., 2011). Trialogs enable interaction patterns where the computer student can react to potentially wrong answers by stating the misconception themselves, allowing the computer tutor to correct them. This pattern allows a human student's misconception to be corrected, but does not require the tutor to explicitly correct them. In general, AutoTutor focuses on trying to help students explain the expectations (ideal answers) for a conceptual question and also tries to correct any misconceptions that they address (Graesser et al., 2005). Interventions such as hints (leading questions), prompts (fill-in-the-blank type questions), and assertions (providing some or all of the answer) are used to help the student cover the key concepts.

Specializations for Authoring: Content vs. Logic

Due to its origins in discourse theory, AutoTutor differentiates between speech acts (the functional purpose a piece of dialog) and the specific speech and/or text used when delivering the utterance. To a significant extent, this decouples the logic that drives pedagogy from the specific content associated with ideal answers, misconceptions, hints, or other discourse acts: the production rules that drive an AutoTutor script seldom (if ever) explicitly refer to specific text or student responses (Graesser et al., 2005). This decoupling offers a significant opportunity: the same content for a dialog script can be delivered using different pedagogical rules for interactively tutoring the student.

Decoupling content authoring from logic and programming authoring lies at the heart of the authoring tool presented in this paper. This is an important division: as a general rule, experts in domain content are not experts in programming concepts (even simpler ones such as rule-based authoring). Since the reverse is also true (programmers are seldom experts in domain content), many authoring tools are only fully effective in the hands of interdisciplinary experts, such as specially trained graduate researchers. AutoTutor has an effective general-purpose tool called the AutoTutor Script Authoring Tools (ASAT), which allows a trained author to create a tutoring script in under an hour (Song, Hu, Olney & Graesser 2004). However, both the ease of authoring and the reusability of the script depend on the author's ability to conceptualize an effective rule set that drives tutoring: a novice might need dozens of rules where an expert might need only a handful. While authoring these rules does not require knowledge of a programming language, it does involve programming reasoning and concepts. While even laymen (e.g., recent high school graduates) can learn to use the tool fairly quickly, ideal content authors are subject matter experts and teachers. The time of such authors is valuable and often highly constrained, making this type of learning curve unpalatable.

The goal of this work was to develop a tool that allows an author to create natural language ITS content, without any knowledge of programming, to even at a conceptual level. A second AutoTutor authoring tool suite called AutoTutor Lite authoring tools, already allows basic authoring without programming knowledge (Hu et al., 2009). However, the AutoTutor Lite tool has a large number of nested menus and features that increase the learning curve and make simple authoring tasks more difficult. The authoring tools described here simplify the process further, building a step-by-step set of forms to guide the author through creating the tutoring content.

In the following sections, we describe a form-based authoring tool for natural language tutoring trialogs. This tool authors the content for the tutoring trialog only, using a simple step-by-step authoring process. This tutoring content is then combined with a set of production rules from an ASAT template, which determine how this content is interactively tutored. This research builds on work with ASAT, which has been looking into generic templates that drive different pedagogical strategies. Research on such templates is still in its early phases. However, even a single effective and generic template can be a powerful tool. As such, this splits the authoring process into separate phases: one for experts in subject matter and the other for experts in rule-based pedagogical strategies. It is hoped that this approach should accelerate the authoring process by matching tool specializations to user specializations.

Form-based Authoring Tool Design

The intended author for the new form-based authoring tool has high domain knowledge and moderate pedagogical knowledge, and requires no knowledge of programming concepts. Murray (2003) lists six main design tradeoffs for tutoring: breadth, depth, learning curve, productivity, fidelity, and cost. This research focuses on optimizing for the breadth (variety of topics), learning curve, productivity, and cost. To keep the learning curve at a minimum, a sequential form-based approach is followed. This approach has the advantage that a new author should be able to easily populate new content and learn the basic functionality. The tool is also designed for fast input (e.g., limited mouse clicks and the ability to quickly enter many pieces of content). This should allow fast production of content. With that said, being a series of forms, it is not optimized as much for maintenance and viewing of the tutoring content. As a dialog-based authoring tool, a wide variety of topics can be tutored, translating to high breadth of coverage.

This tool does not focus on optimizing fidelity, depth, or ITS modules other than domain content. Fidelity is hard to qualify in this context: while the natural language tutoring is intended to emulate real human-to-human tutoring, conversational tutoring typically talks *about* topics rather than emulating them (as opposed to a simulation, for example). However, in domains such as reading comprehension, understanding natural language is also a central part of the domain. This form-based research also does not focus on the depth of tutoring, such as by allowing arbitrarily complex knowledge structures to drive conversations. AutoTutor shows strong gains for deep learning and concepts (Graesser et al., 2005), but this authoring tool does not address complex authoring scenarios, which are left to the ASAT tool. Instead, the knowledge representation for the form-based tool is one-size-fits-all, with the understanding that different pedagogical rule sets may process it quite differently. Finally, it is notable that the form-based authoring tool only provides functionality for authoring domain content. Pedagogical strategies, student models, and communication through user interfaces and learning environments are not authored using this tool. This research assumes that pedagogical strategies are handled by ASAT authoring and that other modules (student model and communication) are developed primarily by programmers.

Content data model

The data model for this tutoring content is based around AutoTutor's discourse moves, which is a set of speech acts that tutors perform (Graesser et al., 2005). The core discourse moves include 1) Main Questions, which are a question that starts off the dialog, focused on a particular topic or goal; 2) Pumps, which ask the student to provide more information; 3) Hints, which are leading questions or statements that attempt to direct the student to answer part of the main question; 4) Prompts, which lead the student to express a missing word from an important idea for the main question (e.g., "Is it moving up or down?"); 5) Short Feedback, which signals about the quality of the student's last statement; 6) Corrections, which correct a misconception or incorrect statement by the learner; 7) Assertions, which present an important idea within the problem or the answer to the problem; 8) Answers, which are responses to questions about a concept; and 9) Summaries, which present a full answer to the main question. In addition to these, a full AutoTutor script includes information about expected student responses (e.g., good answers vs. bad answers) that are associated with certain conversational states.

To keep the authoring process streamlined, some constraints and assumptions were designed into the authoring tool. First, not all speech acts are authored using this tool. The form-based tool focuses exclusively on main questions, hints, prompts, corrections, assertions, and summaries. The remaining speech acts were omitted because they tend to be more general and reusable. Pumps, such as "Can you say a little more about that?", can be designed in a domain-agnostic fashion and included as part of a template. Likewise, short feedback can be drawn from a pool of general statements (e.g., "Great!"). AutoTutor student question and answers were not included in the tool because students seldom ask questions and, when questions are asked, they could be touch on a wide variety of concepts that are not the main focus of the current session. As such, these may also be better suited for being handled by a domain-level template rather than ad-hoc for each conversation.

This tool allows the author to create the six types of speech acts noted. For example, assume that a speech act for a main question is created. To deliver this question, the system needs sufficient information to know what to say (the text of the question), who should say the question (e.g., the computer tutor vs. computer student), and also some expected responses to help evaluate human input (e.g., good and bad answers). Notably, this tool, unlike ASAT, does not distinguish between authoring the agents' speech acts and the expected human student responses. Instead, the agents' speech acts are also used to populate data about expected human responses (e.g., good answers, bad answers, misconceptions, etc.). While this adds extra burden on the semantic analysis, it also reduces the number of authored fields by up to half.

Two types of conversations can be authored using the tool: vicarious and tutoring. Figure 1 shows a high-level view of how each type of content is authored as part of larger content objects. To note, this is not a representation of how the data is stored or used by AutoTutor, but is a conceptual grouping used to design the tool. Vicarious conversations (Figure 1, left) do not react dynamically to the human student, and typically shows a computer tutor tutoring a computer student (Craig et al., 2006). While the human can be addressed during this type of tutoring (e.g., "Do you think he's right?"), the human response does not impact the flow of the conversation. As such, vicarious tutoring is also sometimes called information delivery or a "rigid" pack. Vicarious tutoring is particularly helpful for low-knowledge learners.

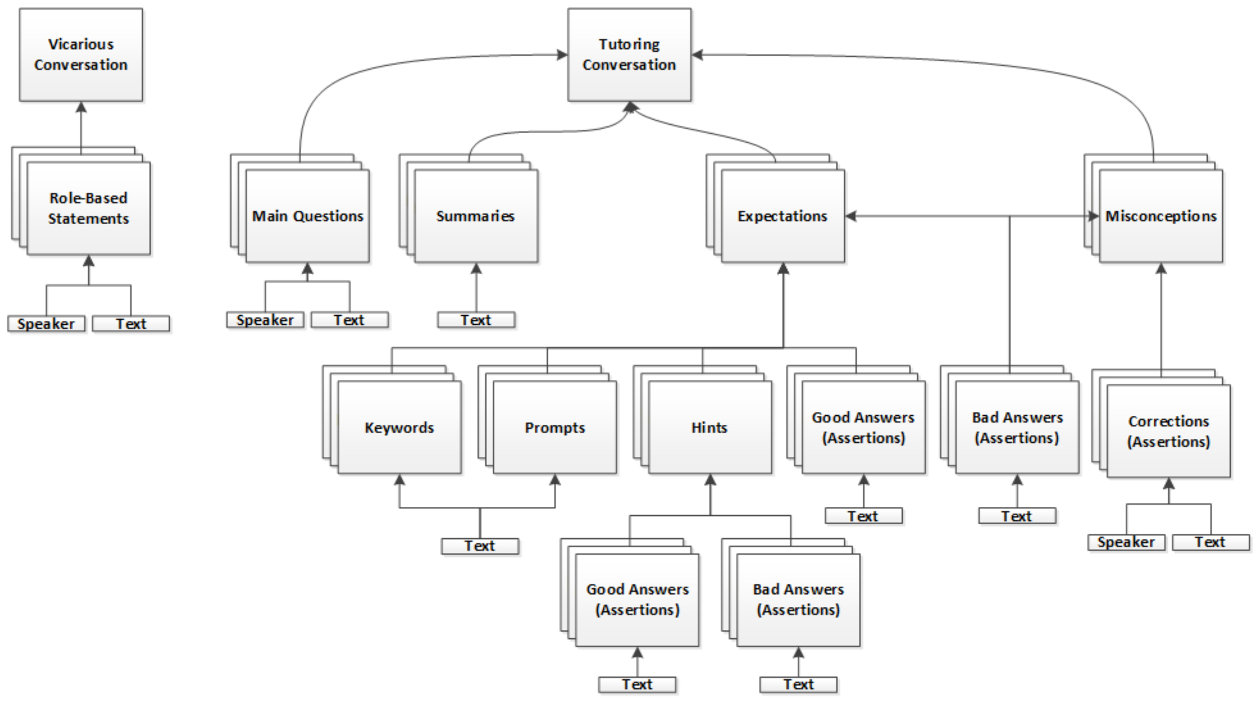


Fig 1. A high-level view of the content model for the form-based authoring tool that shows vicarious conversations on the far left and tutoring conversations on the right.

