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)

Workshop Co-Chairs:

Robert A. Sottilare & Heather K. Holden
Army Research Laboratory, Human Research and Engineering Directorate, Orlando, Florida, USA

https://gifttutoring.org/news/14

Preface

This workshop provides the AIED community with an in-depth exploration of the Army Research Laboratory's effort to develop tools, methods and standards for Intelligent Tutoring Systems (ITS) as part of their Generalized Intelligent Framework for Tutoring (GIFT) research project. GIFT is a modular, service-oriented architecture developed to address authoring, instructional strategies, and analysis constraints currently limiting the use and reuse of ITS today. Such constraints include high development costs; lack of standards; and inadequate adaptability to support tailored needs of the learner. GIFT's three primary objectives are to provide: (1) authoring tools for developing new ITS, ITS components (e.g., learner models, pedagogical models, user interfaces, sensor interfaces), tools, and methods based on authoring standards that support reuse and leverage external training environments; (2) an instructional manager that encompasses best tutoring principles, strategies, and tactics for use in ITS; and (3) an experimental testbed for analyzing the effect of ITS components, tools, and methods. GIFT is based on a learner-centric approach with the goal of improving linkages in the adaptive tutoring learning effect chain in Figure 1.

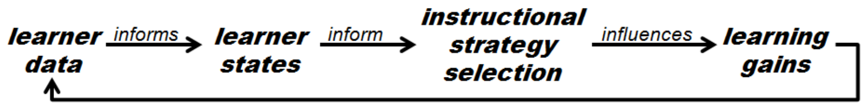


Figure 1: Adaptive Tutoring Learning Effect Chain

The goal of GIFT is to make ITS affordable, effective, usable by the masses, and provide equivalent (or better) instruction than expert human tutors in one-to-one and one-to-many educational and training domains. GIFT's modular design and standard messaging provides a largely domain-independent approach to tutoring where domain-dependent information is concentrated in the one module making most of its components, tools and methods reusable across training domains. More information about GIFT can be found at **www.GIFTtutoring.org**.

The workshop is divided into five themes: **(1)** *Fundamentals of GIFT* (includes a tutorial on GIFT and a detailed demonstration of the latest release); **(2)** *Authoring ITS using the GIFT Authoring Construct*; **(3)** *Adapting Instructional Strategies and Tactics using GIFT*; **(4)** *Analyzing Effect using GIFT*; and **(5)** *Learner Modeling*. Themes include presentations from GIFT users regarding their experiences within the respective areas and their recommendations of design enhancements for future GIFT releases. Theme 5 is dedicated to discussing the outcomes of the learner modeling advisory board meeting conducted at the University of Memphis Meeting in September 2012.

July, 2013
Robert Sottilare, Heather Holden.

Program Committee

Co-Chair: Robert Sottilare, Army Research Laboratory, Orlando, FL, USA
(robert.a.sottilare@us.army.mil)

Co-Chair: Heather Holden, Army Research Laboratory, Orlando, FL, USA
(heather.k.holden@us.army.mil)

Arthur Graesser, *University of Memphis*

Xiangen Hu, *University of Memphis*

James Lester, *North Carolina State University*

Ryan Baker, *Columbia University*

Table of Contents

Motivations for a Generalized Intelligent Framework for Tutoring (GIFT) for Authoring, Instruction and Analysis

Robert A. Sottilare, Ph.D. and Heather K. Holden, Ph.D.

U.S. Army Research Laboratory – Human Research and Engineering Directorate
{robert.sottilare, heather.k.holden}@us.army.mil

Abstract. Intelligent Tutoring Systems (ITS) have been shown to be effective tools for one-to-one tutoring in a variety of well-defined domains (e.g., mathematics, physics) and offer distinct advantages over traditional classroom teaching/training. In examining the barriers to the widespread use of ITS, the time and cost for designing and author-ing ITS have been widely cited as the primary obstacles. Contributing factors to time and cost include a lack of standards and minimal opportunities for reuse. This paper explores motivations for the development of a Generalized Intelligent Framework for Tutoring (GIFT). GIFT was conceived to meet challenges to: author ITS and ITS components, offer best instructional practices across a variety of training tasks (e.g., cognitive, affective, and psychomotor), and provide a testbed for analyzing the effect of tutoring technologies (tools and methods).