On the converse, Tutoring conversations (Figure 1, right) are determined by the content of human responses and require the full set of speech acts noted previously. As shown in this figure, the high-level components are the main questions, expectations, misconceptions, and summaries. Main questions are authored as role-based statements: who says them and what they say. This allows different levels of formality (e.g., computer tutor vs. computer student) and emphasis. Summaries are the full, complete answer to the question. These consist of one or more main ideas, which are used to develop the expectations. Expectations are authored by creating only their answers (good and bad), hints, keywords, and prompts. Misconceptions are constrained even further and include only an assertion that would indicate that misconception and corrections, in the form of a role-based statement to clarify the misunderstanding. In the AutoTutor data formats, AutoTutor scripts represent expectations and misconceptions using the same data structure, but they are authored quite differently in this tool. This simplification was done for pedagogical reasons, since encouraging students to explain their misconceptions can sometimes reinforce those misconceptions. While experienced authors can navigate such nuances, this level of authoring complexity would not be in line with a minimal learning-curve.

Form-based authoring process

Dialog creation starts with the SKO Wizard Panel, shown in Figure 2. SKOs are containers for tutoring content and web service information, which are stored in cloud-based storage. The author first determines the type of dialog they want to create, and the system will display a border on the author’s selection. The system will create a dialog group object in a cloud-based storage and associate this group with a unique identifier. Each individual dialog object in this group is also associated with a unique id. When the edit button is clicked, the author will begin adding and editing content that is stored inside the dialog object that is currently highlighted in the wizard.

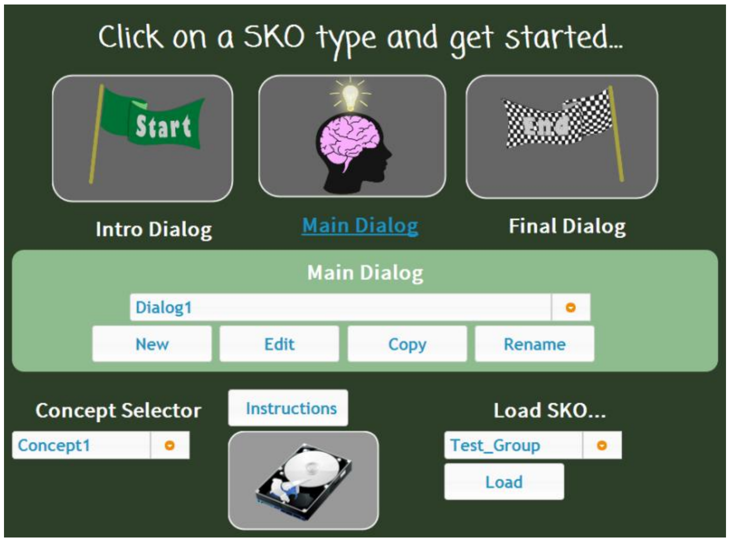
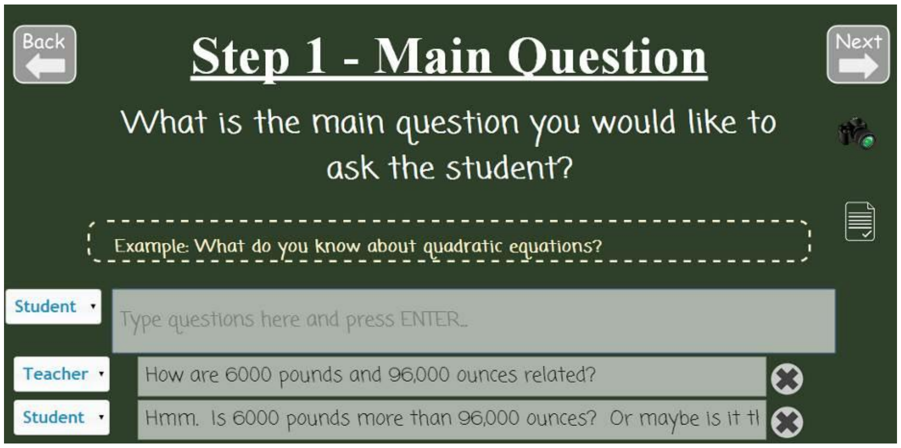


Fig 2. The SKO Wizard Panel is used to author and edit up to three related dialogs that work as a group (e.g., Intro, Main, Closing) and associate them with a concept.

Authoring a dialog occurs through step-by-step forms. Two types of conversations are possible: vicarious or tutoring. The type of dialog is selected using a panel containing a simple drop-down (not shown). The type of dialog selected determines the next forms that are displayed. Vicarious dialogs are very simple to author, but do not take advantage of AutoTutor's intelligent, interactive tutoring. All authoring for a vicarious dialog is performed in a single form, after which the author is returned to the SKO Wizard Panel.

Authoring tutoring conversations is more complicated and requires a number of steps. The first step is to choose a main question to start the overall dialog (Figure 3). Multiple ways of asking this question may be created, which can be helpful if the learner does not understand the question initially and the system needs to restate it. Next, the main ideas contained in an ideal response to that question are authored. These main ideas are each an ideal answer for an AutoTutor expectation (Figure 4). In addition to entering a main idea, the author can associate specific keywords with that main idea. These keywords must be stated by the learner to fully answer the main question. When the author clicks next an expectation will be generated, and contents of later pages (good/bad answers, hints, and prompts) will be stored as sub-components of the current expectation. Once an expectation is filled out, the form repeats each step for the next expectation, until all expectations are completed for the main question.


Back

Next

Step 1 - Main Question

What is the main question you would like to ask the student?

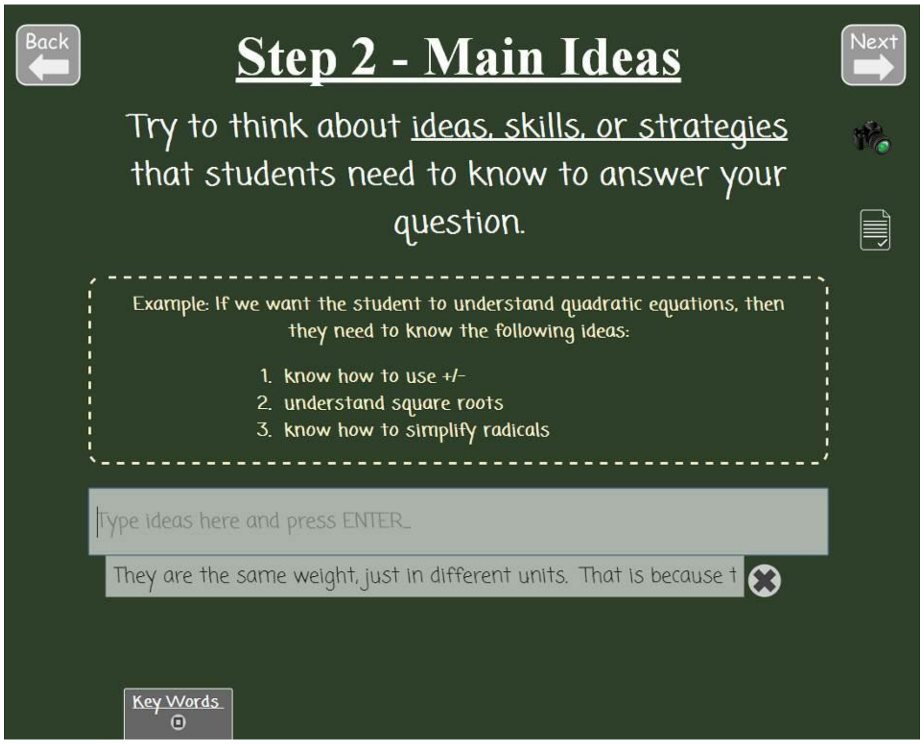
Example: What do you know about quadratic equations?

Student | Type questions here and press ENTER.

Teacher | How are 6000 pounds and 96,000 ounces related?

Student | Hmm. Is 6000 pounds more than 96,000 ounces? Or maybe is it tl

Fig 3. The author writes one or more ways to say the main question that will drive the conversation.


Back

Next

Step 2 - Main Ideas

Try to think about ideas, skills, or strategies that students need to know to answer your question.

Example: If we want the student to understand quadratic equations, then they need to know the following ideas:

1. know how to use +/-
2. understand square roots
3. know how to simplify radicals

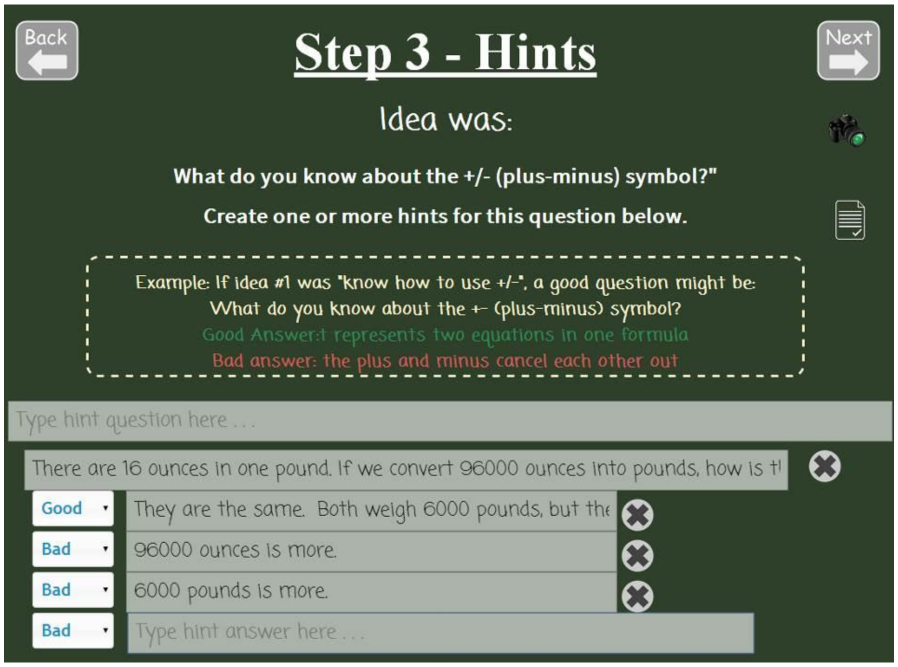
Type ideas here and press ENTER.

They are the same weight, just in different units. That is because t

Key Words

Fig. 4. Main ideas are each a distinct part of an ideal summary to the main question.

After this page, there are two important components of dialog we must create: hints and prompts (Figures 5 and 6, respectively). When the author types a hint in the top-most entry box and clicks enter, a hint object is generated. Hints are leading questions that try to help the learner fully answer the main idea for the current expectation. For each hint, the author can input multiple good or bad answers. The good answers are used to compare against the learner's response to the hint. These answers are also used by the computer tutor and computer student, such as the computer student proposing a wrong answer and the computer tutor correcting them. The specific logic driving these interactions depends on the ASAT template rules. Next, the author creates prompts for the dialog. Prompts help the user cover a specific keyword by asking them a specific question to help elicit it. The author types in a prompt and hits enter. Each prompt is shown in the list below, where the author can select a keyword from the dropdown to its right. This set of keywords is populated from the keywords entered in the Main Ideas panel (see Figure 4). On this page, the author can also open the Keywords control to edit this set of keywords.


Step 3 - Hints

Idea was:

What do you know about the +/- (plus-minus) symbol?"

Create one or more hints for this question below.

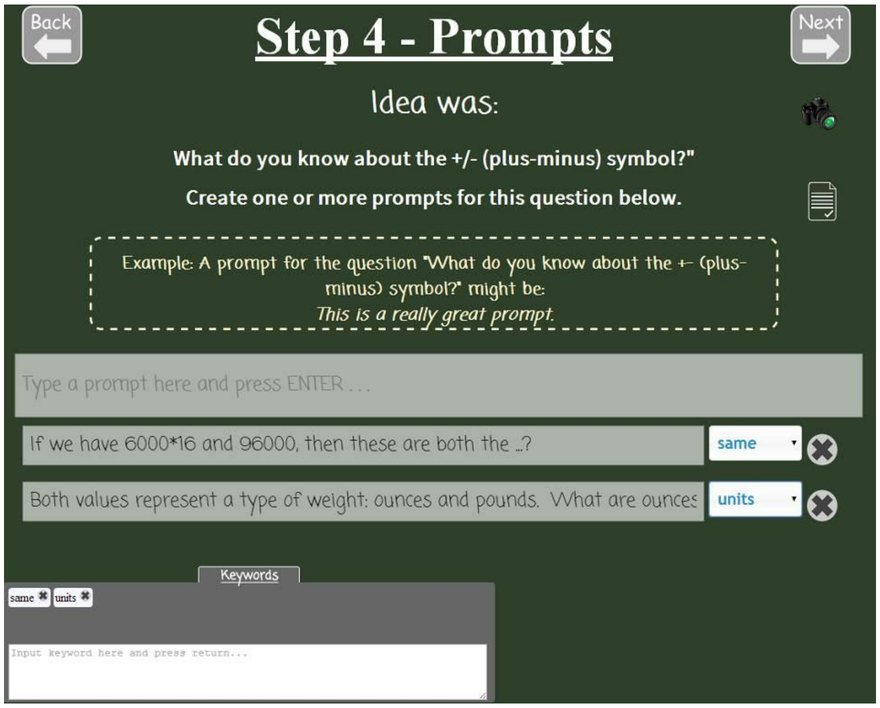
Example: If idea #1 was "know how to use +/-", a good question might be:
What do you know about the +- (plus-minus) symbol?
Good Answer:t represents two equations in one formula
Bad answer: the plus and minus cancel each other out

Type hint question here . . .

There are 16 ounces in one pound. If we convert 96000 ounces into pounds, how is t

Good	They are the same. Both weigh 6000 pounds, but the
Bad	96000 ounces is more.
Bad	6000 pounds is more.
Bad	Type hint answer here . . .

Fig 5. The author creates hints that help the learner cover the expectation's main idea.


Step 4 - Prompts

Idea was:

What do you know about the +/- (plus-minus) symbol?"

Create one or more prompts for this question below.

Example: A prompt for the question "What do you know about the +- (plus-minus) symbol?" might be:
This is a really great prompt.

Type a prompt here and press ENTER . . .

If we have 6000*16 and 96000, then these are both the _?	same
Both values represent a type of weight: ounces and pounds. What are ounces	units

Keywords

same units

Input keyword here and press return...

Fig 6. The author writes prompts that help the learner cover keywords for a main idea.

After finishing the prompts, the author has completed an expectation. If additional expectations have already been authored, the next expectation will be loaded for editing. If this was the last expectation, the author will be asked if they wish to add an additional expectation (which would also start at the Main Questions panel). If they do not wish to add another expectation, the Misconceptions Panel will open. In this authoring tool, a misconception is authored as a single wrong answer that is associated with one or more correction responses. This panel is similar to the hints panel, in that the author first writes a misconception (e.g., "Perpendicular lines never intersect."). After pressing enter, a group is created in the authoring tool that allows them to author corrections, similarly to how they can provide multiple answers to a hint. However, unlike hints, misconceptions do not have bad answers. Instead, the author can write different ways the computer tutor or computer student would correct this misconception.

After misconception authoring is completed, the author will be presented with an Outline Panel with a summary of the tutoring content (the first main question, the list of main ideas, etc.). From the Outline Panel, the author will be able to return to specific authoring steps to edit content. This final panel is still under development. Once the Outline Panel is finished, it will be available for viewing at all steps of the authoring process, allowing the author to easily skip to editing various components of the tutoring content. A second panel under development is the Media Panel, which will allow associating pictures and videos with specific statements by the computer agents. After the author has reached the outline panel, the Next button will return them to the SKO Wizard Panel.

Challenges and Future Directions

This tool is approaching its first prototype and associated usability testing. A significant challenge for this tool has been the issue of hierarchical object creation and management across multiple HTML pages. While the entire content authored for a tutoring script is stored as a single object, its authoring is distributed across an arbitrary number of HTML pages. At present, the entire object is saved after editing each part. However, this has significant drawbacks for a cloud-based tool, since multiple authors might wish to work on different parts of the same dialog at one time and could overwrite each other's changes. Version control systems and partial-update patterns are being considered as possible solutions to this problem. A related difficulty is that this tool does not allow viewing or editing certain tutoring content that is accessible in a tool such as ASAT. As such, translation between the tools must be lossy (e.g., the form-based tool clears fields that it cannot see) or partly opaque (e.g., the form-based tool cannot see certain changes).

Expanding the number of templates is another challenge. At present, a single template set is combined with the authoring tool content that is produced. In the future, more templates will be added so that a wider variety of pedagogical strategies can be employed. Significant debate exists about the balance of allowing authors to specify the templates employed, as opposed to dynamically triggering different pedagogical rule sets using a student model or other features. It is expected that authors will be able to select templates, with the understanding that in some contexts these selections may be used as suggestions, rather than followed consistently. For example, swapping out templates dynamically might be used to help evaluate their effectiveness for different learners, such as Mostow and Beck's (2006) recommendation to randomize interventions to support effective data mining.

Finally, once this tool is tested and mature, a version is expected to help support AutoTutor script development required by the Generalized Intelligent Framework for Tutoring (GIFT) system (Sottilare et al., 2012). Recent releases of GIFT have integrated elements of the AutoTutor framework for delivering conversational tutoring, which must currently be authored using ASAT or the AutoTutor Lite web-based authoring tool set (Hu et al., 2009). This form-based tool should hopefully offer a third method that allows rapid authoring with a minimal learning curve. With testing and refinement, this tool should be a useful piece of the AutoTutor authoring process. Particularly when considering the wide range of potential authors, this approach to form-based authoring seems particularly relevant to a system such as GIFT, as it can be applied to a wide variety of topics and author experience levels.

Supporting the needs of GIFT authors will have challenges, however. First, unlike traditional ASAT scripts, form-based content must be combined with an ASAT rule template. This requires automated preprocessing, so scripts must either be pre-processed before storage (i.e., generate and store a full ASAT script) or a specialized storage service must do this processing on the fly. It is unclear which approach would be most appropriate for GIFT's storage needs. Second, pedagogical strategies rely on ASAT rule templates. As such, to edit or create new strategies, authors still need to learn the ASAT authoring tool. While a visual, flow-chart style of authoring ASAT rules has recently started (ASAT-V), creating templates currently requires authoring production rules. ASAT is also a desktop application, adding additional installation dependencies. Finally, the form-based tool needs requires scaffolds to help design scripts that process human input effectively. For example, ASAT uses regular expressions as a more powerful alternative to keywords in the script content. While it is possible for authors to enter a regular expression instead of a keyword into the form-based tool, it is unlikely that a non-technical author will understand how to do this. Additional interface controls may be required to help authors build common types of regular expressions that are necessary to accurately understand human natural language input. Initial testing without regular expressions has found that the accuracy of parsing human input is sometimes suboptimal. AutoTutor's use of LSA (Landauer et al., 1998) also has authoring implications, since the answers to main questions, expectations, and hints should ideally be aligned semantically (i.e., answers to hints help cover the main ideas). If a script does not do this, the learner could answer all the hints correctly but still be treated like they did not fully answer the main question. Additional design will need to make it easier to diagnose problems with scripts whose semantics match poorly, from the standpoint of the LSA space. Despite the challenges to refine this tool for a GIFT's wider user base, we feel believe that a form-based tool should make authoring simpler and more accessible. Hopefully, with additional development, we can achieve that goal.

References

Cai, Z., Graesser, A. C., Forsyth, C., Burkett, C., Millis, K., Wallace, P., Halpern, D. & Butler, H. (2011). 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.

Craig, S. D., Gholson, B., Brittingham, J. K., Williams, J. L. & Shubeck, K. T. (2012). Promoting vicarious learning of physics using deep questions with explanations. Computers & Education, 58(4), 1042-1048.

Graesser, A. C., Chipman, P., Haynes, B. C. & Olney, A. (2005). AutoTutor: An intelligent tutoring system with mixed-initiative dialogue. Education, IEEE Transactions on, 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. & Magliano, J. P. (1995). Collaborative dialogue patterns in naturalistic one-to-one tutoring. Applied Cognitive Psychology, 9, 1-28.

Heffernan, N. T., Turner, T. E., Lourenco, A. L., Macasek, M. A., Nuzzo-Jones, G. & Koedinger, K. R. (2006). The ASSISTment Builder: Towards an Analysis of Cost Effectiveness of ITS Creation. In FLAIRS Conference (pp. 515-520).

Koedinger, K. R., Aleven, V., Heffernan, N., McLaren, B. & Hockenberry, M. (2004, January). Opening the door to non-programmers: Authoring intelligent tutor behavior by demonstration. In Intelligent Tutoring Systems (pp. 162-174). Springer Berlin Heidelberg.

Landauer, T. K., Foltz, P. W. & Laham, D. (1998). An introduction to latent semantic analysis. Discourse Processes, 25(2-3), 259-284.

Millis, K., Forsyth, C., Butler, H., Wallace, P., Graesser, A. & Halpern, D. (2011). Operation ARIES!: A serious game for teaching scientific inquiry. In Serious games and edutainment applications (pp. 169-195). London: Springer.

Hu, X., Cai, Z., Han, L., Craig, S. D., Wang, T. & Graesser, A. C. (2009). AutoTutor Lite. In V. Dimitrova, R. Mizoguchi, B. Du Boulay & A.C. Graesser, A.C. (Eds.), Proceedings of Artificial Intelligence in Education (AIED) 2009 (p. 802-802). Amsterdam: IOS Press.

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.

Mostow, J. & Beck, J. (2006). Some useful tactics to modify, map and mine data from intelligent tutors. Natural Language Engineering, 12(2), 195-208.

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 The Interservice/Industry Training, Simulation & Education Conference (I/ITSEC), 2012(1). National Training Systems Association.