1 Introduction

GIFT [1] is a modular, service-oriented architecture developed to address authoring, instructional strategies, and analysis constraints currently limiting the use and reuse of ITS today. Such constraints include high development costs; lack of standards; and inadequate adaptability to support tailored needs of the learner. GIFT's three primary objectives are to develop: (1) authoring tools to develop new ITS, ITS components (e.g., learner models, pedagogical models, user interfaces, sensor interfaces), tools, and methods, and develop authoring standards to support reuse and leveraging external training environments; (2) provide an instructional manager that encompasses best tutoring principles, strategies, and tactics for use in ITS; and (3) an experimental testbed to analyze the effect of ITS components, tools, and methods. GIFT is based on a learner-centric approach with the goal of improving linkages in the adaptive tutoring learning effect chain in Figure 1.

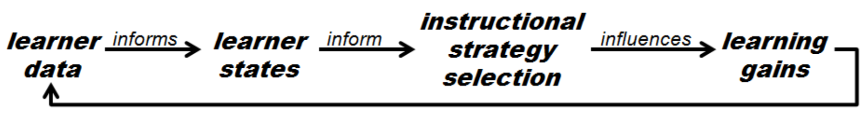


Figure 1: Adaptive Tutoring Learning Effect Chain [2]

GIFT's modular design and standard messaging provides a largely domain-independent approach to tutoring where domain-dependent information is concentrated in the domain module making most of its components, tools and methods reusable across tutoring scenarios.

2 Motivations for authoring tools, standards and best practices

The primary goal of GIFT is to make ITS affordable, usable by the masses, and equivalent (or better) than an expert human tutors in one-to-one and one-to-many educational and training scenarios for both well-defined and ill-defined domains. As ITS seek to become more adaptive to provide tailored tutoring experiences for each learner, the amount of content (e.g., interactive multimedia and feedback) required to support additional adaptive learning paths grows exponentially. More authoring requirements generally means longer development timelines and increased development costs. If ITS are to be ubiquitous, affordable, and holistically learner-centric, it is essential to for ITS designers and developers to develop methods to rapidly author content or reuse existing content. Overcoming barriers to reuse means developing standards. In this context, the idea for GIFT was born.

2.1 GIFT Authoring Goals

Adapted from Murray [3] [4] and Sottilare and Gilbert [5], the authoring goals discussed below identify several motivating factors for the development of authoring methods and standards. First and foremost, the idea of a GIFT is founded on decreasing the effort (time, cost, and/or other resources) required to author and analyze the effect of ITS, ITS components, instructional methods, learner models, and domain content. ITS must become affordable and easy to build so we should strive to decrease the skill threshold by tailoring tools for specific disciplines to author, analyze and employ ITS.

In this context, we should provide tools to aid designers, authors, trainers/teachers, and researchers organize their knowledge for retrieval and application at a later time. Automation should be used to the maximum extent possible to data mine rich repositories of information to create expert models, misconception libraries, and hierarchical path plans for course concepts.

A GIFT should support (structure, recommend, or enforce) good design principles in its pedagogy, its user interface, etc. It should enable rapid prototyping of ITS to allow for rapid design/evaluation cycles of prototype capabilities. To support reuse, a GIFT should employ standards to support rapid integration of external training/tutoring environments (e.g., serious games) to leverage their engaging context and avoid authoring altogether.