Theme VII: Intelligent Tutoring for Teams

Taxonomy of Teams, Team Tasks, and Tutors

Desmond Bonner[1], Stephen Gilbert[1], Michael C. Dorneich[1], Shawn Burke[2], Jamihus Walton[1], Colin Ray[1], Eliot Winer[1]
[1]Iowa State University
[2]University of Central Florida

Introduction

While significant research has been done on teams and teaming (Salas, et al. 2004), less work has been done to characterize teams and team tasks in terms of the feasibility for them to benefit from intelligent tutoring. This theoretical paper begins to describe how the parameters of team structures addressed may affect the ways in which a team can accommodate external guidance. In addition, parameters of team tasks and resulting team tutors are also described. Examples of both team structures and team tasks are provided so that the resulting theoretical framework offers guidance for design decisions during the construction of intelligent tutoring systems (ITSs) for teams and the Generalized Intelligent Framework for Tutoring's (GIFT) supporting team architecture.

ITSs have been successful at improving performance in a wide variety of domains ranging from academic topics such as math (e.g., Koedinger, Anderson, Hadley & Mark, 1997) to work-based tasks such as management of power plants (Faria, Silva, Vale & Marques, 2009). However, there have been few ITSs designed for educating or training teams (Sottilare, Holden, Brawner & Goldberg, 2011). Despite much research on teaming since the 1970s, team performance is widely variable and difficult to predict (Sims & Salas, 2007), and there is a significant need for team-based ITSs. A taxonomy of team tutoring is presented (see Figure 29 for top level key elements). This paper describes three taxonomies: teams, team tasks, and relevant tutoring factors. The taxonomies are based on reviewing the teaming literature with a particular focus on the characteristics of each that would influence the design of a team-based intelligent tutoring system. This work leverages the extensive literature review of teaming by Burke et al. (in progress) as well as recent work that has sought to identify those major factors which impact team performance Salas, Shuffler, Thayer, Bedwell & Lazzara (in press).

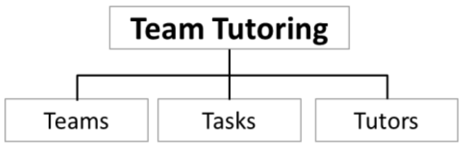


Figure 29. The three key elements of a team tutoring taxonomy.

The taxonomies provided below are designed to help guide the design of software architecture to support team ITSs within GIFT. GIFT is a powerful software architecture designed to support a wide spectrum of intelligent tutoring. It supports the traditional components of most ITSs: the learner model, the domain model, the pedagogical model, and the learner interface, but does so generically (Sottilare, Brawner, Goldberg & Holden, 2012; Sottilare, Graesser, Hu & Holden, 2013). Thus, a multitude of learners might manipulate a wide range of user interfaces as they engage with various domains while being taught using a variety of pedagogies. However, the GIFT architecture does not naturally support teams. Team components are necessary if GIFT is to support team tutoring, but they are not present in the current release. In their 2011 paper, Sottilare et al., the creators of GIFT, describe the challenges of creating team tutors in detail.

Team Tutor Taxonomy

The three key elements of the taxonomy are described below, followed by examples and implications for the GIFT architecture. The first key element in the taxonomy is based on team characteristics. This element of the taxonomy notes the factors that can vary within teams that would have an impact on the design of a team tutor. The second key element in the taxonomy is based on the characteristics of team tasks. The final key element in the taxonomy focuses on team tutors itself, and the potential features that must be considered.

Teams

Characteristics of teams and team members that are important to consider when designing a team tutor can be seen in the taxonomy in Figure 2. Parameters of team structures such as leadership (e.g., vertical, shared, mixed, leaderless, and confederate), organization, communication styles, and location (e.g., co-located or distributed, asynchronous or not) affect the ways in which a team can accommodate external guidance such as tutoring feedback**.** The leadership structure of a group varies. For instance, there may be a shared responsibility, or the team may use a leaderless structure in which members are accountable primarily for their own roles and/or responsibilities. The organization of a team can dictate collaboration as well. Communication styles can determine how groups coordinate responsibility and facilitate collaboration, which eventually affects performance.

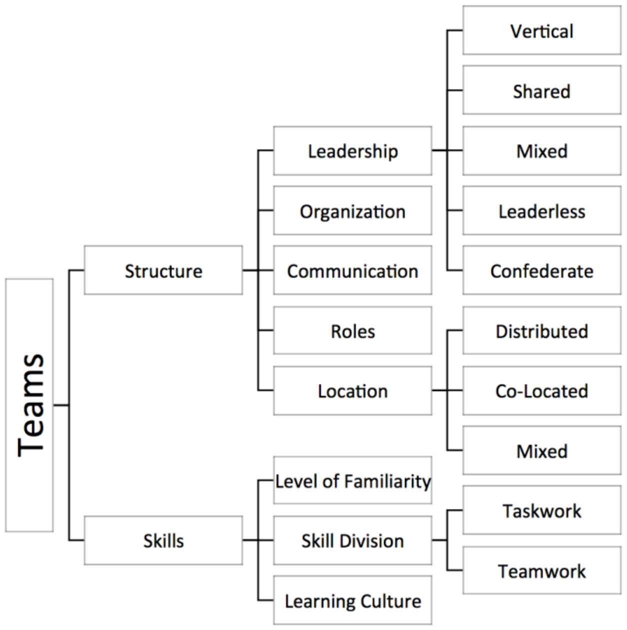


Figure 30. Characteristics of teams and team member characteristics.

When assessing teams, it is also important to consider the team's purpose as well as the members' roles (Salas et al. 2004, Sottilare et al. 2011). Different teams have unique rosters including special roles as well as varying levels of expertise. For example, when comparing an infantry squad with a combat engineer squad, both consist of team members who have a contribution appropriate to the squad's purpose or overall goals. An infantry squad will be better trained to handle a fire fight while the combat engineers would be the preferred choice to construct bridging for heavy vehicles over rough terrain. Team skills are parameters of teams that further support the team's purpose. Differing from roles, skill division refers to the distribution of competencies possessed by team members that they can apply towards goal completion. These competencies may relate to the task or to teaming itself. Within teams there can also be varying levels of familiarity. How long the team members have been working together can influence a team's success. Finally, within a team there may exist a learning culture in which individuals contribute their knowledge to other members of the team as the task requires, which ultimately makes the team more effective.

In the creation of a team intelligent tutor, it is critical to consider these team variables. A tutor designed for highly vertical leadership structure, for example, would likely offer differentiated feedback for members at different levels. For a leaderless team, the feedback would have to be structured more for peers. In general, the timing, structure, and level at which feedback is given (to the individual vs. the whole team) may differ based on the characteristics described in the taxonomy as well as on the team's current developmental trajectory.

When appreciate the implications for the GIFT architecture, consider the above team taxonomy with the only the elements that is relevant to individual tutors: the skills. Structure is not relevant without the other team members, and the subcomponents of skills are also not relevant (e.g., level of familiarity). This significant difference in the complexity of the taxonomy means that GIFT must be able to maintain the state of many more elements of a training mission: skill state for each team member, skill state of the team as a unit, and all the parameters of the team structure. This more complex set of information can be then used to make decisions such as giving feedback to individuals vs. to the whole team.

Tasks

The taxonomy of task characteristics in Figure 31 is again an initial attempt to establish a checklist of key factors that must be considered in the design of a team-based ITS. Tasks are the activities teams perform. In the context of this research, the tasks would be performed within a training environment with the purpose of providing a detailed assessment. The task at hand has a strong influence on how an intelligent tutor is designed, and with team tasks, the numerous roles and relationships between members' tasks can make the tutor design much more complex than the sum of analogous tutors for individuals. Factors that influence tasks as a whole are solutions, task interdependence, routine, complexity, time constraints, information exchange, environmental fidelity and task type (Figure 2). Solutions refers to how tasks are negotiated and solved. There may be a single solution, multiple solutions, or the task may be open-ended, with no particular solution planned. Training tasks focused on reaching a specific outcome typically have a small number of solutions, while training focused on a process is typically more open-ended. In some tasks designed to teach compromise and negotiation, the scenario may be specifically designed so that there is no solution.

The interdependence of tasks is also a key factor. Task interdependence consists of four types: pooled, sequential, reciprocal, and team interdependence (Saavedra, Earley & VanDyne, 1993). Pooled interdependence occurs when members of a group complete similar tasks mostly independently, such as a team of customer service support agents. With sequential interdependence, members all perform separate but related tasks in a specific order to accomplish a greater goal, e.g., in an assembly line. Members' tasks depend on the work of previous members. With reciprocal interdependence, members' tasks mutually depend on each other's work, and the order is flexible, e.g., the writing of a paper between co-authors. Lastly, team interdependence is the most dynamic structure, in which each member's tasks and the particular sequence of work are unspecified and are decided within the team.

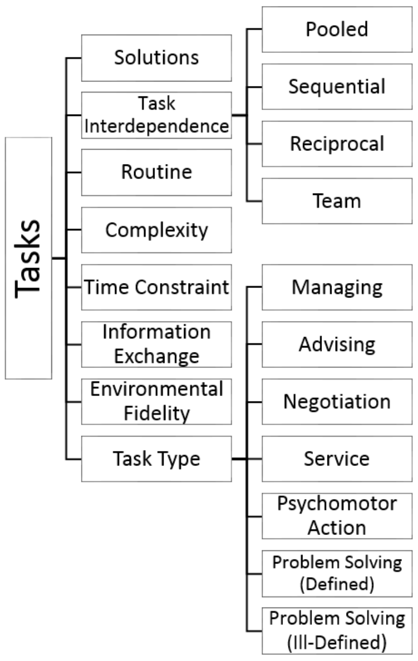


Figure 31. Characteristics of team tasks.

Furthermore, it is important to characterize the variability of the routine and the task complexity, because both affect the learning curve and overall performance. Tasks become less strenuous or challenging as they become familiar to the learner. For instance, in the process of two people stocking the back room of a coffee shop, the members may at first become overwhelmed with the complexity of the task and figuring out how to work together, but over time the process becomes less complex as the members build a shared mental model of the equipment, the task, and their team members, which allows them to efficiently coordinate. The cognitive load is reduced as the task becomes familiar. A tutor for a team must be able to accommodate these team learning curves.

Time pressure typically occurs as a result of a set time constraint, although time constraints do not always lead to time pressure (Ordonez & Benson, 1997). It can also be caused by a factor of increased workload. The amount of time pressure felt by participants in an exercise may positively or negatively affect the team's performance, and can at times be an effective motivator (Andrews & Farris, 1972; Stewart, Lam, Betson, Wong & Wong, 1999). Information exchange within the current context is related to team communication, but refers to the forms of information that are generated by the systems used within the task, the user interfaces used by the team members, and how easily the information may be shared. Environmental fidelity, in a tutor design context, is the degree to which the team task is conducted in the actual context of practice (e.g., the "field") as opposed to within a lab or simulation. The necessary degree of fidelity varies according to the skills being trained. The feedback given to a team and its members by a team ITS might not differ in the field vs. in the lab, but the thresholds for feedback activation might vary, since team members in the field might be experiencing a higher levels of stress.

Task types include, but are not limited to, managing, advising, negotiating, performing a service, problem solving, and psychomotor action (Wildman, Thayer, Rosen, Salas, Mathieu & Rayne, 2012). Managing tasks involves directing, supervising, or overseeing the work of others in an authoritative role for the specific purpose of achieving a goal despite given conditions. Meanwhile, advising refers to providing means to solve a problem remotely via professional support through expert assistance in a consultative role where the advisor does not have managerial. A service task is a social interaction in which another individual or group. Negotiating involves coming to a compromise or successfully overcoming differences. Psychomotor action is the operation of a product, machine, or object. Problem solving can refer to solving ill-defined or novel problems.

One implication of this task taxonomy for GIFT is the needed support for a wide variety of methods of monitoring individual and team performance—not only the typical individual performance with a system but also the communications between team members, and the actions of the team as well as actions of individuals. Also, whenever a team is at work on a task, there is the metatask of performing well as a team. GIFT will need to support these parallel tasks simultaneously. For example, it's possible that a task is being accomplished well, but the team members doing it don't trust each other and are not collaborating well. GIFT will need to be able to tutor on both the task work and the teamwork.

Team Tutor Structure

Once the team structure is defined and the task is chosen, the team tutor can be initiated. However, there remain a number of choices in tutor design. This section describes a taxonomy (Figure 32) of the variables that characterize team tutors in particular. Tutors are broken down into categories consisting of feedback, pedagogy, adaptivity, environmental context, and evaluation. The subcategories of these variables are not intended to be exhaustive; there are additional options under each, but these categories offer a list of decisions that must be made during tutor design.

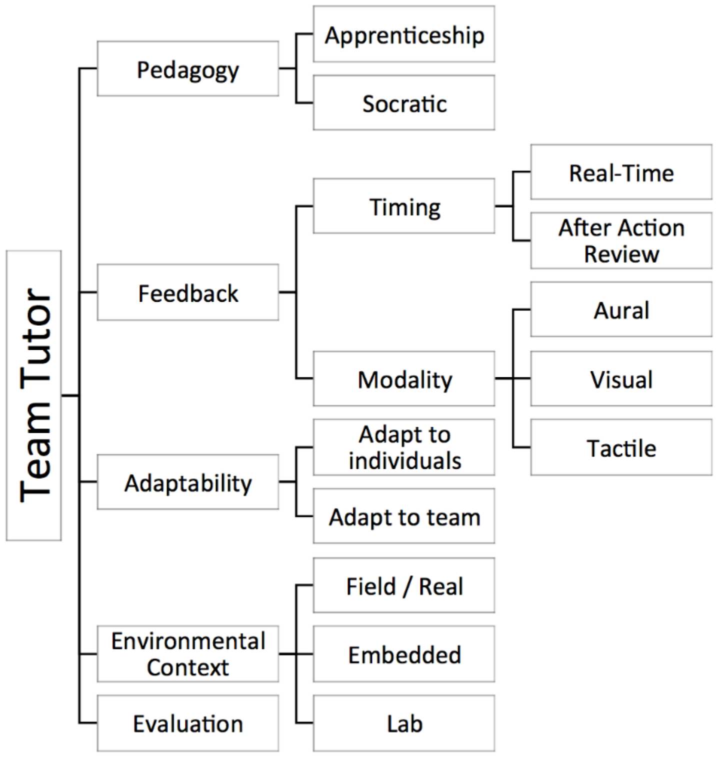


Figure 32. Characteristics of team tutors.

The pedagogical decision includes options such as apprenticeship scaffolding, in which the tutor guides the learner to mastery in a series of successive approximations with less and less scaffolding. A Socratic dialogue, on the other hand, would promote reflection by the learner in a series of questions and answers. Both of these are common approaches in existing one-on-one ITSs, but are complicated when tutoring a team. Performing Socratic dialogue at a team level requires interrupting the team (e.g., event-based training (Fowlkes, Dwyer, Oser & Salas, 1998)), waiting until the task is complete and doing an after action review (AAR), or offering the team advice when the team decides on its own to pause and consult the tutor.

Feedback is also an important element of tutoring. Individuals are motivated upon receiving feedback as it informs them of their progress and directing the learner towards a desired behavior. In terms of timing, real-time feedback provides the learner instruction without delay and avoids the development of inaccurate cognitive structures, behaviors, and attitudes that could be difficult to mitigate later. A primary challenge with this approach is that it may interrupt action or team communication, adding to members' cognitive load. Thus, after AARs are favorable when evaluating learner's progress through the entire task scenario. The tutor may observe the team's performance without interfering with the task. Feedback modality can also vary, and must be considered based on the cognitive loading of the tasks. If the learners have visual tasks, for example, feedback should probably arrive in a different modality such as audio. If team members are moving during the task, strong vibrotactile feedback might be useful. See the companion paper in this volume (Walton et al., 2014) for a further look into feedback for team tutoring.

The adaptability of the tutor is also a design decision. A simple tutor will contain mainly a set of static feedback responses that are triggered by behavioral predicate rules: If the learner does X, display feedback Y. A more complex tutor will make the conditional of that rule more complex, changing the feedback depending on the learner's history and accumulated skills, e.g., "If the learner does X, and has low skill ratings on skills A, B, or C, then display feedback Y." A yet more complex adaptive approach is changing the consequent of the rule, changing the feedback itself based on learner behavior or conditions within the training environment, e.g., "If the learner does X, and has a history of other actions A, B, C, then give feedback Y(A, B, C, X)." Finally, the tutor may also adapt to overall team behaviors, which could become inputs to both the conditional and the consequent, e.g., "If 70% of team members do A and B, then give the entire team feedback Y(A, B)." The balance between individual vs. team feedback depends on many of the characteristics described above. DeShon, Kozlowski, Schmidt, Milner, and Wiechmann (2004) offer a brief overview of this issue.

The tutor may vary based on environmental context. In the field, the tutor must consider a broader variety of states within the training scenarios; it may not have specific feedback for all possible actions. In the lab, or an embedded context (a simulated training scenario is embedded within a more realistic setting in the field), the tutor authors have more control over the scenario and its possible states, and can build a more exhaustive tutor.

Finally, the evaluation approach of a team tutor may vary significantly based on the task and the learning goals. However, the mechanics of the evaluation affect the design of the tutor. For example, if performance evaluation is based on direct, accurately measurable actions by learners, the evaluation is simpler than if the learner's performance must be inferred from a variety of indirect cues, such as biometrics.

The implications for GIFT are significant. Most notably, GIFT must be able to consult multiple individual learner models and a team learning model when choosing its next pedagogical action. Also, GIFT must be able to support the increased complexity of scoring rules that 1) take multiple individuals' states into account, and 2) take team roles into account, e.g., "If the pilot-learner has done X and said Y to the co-pilot-learner, but not said Z to the crew-learners, and the co-pilot-learner has done A but not B, then give feedback F1 to the pilot-learner, F2 to the co-pilot-learner, and F3 to all learners."

Examples

Advanced Embedded Training System

An example of a team tutoring in practice is the Advanced Embedded Training System (AETS). AETS is an ITS focused on improving training and reducing manpower usage within shipboard simulation training (Zachary et al., 1999). While effective, it does not replace the role of a human instructor, but rather makes the workload lighter for an instructor through the use of multiple feedback components. The AETS focuses primarily on the Air Defense Team, which is considered one of the most important groups within a ship's Combat Information Center: that team focuses on representations of airborne objects around the ship and differentiates threats from friendly entities.

AETS touches on several points contained within the taxonomies. The use of an experienced team with a distributed skillset is identified in the team taxonomy. There is a shared responsibility as each member plays their role on the team, which is co-located in a real-world environment setting. The apprenticeship pedagogical style is used, since the team is monitored by a human instructor who uses a defined problem-solving method. Due to this intensity of the work task, real-time feedback was a detriment to task attention and AAR was chosen.

Jigsaw II

While not an ITS, Jigsaw II is a team learning technique that heightens a student's sense of responsibility for learning by making each team member an expert for a particular portion of an instructional unit (Slavin, 1978). In using that sense of responsibility, the concept of team maintenance is applied to this activity. Each student is responsible for mastering a segment, and then teaching that segment to the rest of their team.

The activity begins when each student is assigned a chapter to read and given what is called an "expert sheet" with necessary benchmarks to meet. This can be seen as role definition within our taxonomy of teams. Once the material is reviewed, member from various teams who studied the same portion meet to compare notes on their topics for a period of 30 minutes. After this session, each expert returns to their respective teams and then teaches their subject while learning their compatriots' designated subjects. Finally, the team takes a quiz on all of the subjects. This is a form of evaluation, with the team score as a form of AAR. A key feature of this task is reciprocal interdependence (see task taxonomy) as each member depends on his or her teammates to provide the information needed to succeed.

Hidden Profile Task

This classic well-studied problem-solving task (Lu, Yuan & McLeod, 2012; Stasser & Titus, 1985) gives a small group the role of a hiring committee and distributes information about candidates to each member. However, the information varies by team member and is designed to mislead each team member about the best candidate. Only by pooling their information can the group make the best decision as a team. The leadership structure (see team taxonomy) may vary, being leaderless in some groups or with a specifically defined group leader in others. Task actions are mainly discussion, though they are based on a number of written documents. It can be easily done with co-located or distributed teams. An effective team is able to share all needed information with each other, and this is typically difficult (see communication in team taxonomy).

Small Group Survival Scenario

This classic small group problem-solving task, typically based on variations of the NASA Moon Survival scenario (Hall, Mouton & Blake, 1963), the Winter Survival Scenario (Johnson & Johnson, 1975), assigns the challenge of planning for survival by prioritizing the importance of 20–30 items. Task actions are mainly discussion, and therefore can be easily done with co-located or distributed teams. An effective team is able to elicit ideas from all members and come to consensus on a plan. To measure individual performance, the individual's contribution is assessed. This can be done by recording the number of ideas initiated by an individual as well as affirmation of teammates. In this task, there is a correct solution, though the process is also key to the learning. For the session to be an effective learning experience, participants must reflect on their behaviors, so a Socratic pedagogy, accompanied by video or audio replays of what happened, is effective.

Conclusion

These team, task, and team tutor taxonomies can be used to identify the variables that must be considered when designing ITSs for teams. Similarly, they can also be used to guide functional requirements for software like GIFT, so that it may support a large variety of team tutoring experiences. Identification of the key elements in a team tutor can be used to inform the GIFT development roadmap. The biggest impact is that GIFT should be able to consult multiple learner models when choosing its next pedagogical action. For users of GIFT, this addition will translate to an increase in the expressiveness of the tutor. In addition, our taxonomies may be used individually to compare other tutors by placing their characteristics within the taxonomies. Future work will include the expansion of these taxonomies and validation of them through the development of actual team tutors.

Acknowledgements

The research described herein has been sponsored by the U.S. Army Research Laboratory - Human Research & Engineering Directorate (ARL-HRED). Statements and opinions expressed in this paper do not necessarily reflect the position or the policy of the U.S. Government, and no official endorsement should be inferred.