2.2 Serious Games and ITS

Serious games, which are computer-based games aimed at training and education rather than pure entertainment, are one option for reuse if they can easily be integrated with tutoring architectures like GIFT. Serious games offer high-level interactive multi-media instructional (IMI) content that is engaging and is capable of supporting a variety of scenarios with the same basic content. While most serious games offer prescriptive feedback based on learner task performance, the integration of serious games with ITS opens up the possibility of more adaptive feedback based on a more comprehensive learner model.

In order to facilitate the use of serious games in a tutoring context (game-based tutoring), standards are needed to support the linkage of game actions to learning objectives in the tutor. To this end, Sottilare and Gilbert [5] recommend the development of two standard interface layers, one layer for the game and one for the tutor. The game interface layer captures entity state data (e.g., behavioral data represented in the game), game state data (physical environment data), and interaction data, and passes this information to the tutor interface layer. The tutor interface layer passes data from the game to the instructional engine which develops strategies and tactics (e.g., feedback and scenario changes) which are passed back to the game to initiate actions (e.g., non-player character provides feedback or challenge level of scenario is increased).

Additional options for reuse should be explored to minimize/eliminate the amount of authoring required by ITS designers and developers. The ability to structure approaches for configuring a variety of tutoring experiences and experiments is discussed next.

2.3 Configuring tutoring experiences and experiments

Another element of authoring is the ability to easily configure the sequence of instruction by reusing standard components in a script. This is accomplished in GIFT though a set of XML configuration tools used to sequence tutoring and/or experiments. Standard tools include, but are not limited to functional user modeling, learner modeling, sensor configuration, domain knowledge file authoring, and survey authoring which are discussed below.

While not yet implemented in GIFT, functional user models are standard structures and graphical user interfaces used to facilitate tasks and access to information that is specific to the type of user (e.g., learners, subject matter experts, instructional system designers, system developers, trainers/instructors/teachers, and scientists/researchers).

Learner models are a subset of function user models used to define what the ITS needs to know about the learner in order to inform sound pedagogical decisions per the adaptive tutoring learning effect model. The learner configuration authoring tool provides a simple tree structure driven by XML schema which prevents learner model authoring errors by validating inputs against the learner model XML schema. This configuration tool also provides ability to validate the learner model using GIFT source without having to launch the entire GIFT architecture. Inputs to the learner modeling configuration include translators, classifiers, and clustering methods which use learner data to inform learner states (e.g., cognitive and affective).

The sensor configuration authoring tool allows the user to determine which sensors will be used during a given session and which translators, classifiers, and clustering methods the sensor data will feed. Again, this is an XML-based tool which allows the user to select a combination of behavioral and physiological sensor to support their tutoring session or experiment. Several commercial sensors have been integrated into GIFT through plug-ins.

Survey authoring is accomplished through the GIFT survey authoring system (SAS) which allows the generation and retrieval of questions in various formats (e.g., true/false, multiple choice, Likert scales) to support assessments and surveys to support tailoring decisions within GIFT. Through this tool, questions can be associated with assessments/surveys and these in turn can be associated with a specific tutoring event or experiment.

Domain authoring is accomplished through the domain knowledge file authoring tool. This tool allows an instructional designer to sequence events (e.g., scenarios, surveys, content presentation). GIFT currently support various tutoring environments expand the flexibility of course construction. These include Microsoft PowerPoint for content presentation, surveys and assessments from the GIFT SAS, serious games (e.g., VMedic and Virtual BattleSpace (VBS) 2). More environments are needed to support the variety of tasks that might be trained using GIFT.

3 Motivations for expert instruction

Significant research has been conducted to model expert human tutors and to apply these models to ITS to make them more adaptive to the needs of the learner without the intervention of a human instructor. The INSPIRE model [6] [7] is noteworthy based on the extensive scope of this studies that led to this model. Person and others [8] [9] seek to compare and contrast how human tutors and ITS might most effectively tailor tutoring experiences.