References

Andrews, F. M. & Farris, G. F. (1972). Time pressure and performance of scientists and engineers: A five-year panel study. *Organizational Behavior and Human Performance*, *8*(2), 185-200.

DeShon, R. P., Kozlowski, S. W., Schmidt, A. M., Milner, K. R. & Wiechmann, D. (2004). A multiple-goal, multilevel model of feedback effects on the regulation of individual and team performance. *Journal of Applied Psychology*, *89*(6), 1035.

Faria, L., Silva, A., Vale, Z. & Marques, A. (2009). Training Control Centers" Operators in Incident Diagnosis and Power Restoration Using Intelligent Tutoring Systems. *Learning Technologies, IEEE Transactions on*, *2*(2), 135-147.

Fowlkes, J., Dwyer, D. J., Oser, R. L. & Salas, E. (1998). Event-based approach to training (EBAT). *The International Journal of Aviation Psychology*, *8*(3), 209-221.

Hall, E. J., Mouton, J. S. & Blake, R. R. (1963). Group problem solving effectiveness under conditions of pooling vs. interaction. *The Journal of Social Psychology*, *59*(1), 147-157.

Johnson, D. W. & Johnson, F. P. (1975). *Joining together: Group theory and group skills*. Englewood Cliffs, NJ: Prentice-Hall, Inc.

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.

Lu, L., Yuan, Y. C. & McLeod, P. L. (2012). Twenty-Five Years of Hidden Profiles in Group Decision Making A Meta-Analysis. *Personality and Social Psychology Review*, *16*(1), 54-75.

Ordonez, L. & Benson, L. (1997). Decisions under Time Pressure : How Time Constraint Affects Risky Decision Making. *Organizational Behavior and Human Decision Processes, 71*(2), 121–140.

Saavedra, R., Earley, P. C. & Van Dyne, L. (1993). Complex interdependence in task-performing groups. *Journal of Applied Psychology*, 78(1), 61-72.

Salas, E., Burke, C. S., Fowlkes, J. E. & Priest, H. A. (2004). On measuring teamwork skills. *Comprehensive handbook of psychological assessment*, *4*, 427-442.

Salas, E.,Kosarzycki, M.P.,Tannenbaum, S . I. & Carnegie, D. (2004). Principles and advice for understanding and promoting effective teamwork in organizations. In R. Burke & C. Cooper (E ds.), *Leading in turbulent times: Managing in the new world of work* (pp. 95 -120). Malden, MA: Blackwell.

Salas, E., Shuffler, M.L., Thayer, A. L., Bedwell, W. L. & Lazzara, E. H. (in press). Understanding and improving teamwork in organizations: A scientifically based practical guide. Accepted for publication at *Human Resource Management.*

Sims, D. E. & Salas, E. (2007). When teams fail in organizations: what creates teamwork breakdowns?. In J. Langan-Fox, C. L. Cooper & R. J. Klimoski (Eds.), *Research companion to the dysfunctional workplace: Management challenges and symptoms*. Edward Elgar Publishing.

Slavin, R. E. (1978). Using Student Team Learning. The Johns Hopkins Team Learning Project.

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. & Hold en, H. K. (Eds.). (2013). *Design Recommendations for Intelligent Tutoring Systems - Volume 1 - Learner Modeling* (1ed.). Orlando, Florida: U.S. Army Research Laboratory.

Sottilare, R. A., Holden, H. K., Brawner, K. W. & Goldberg, B. S. (2011, January). Challenges and Emerging Concepts in the Development of Adaptive, Computer-based Tutoring Systems for Team Training. In *The Interservice/Industry Training, Simulation & Education Conference (I/ITSEC)*(Vol. 2011, No. 1). National Training Systems Association.

Stasser, G. & Titus, W. (1985). Pooling of unshared information in group decision making: Biased information sampling during discussion. *Journal of personality and social psychology, 48*(6)1467.

Stewart, S. M., Lam, T. H., Betson, C. L., Wong, C. M. & Wong, A. M. P. (1999). A prospective analysis of stress and academic performance in the first two years of medical school. *Medical Education, 33*, 243–250.

Walton, J., Dorneich, M., Gilbert, S., Bonner, D., Winer, E. & Ray, C. (2014). Modality and Timing of Feedback: Implications for GIFT. In *Proceedings of the 2nd Annual Generalized Intelligent Framework for Tutoring (GIFT) Users Symposium*. Army Research Laboratory.

Wildman, J. L., Thayer, A. L., Rosen, M. A., Salas, E., Mathieu, J. E. & Rayne, S. R. (2012). Task Types and Team-Level Attributes Synthesis of Team Classification Literature. *Human Resource Development Review, 11*(1), 97-129.

Zachary, W., Cannon-Bowers, J. A., Bilazarian, P., Krecker, D. K., Lardieri, P. J. & Burns, J. (1998). The Advanced Embedded Training System (AETS): An intelligent embedded tutoring system for tactical team training. *International Journal of Artificial Intelligence in Education (IJAIED)*, *10*, 257-277.

Modality and Timing of Team Feedback: Implications for GIFT

Jamiahus Walton, Michael C. Dorneich, Stephen Gilbert, Desmond Bonner, Eliot Winer, Colin Ray
Iowa State University

Introduction

This paper discusses considerations relevant to the design of team feedback in intelligent tutoring systems (ITSs). While team tutoring is a goal for the Generalized Intelligent Framework for Tutoring (GIFT), further research must be done to explore the focus, modalities, and timing of feedback for teams. Although there have been a number of studies on feedback, there are a limited number of studies on feedback for teams. This theoretical paper leverages previous research on ITSs, training, individual feedback, and teamwork models to inform appropriate decisions about the most effective feedback mechanisms for teams. Finally, the implications of team feedback on the design of GIFT are discussed.

Teams have the ability to achieve goals that are unobtainable by individuals alone. It is important to implement effective training for teams to support performance effectiveness. An important element of training is feedback. Feedback has the function of guiding or motivating individuals based on their past performance. The purpose of guiding feedback is to direct an individual to a desired behavior. The purpose of motivational feedback is to motivate the individual by mentioning future rewards (Ilgen, Fisher & Taylor, 1979). Although there have been a number of studies on feedback, there are a limited number of studies on feedback for teams. A common theme among these studies is determining whether feedback should be given at an individual or team level (Tindale, 1989). Some studies for teams suggest that team performance is influenced by feedback on an individual level (Berkowitz & Levy, 1956) and some studies suggest that groups outperform individuals when feedback is given to the entire team after each decision is made (Tindale, 1989). The purpose of the current paper is to characterize the range of modalities of feedback, timing of feedback, focus level of feedback, and who should receive feedback (i.e., individual vs. feedback) for teams to assist in the design of feedback for ITSs for teams. Finally, the implications of team feedback on the design of GIFT is discussed.

Related Work and Implications for Team Feedback

There are several aspects that are relevant to a discussion of team feedback. A definition and description of feedback is given in the next subsection. Feedback itself has several considerations, including 1) the type, or focus, of the feedback given, 2) the timing of feedback, and 3) to whom feedback is given. Previous work related to these aspects are discussed in the following subsections, as well as some initial discussion of the implications towards effective feedback for teams.

Taxonomy of teams

Feedback is an important aspect of team tutors. In addition to feedback there are other aspects to consider in team tutors: Teams, tasks, and tutoring approach. A companion paper in this symposium (Bonner et al.,

2014) presents a taxonomy of team tutoring, of which feedback is one element, and serves as the basis for design decisions on the interaction between team variables and tutor decisions for team feedback.

Feedback

There are a number of studies that have been conducted on feedback. DeShon, Kozlowski, Schmidt, Milner & Wiechmann (2004) present a model of how feedback on goals and performance influences learning and performance. Dominick et al. (1997) studied the influence of behavioral-based peer feedback on team behavior that occurs in a team-based task. Ilgen et al. (1979) provided a process-orientated review of how feedback influences behavior of individuals in organizations. Kluger and Denisi (1996) sought to show that the assumption that feedback intervention consistently improves performance is a result of the disregarded fact that the influence of feedback on performance varies.

Ilgen et al. (1979) describes feedback as a unique case of a communication process where a source conveys a message to a receiver. Ilgen et al. (1979) defines three different sources of feedback: 1) individuals who are in a position to evaluate a recipient’s behavior, 2) task environment, and 3) self-evaluation from recipients. The information in feedback that an individual receives from any source deals with his or her past performance.

Focus level of feedback

Hattie and Timperley (2007) make the claim that there are four levels of focus for feedback: 1) feedback can focus on the task at hand and whether or not it is correct, 2) feedback can focus on the process used to complete a task, 3) feedback can focus on the user’s ability to self-regulate (e.g., self-evaluate), and 4) feedback can focus on the “self”. They argue that the levels of feedback that focus on how a user processes a task and the level that focuses on the user’s self-regulation are both powerful with respects to the deep processing and mastery of tasks. This suggests that if an ITS should give feedback that focuses on the user’s process of task completion and the user’s self-regulation ability.

Individual vs. team feedback

Feedback for a team is different from feedback for an individual because of the information and the ability of a team member to act on the information. The three types of feedback are individual feedback, individual feedback in groups, and group feedback (Nadler, 1979). A study conducted by Zander and Wolfe (1964) concluded that the teams that received both individual feedback and group feedback had the greatest individual performance increase. This conclusion suggests that a tutor needs to provide individual and group feedback in order to effectively train a team. This method of mixed individual and group feedback has been implemented in existing training systems such as the Advanced Embedded Training System (AETS). Zachary et al. (1999) found the traditional feedback approach with ITSs difficult to apply to teams. The traditional feedback approach is difficult to apply because all the members of a team are collaboratively and simultaneously working together during the simulation. As a result, it was not possible for one team member to stop the simulation without hindering the mental flow the whole problem-solving process for the other members of the team.

Timing of feedback

The timing of feedback is important to consider. Feedback during task execution may interrupt task performance, interrupt the cognitive process of task execution, and prevent users from learning how to identify their own error (Corbett & Anderson, 2001). The timing of feedback given to a user can be influenced by the user's affect state. Common affect states that occur within students are confusion, frustration, engagement, and boredom (Calvo & D'Mello, 2012). For example, a user who is being trained on a topic, especially if the topic is novel, will most be confused as a result of irregularities, inconsistencies, and qualms in subject matter. (Calvo & D'Mello, 2012). Confusion, however, should not be avoided by an ITS, rather an ITS should adapt to the user's uncertainty. Studies have shown that the users overall learning is improved when an ITS is able to recognize and adapt to confusion or uncertainty (Forbes-Riley & Litman, 2011).