For its initial instructional model a strategy-tactic ontology, the *engine for Macro-Adaptive Pedagogy* (eMAP), was developed based on Merrill's Component Display Theory [10], the literature, and variables that included the type of task (e.g., cognitive, affective) and instruction (e.g., individual, small group instruction). Instructional strategies are defined as domain-independent policies that are implemented by the pedagogical engine based on input about the learner's state (e.g., cognitive, affective, domain-independent progress assessment (at expectation, below expectation, or above expectation)). Strategies are recommendations to the domain module in GIFT which selects a domain-dependent tactic (action) based on the strategy type (e.g., prompt, hint, question, remediation) and specific instructional context, where the learner is in the instructional content.

A goal for GIFT is for it to be a nexus for capturing best practices from tutoring research in a single place where scientists can compare the learning effect of each model and then evolve new models based on the best attributes of each model analyzed. To support this evolution, GIFT includes a testbed methodology called the *analysis construct* which is discussed below.

4 Motivations for an effect analysis testbed

As noted in the previous section, GIFT includes an analysis construct which is not only intended to evolve the development of expert instructional models, but is also available to analyze other aspects of ITS including learner modeling, expert modeling, and domain modeling. The notion of a GIFT analysis construct shown in Figure 2 was adapted from Hanks, Pollack, and Cohen's testbed methodology [11].

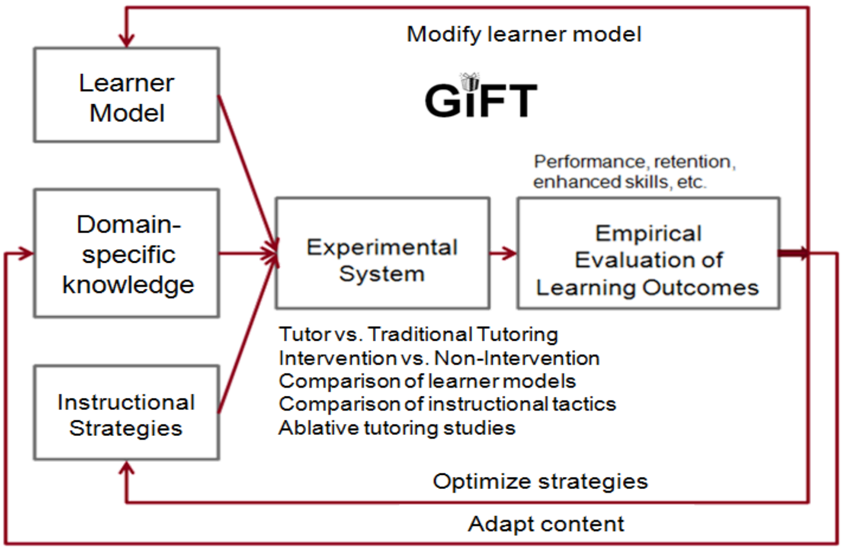

Modify learner model
GIFT
Learner Model
Domain-specific knowledge
Instructional Strategies
Experimental System
Performance, retention, enhanced skills, etc.
Empirical Evaluation of Learning Outcomes
Tutor vs. Traditional Tutoring
Intervention vs. Non-Intervention
Comparison of learner models
Comparison of instructional tactics
Ablative tutoring studies
Optimize strategies
Adapt content

Figure 2: GIFT analysis construct

A great benefit of GIFT's analysis construct it is ability to conduct comparisons of whole tutoring systems as well as specific components (either entire models or specific model elements). To date, ITS research has been limited in its ability to conduct such comparative analyses due to the high costs associated with redesign and experimentation. This construct can be leveraged to assess the impact and interplay of both learner characteristics directly contributing to the learning process (i.e., abilities, cognition, affect, learning preferences, etc.) and those that are external and indirectly effect the learning process (i.e., perceptions of technology, the ITS interface, and learning with technology, etc.). Similarly, GIFT can provide formative and summative assessments to identify the influence of various instructional strategies and tactics; based on these assessments, GIFT is able to better improve and guide instruction dynamically and more effectively.