The timing of feedback for teams and individuals may differ. Some research suggest that delayed feedback for individuals is more beneficial to students' retention (Butler, Karpicke & Roediger, 2007) and learning (Walsh, Ling, Wang & Carnahan, 2009) when compared to immediate feedback. This finding suggests that an ITS should give feedback at the end of a given task (e.g., After Action Review). Delayed feedback would be most beneficial for tasks that are complex and fast-paced because it would allow a team to finish a task without interrupting the team member's flow. Once the team has completed the task, the ITS can identify problems and offer suggestions on how to address any problems that are identified. However, other research suggests that delayed feedback would decrease the group motivation as compared to receiving immediate feedback (Gabelica, Bossche, Segers & Gijselaers, 2012). This suggests that individual feedback should be given immediately, or close to real time, to be most beneficial. This method may not be ideal for complex tasks but there are task that are more easily interrupted and could allow for real-time feedback without interrupting the flow of the task. Corbett and Anderson (Corbett & Anderson, 2001) identified two principles of immediate feedback from the results of their research. The first principle was that giving feedback on each problem-solving step is an effective form of tutor support for students attempting to understand a complex problem solving skill. The second principle was that although lessening of immediate feedback on tasks (e.g., coding) may be essential, it will not promote error detection and other process monitoring skills of individuals.

Teamwork

Teams are becoming more important today as the complexity of tasks increase. Research on teamwork is sometimes difficult to generalize because different teams function differently depending on the task and the domain. Salas, Sims, and Burke (2005) proposed a model that is supported by empirical evidence and practically relevant. The authors described five important components of teamwork that they call the "big five." They argue that the big five are required to complete any team task. The big five include team leadership, mutual performance monitoring, backup behavior, adaptability, and team orientation.

Team Leadership

Though some research concludes that team leadership is not important in most situations (Fransen, Weinberger & Kirschner, 2013), others contend that leadership of a team is an important contribution to the effectiveness of a team (Zaccaro, Rittman & Marks, 2001). There are certain functions that a team leader must be capable of in order for a team to be successful. The team leadership needs to facilitate team problem solving through mental processes (e.g., shared mental models), coordinating the team, and keep the team motivated (Salas et al., 2005). The team leadership should receive feedback that shows how well they are facilitating the team. If the team leadership is in training and is learning how to properly facilitate a team then that would suggest real-time feedback should be used. Giving real-time feedback on each problem-solving step is an effective form of tutor support for users attempting to understand a complex problem solving skill (Corbett & Anderson, 2001). The feedback given to the leadership team and the timing of that feedback would change if the focus of the task at hand changes (e.g., if the team already had the knowledge to complete the task and was focus on efficiency).

Mutual performance monitoring

Mutual performance monitoring is the ability of each team member to keep track of other members' work while continuing to carry out tasks, make sure that everything is functioning as expected, and make sure the other members are following procedures (Mclntyre & Salas, 1995). This component of the big five is important for a team throughout a team task but it becomes especially important when the task has a high stress level. However, it is difficult to give feedback on mutual performance monitoring because it is difficult to measure, due to a lack of an accepted method of detecting when is occurring (Salas et al., 2005). Feedback pertaining to the mutual performance monitoring of a team would most likely be focused on the team's process of the given task. If a team processes a task correctly then they will be able to determine when members need additional help. If members of the team do not give support to members then feedback should be given on how the team is processing the task at hand. If the purpose of the training is to teach members to recognize when another member needs additional help then feedback should be delayed. It is difficult to determine if members of a team exhibit successful mutual performance monitoring if no problem ever arises during a task.

Backup behavior

Backup behavior is defined as providing resources and task-related efforts to another member when it is recognized by possible backup providers that there is a problem with the distribution of workload within the team (Porter et al., 2003). There are different ways that members of a team can provide backup behavior. For example, members of a team can provide verbal feedback and coaching to help improve performance (Marks, Mathieu & Zaccaro, 2001). Members of a team can go beyond providing feedback and coaching to other members by assisting a teammate in performing a task (Marks et al., 2001; Salas et al., 2005). Doing this will allow other members to observe a task conducted correctly and allow members to correctly complete the task themselves. Lastly, if assisting a team member is not enough to help improve team performance then members of a team can complete tasks for other team members when an overload is detected (Marks et al., 2001; Salas et al., 2005). Feedback pertaining to backup behavior depends on the goal of the task at hand. If the goal of task is teach members to identify and take action when other members need help, then feedback should be given in real time. However, if the task at hand is to give the team a chance to practice their skills, then feedback should be delayed until the end of the end of the task.

Adaptability

Teams need to be able to adapt to tasks that are continuously changing. A team is required to have the ability to utilize knowledge, skills, and attitudes that allow members to recognize deviations from anticipated actions and readjust actions accordingly to acquire the adaptability component of the big five (Priest, Burke, Munim & Salas, 2002). There are many different ways that adaptability can appear, depending on the task and the challenge the team face (Salas et al., 2005). For example, let's assume a football team (specifically the offensive line) was told to run an offensive play. A coach may first allow the team to run the play as if everything ran perfectly. Once a team has mastered that play, a coach may have the defensive line set up to defend against the offensive line. The coach may have the defense set up in a way where another play (modified from the play they plan to run) may be more successful. The Quarterback (QB) can change the play by calling an audible and changing the play. Feedback can be given on how well the QB handles the situation. The QB can be evaluated on whether or not an appropriate play was called or if an audible was called at all. Similarly, an ITS should be able to compare the actions of a team during a task to the expected action in order to understand adaptability.

Team orientation

Team orientation does not focus on that behavioral aspect of teams but rather the attitudinal. It has been found to improve satisfaction of individuals, individual effort, and performance (Salas et al., 2005); facilitate overall performance (Driskell & Salas, 1992; Eby & Dobbins, 1997), and influence team cooperation behaviors (Eby & Dobbins, 1997). Feedback pertaining to team orientation should be given no matter the focus of feedback. If the team orientation is not sufficient (i.e., if the members of negative attitudes toward the team) then the teams efficiency will suffer. Feedback that pertains to a teams' orientation, also called collectivistic orientation (Eby & Dobbins, 1997), can be gathered from members of a team using a desired data collection method (e.g., Wagner and Moch's (1986) individualism-collectivism measure).

Modality

An issue to address when building an ITS is the modality of feedback. For instance, should the feedback given to a team be text, visual, verbal, or tactical? The domain of athletic sports teams can provide useful insight to team feedback because players are continuously receiving feedback from their coaches and teammates in order to improve their skill levels. Literature suggests that the combination of visual and vocal feedback is beneficial to performance. A study conducted on high school football athletes by Stokes et al. (2010) concluded that vocal and visual (and sometimes acoustical) performance feedback improved the players' pass-blocking skills. Another study conducted by Smith and Ward (2006) showed that performance was the best during the goal-setting condition where verbal feedback was given during practice. Smith and Ward also discovered, via a questionnaire, that players did not prefer individual goal setting intervention because it was missing the visual feedback. These conclusions suggest that an intelligent tutor should, at minimum, give visual and vocal forms of feedback.

Team Feedback and Implications for GIFT

There are several factors that are important to team feedback. The first factor is the how feedback is given to teams. In order to provide effective feedback in team training, there needs to be both individual and group feedback. Although both group and individual feedback should be administered to a group, Smith (1972) concluded that individual reinforcement feedback produced more satisfaction with the task than group reinforcement feedback. A feature that allows GIFT to give feedback to the individual members of teams and to the team as a whole should be added to the roadmap for GIFT. The second factor that is important to team feedback is the timing of the feedback. Literature suggests that individual and group feedback should be given at the end of a task. Real-time feedback is more difficult to apply to a team task, although the benefits can also be immediate. Often the timing of feedback is governed by the nature of the focus of the feedback. A feature that allows users to set parameters to tell GIFT when to give feedback to a team should be added to the roadmap for GIFT. The third important factor is feedback that describes how effectively a team completes a task. Salas et al. (2005) suggest that teamwork has five components that influence the effectiveness of task completion. These five elements are team leadership, mutual performance monitoring, backup behavior, adaptability, and team orientation. Feedback about the different elements of the team should be given when feedback is given to the team. A feature that allows GIFT to determine if a team is effectively completing a task should be added to the roadmap for GIFT. Ultimately GIFT should be able to use different models for effective teams but the big five presented by Salas et al. (2005) could be used as a starting point. The fourth important factor of feedback is the modality of feedback. Research suggests that the mode of feedback that benefits performance the most is visual and vocal (Smith & Ward, 2006; Stokes et al., 2010).

There are several implications for GIFT functional requirements in order achieve effective team training: 1) GIFT should to be able to differentiate different members of the team, 2) GIFT should collect data to support different metrics to evaluate team performance, 3) GIFT should have the ability to collect and evaluate data in real-time, and 4) GIFT should be able to understand the goal of the training and the current state of the team as a whole.

GIFT needs to be able to track different members of the team in order to provide individualized feedback. Team members may have the same role but tracking of team members increases in complexity if team members have different roles. If the members have different roles then the members are more likely to need feedback that is unique to their assigned role. GIFT would need to be able to track the members, differentiate each member, understand that member’s role within the team, and then understand how that member’s role relates to the task at hand. Tracking the different members will look different depending on the task. For example, if the members are in the same room then GIFT will need to track the member’s different locations and know where they are in relation to the task space. If GIFT cannot track team members then the feedback that GIFT gives will be limited. Furthermore, tracking the team members will allow GIFT to evaluate how the members interact with one another.

GIFT also needs to have the ability to collect data to support different metrics to evaluate team performance. It is difficult to tell which metrics are important for determining team performance because little research has been conducted in that area. Further research is needed to understand the different metrics that are domain independent indicators of team performance. Once these metrics are identified then research will need to be conducted to better understand how those metrics connect to the different elements (i.e., the big five) of teams.

Once these metrics are explored then the understanding of those metrics will need to be translated so that GIFT can accurately interpret the data input. For example, the way that members respond when a member experiences a very high workload during a team task can indicate how well a team shows backup behavior. If a tutor notices that a lot of time passes before any assistance is given to the team member with a high workload then that may indicate that the team, as a whole, lacks backup behavior. On the other hand, if little time passes before the team member with the high workload receives assistance from the other team members then that team may be able to effectively show backup assistance. This example may also indicate something about mutual performance monitoring as well since backup behavior and mutual performance monitoring are closely related. The resulting backup behavior that a team exhibits may be a result of good mutual performance monitoring.

Research indicates that it would benefit a team member's performance to give feedback to a team after the task (e.g., After Action Review) but ultimately GIFT should have the ability to collect and evaluate data in real-time. This ability may not be used to give feedback to the student participating in the task but it may be used to give feedback to the instructors that are overseeing the task. For example, imagine a team conducting a task in a virtual world. During that task in the virtual world there is a scenario where GIFT notices that the team is currently lacking in team leadership. If GIFT were to notify the instructor about the team's lacking leadership, then that instructor can throw in an unexpected task to the virtual world that exploits the team's lack of team leadership. However, further research needs to be done in order to evaluate if the previously mentioned scenario is achievable and the impact it would have on student and team learning.

In order for GIFT to have the ability to give influential feedback it needs to be able to understand the goal of the training and the current state of the team as a whole (Ilgen et al., 1979). If GIFT is unable to understand the current state of the team as a whole then it will not be able to determine what the team needs to do in order to reach the goal. Further research is needed to better understand the different independent elements of team that will allow the evaluation of a team. Once these elements are identified then those elements need to be translated into empirical data that can be automatically tracked by GIFT. The better GIFT understands the current state of a team the better feedback it will be able to give teams.

Conclusion

While feedback to teams in tutoring contexts has many implications for GIFT, there are still many areas that warrant further research. For example, research needs to explore the different empirical data that will allow us to evaluate a team's effectiveness relative to a given task. Further research is also needed to explore the different types of feedback (e.g., immediate vs. delayed) and provide evidence of how the feedback influences the team and its members. For GIFT, the next step is to develop different modules that allow GIFT to handle giving and receiving information from multiple team members simultaneously. GIFT modules to support team evaluations need to be developed. As the modules are incrementally and iteratively developed they should be tested and evaluated to better understand the accuracy of the module. The ultimate goal is to develop GIFT's ability to support evaluation of a team in real time and to provide effective feedback to positively influence team learning.

Acknowledgement

The research described herein has been sponsored by the U.S. Army Research Laboratory - Human Research & Engineering Directorate (ARL-HRED). Statements and opinions expressed in this paper do not necessarily reflect the position or the policy of the U.S. Government, and no official endorsement should be inferred.

References

Berkowitz, L. & Levy, B. I. (1956). Pride in group performance and group-task motivation. *Journal of Abnormal Psychology*, *53*(3), 300–6.

Bonner, D., Gilbert, S., Dorneich, M., Burke, S., Walton, J., Ray, C. & Winer, E. (2014). Taxonomy of Teams, Team Tasks, and Tutors. In *Proceedings of the 2nd Annual Generalized Intelligent Framework for Tutoring (GIFT) Users Symposium*. Army Research Laboratory.

Butler, A. C., Karpicke, J. D. & Roediger, H. L. (2007). The effect of type and timing of feedback on learning from multiple-choice tests. *Journal of Experimental Psychology. Applied*, *13*(4), 273–81. doi:10.1037/1076-898X.13.4.273

Calvo, R. A. & D'Mello, S. (2012). Frontiers of Affect-Aware Learning Technologies. *Intelligent Systems, IEEE, 27*(6), 86 – 89.

Corbett, A. T. & Anderson, J. R. (2001). Locus of Feedback Control in Computer-Based Tutoring: Impact on Learning Rate , Achievement and Attitudes. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 245–252).

DeShon, R. P., Kozlowski, S. W. J., Schmidt, A. M., Milner, K. R. & Wiechmann, D. (2004). A Multiple-Goal, Multilevel Model of Feedback Effects on the Regulation of Individual and Team Performance. *The Journal of Applied Psychology*, *89*(6), 1035–56. doi:10.1037/0021-9010.89.6.1035

Dominick, P. G., Reilly, R. R. & Mcgourty, J. W. (1997). The Effects of Peer Feedback on Team Member Behavior. *Group & Organization Management*, *22*(4), 508–520. doi:10.1177/1059601197224006

Driskell, J. E. & Salas, E. (1992). Collective Behavior and Team Performance. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, *34*(3), 277–288. doi:10.1177/001872089203400303

Eby, L. T. & Dobbins, G. H. (1997). Collectivistic orientation in teams : an individual and group-level analysis. *Journal of Organizational Behavior*, *18*, 275–295.

Forbes-Riley, K. & Litman, D. (2011). Benefits and challenges of real-time uncertainty detection and adaptation in a spoken dialogue computer tutor. *Speech Communication*, *53*(9-10), 1115–1136. doi:10.1016/j.specom.2011.02.006

Fransen, J., Weinberger, A. & Kirschner, P. a. (2013). Team Effectiveness and Team Development in CSCL. *Educational Psychologist*, *48*(1), 9–24. doi:10.1080/00461520.2012.747947

Gabelica, C., Bossche, P. Van Den, Segers, M. & Gijselaers, W. (2012). Feedback, a powerful lever in teams: A review. *Educational Research Review*, *7*(2), 123–144. doi:10.1016/j.edurev.2011.11.003

Hattie, J. & Timperley, H. (2007). The Power of Feedback. *Review of Educational Research*, *77*(1), 81–112. doi:10.3102/003465430298487

Ilgen, D. R., Fisher, C. D. & Taylor, M. S. (1979). Consequences of Individual Feedback on Behavior in Organizations. *Journal of Applied Psychology*, *64*(4), 349–371. doi:10.1037//0021-9010.64.4.349

Kluger, A. N. & Denisi, A. (1996). The Effects of Feedback Interventions on Performance : A Historical Review, a Meta-Analysis , and a Preliminary Feedback Intervention Theory. *American Psychological Association, Inc*, *119*(2), 254–284.

Marks, M. A., Mathieu, J. E. & Zaccaro, S. J. (2001). A Temporally Based Framework and Taxonomy of Team Processes. *Academy of Management Review*, *26*(3), 356–376.

Mclntyre, R. M. & Salas, E. (1995). Measuring and managing for team performance: Emerging principles from complex environments. In R. A. Guzzo & E. Salas (Eds.), *Team effectiveness and decision making in organizations* (pp. 9–45). San Francisco: Jossey-Bass.

Nadler, D. A. (1979). The effects of feedback on task group behavior: A review of the experimental research. *Organizational Behavior and Human Performance*, *23*(3), 309–338. doi:10.1016/0030-5073(79)90001-1

Porter, C. O. L. H., Hollenbeck, J. R., Ilgen, D. R., Ellis, A. P. J., West, B. J. & Moon, H. (2003). Backing up behaviors in teams: the role of personality and legitimacy of need. *The Journal of Applied Psychology*, *88*(3), 391–403. Retrieved from http://www.ncbi.nlm.nih.gov/pubmed/12814289

Priest, H. a., Burke, C. S., Munim, D. & Salas, E. (2002). Understanding Team Adaptability: Initial Theoretical and Practical Considerations. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, *46*(3), 561–565. doi:10.1177/154193120204600372

Salas, E., Sims, D. E. & Burke, C. S. (2005). Is there a "Big Five" in Teamwork? *Small Group Research*, *36*(5), 555–599. doi:10.1177/1046496405277134

Smith, K. H. (1972). Changes in group structure through individual and group feedback. *Journal of Personality and Social Psychology*, *24*(3), 425–428. doi:10.1037/h0033729

Smith, S. L. & Ward, P. (2006). Behavioral Interventions to Improve Performance in Collegiate Football. *Journal of Applied Behavior Analysis*, *39*(3), 385–391. doi:10.1901/jaba.2006.5-06

Stokes, J. V, Luiselli, J. K., Reed, D. D. & Fleming, R. K. (2010). Behavioral coaching to improve offensive line pass-blocking skills of high school football athletes. *Journal of Applied Behavior Analysis*, *43*(3), 463–72. doi:10.1901/jaba.2010.43-463

Tindale, R. S. (1989). Group vs Individual Information Processing : The Effects of Outcome Feedback on Decision Making. *Organizational Behavior and Human Decision Processes*, *44*, 454–473.

Wagner, J. A. & M. K. Moch. (1986). Individualism-collectivism: Concept and measure. *Group & Organization Management*, *11*(3), 280 – 304.

Walsh, C. M., Ling, S. C., Wang, C. S. & Carnahan, H. (2009). Concurrent versus terminal feedback: it may be better to wait. *Academic Medicine : Journal of the Association of American Medical Colleges*, *84*(10), S54–7. doi:10.1097/ACM.0b013e3181b38daf

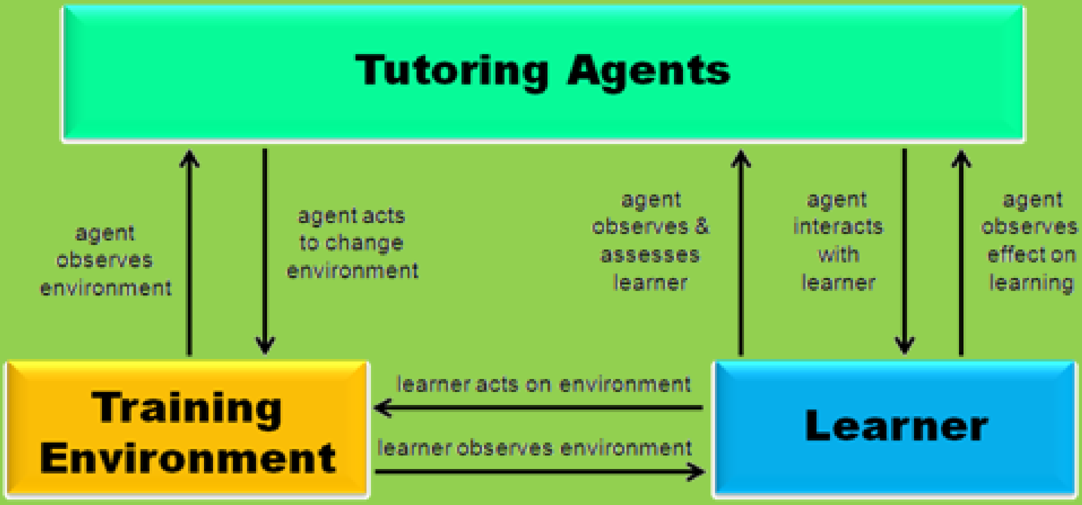
Zaccaro, S. J., Rittman, A. L. & Marks, M. a. (2001). Team Leadership. *The Leadership Quarterly*, *12*(4), 451–483. doi:10.1016/S1048-9843(01)00093-5

Zachary, W., Cananon-Bowers, J., Bilazarian, P., Krecker, D., Lardieri, P. & Burns, J. (1999). The Advanced Embedded Training System (AETS): An Intelligent Embedded Tutoring System for Tactical Team Training. *International Journal of Artificial Intelligence in Education*, *10*, 257–277.

Zander, A. & Wolfe, D. (1964). Administrative Rewards and Coordination Among Committee Members. *Administrative Science Quarterly*, *9*(1), 50–69.

Proceedings of the Second Annual GIFT Users Symposium

GIFT, the Generalized Intelligent Framework for Tutoring, is a modular, service-oriented architecture developed to lower the skills and time needed to author effective adaptive training. Design goals for GIFT also include capturing best instructional practices, promoting standardization and reuse for adaptive instructional content and methods and evaluation of the effectiveness of tutoring technologies. Truly adaptive systems make intelligent (optimal) decisions about tailoring instruction in real-time and make these decisions based on information about the learner and the instructional context (training environment.


Tutoring Agents
agent observes environment
agent acts to change environment
agent observes & assesses learner
agent interacts with learner
agent observes effect on learning
Training Environment
learner acts on environment
learner observes environment
Learner

The GIFT Users Symposia were started in 2013 to capture successful implementations of GIFT from the user community and to share recommendations leading to more useful capabilities for GIFT authors, researchers, and learners.

***About the Editor: Dr. Robert Sottilare** leads adaptive training research within US Army Research Laboratory's Learning in Intelligent Tutoring Environments (LITE) Lab Orlando Florida. He is a co-creator of the Generalized Intelligent Framework for Tutoring (GIFT).*

Part of the Adaptive Tutoring Series

9 7 8 - 0 - 9 8 9 3 9 2 3 - 4 - 1