Across all levels of education and training populations, regardless of the mode of instruction (i.e., live, virtual, or constructive), a paradigm shift in the learning process is occurring due to the evolution of technology and the increase in ubiquitous computing. This notion has become noticeably apparent over the last few years. Even Bloom's revised taxonomy has been recently updated to account for new actions, behaviors, processes, and learning opportunities brought forth by web-based technology advancements [12]. Moreover, with the increasing recognition of the importance of individual learning differences in instruction, GIFT can ultimately be able to support the educational framework and principles of the universal design for learning (UDL) [13, 14]. This framework highlights the need for multiple means of r*epresen-*

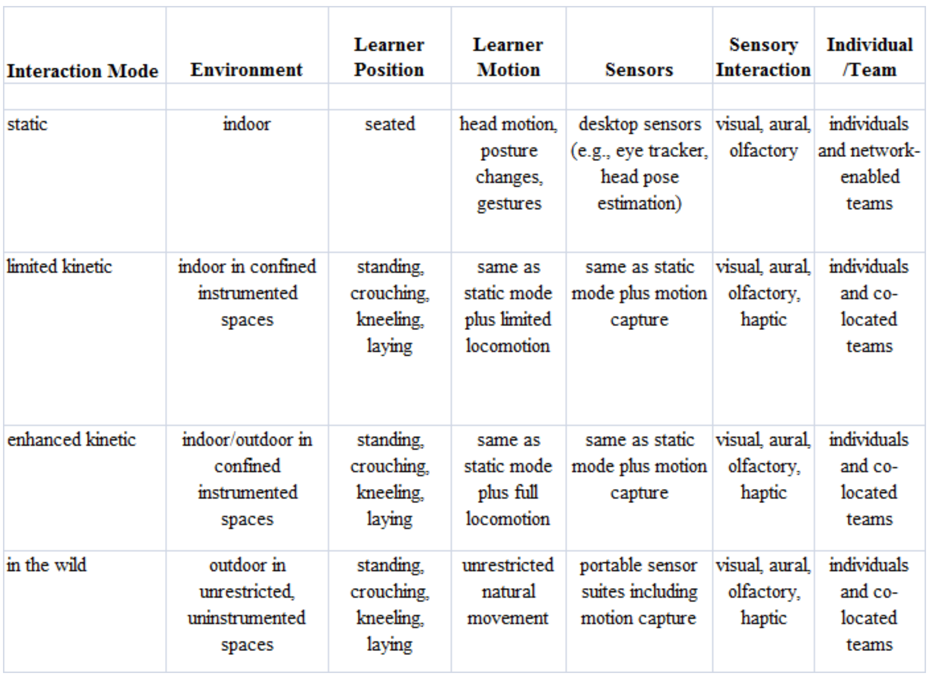
tation, *expression*, and *engagement* to reduce barriers of learning and provide fruitful learning experiences for all types of learners. While this concept has evolved over the past decade, practicality and experimentation to progress this notion to true reality has been limited. However, GIFT's analysis construct can be used to access the effectiveness of UDL principles in an empirically-driven fashion.

5 Expanding the horizons of ITS through future GIFT capabilities

The potential of GIFT is dependent on two primary objectives: 1) focus research and best practices into authoring, instructional, and analysis tools and methods within GIFT to enhance its value to the ITS community and 2) expanding the horizons of traditional ITS outside the bounds of traditional ITS. This section concentrates on examining areas for future development which will expand the current state-of-practice for ITS including tutoring domains, interaction modes, and automation processes for authoring.

The application of ITS technologies has largely been limited to one-to-one, well-defined tutoring domains where information, concepts, and problems are presented to the learner and the learner's response is expected to correspond to a single correct answer. This works well for mathematics, physics and other procedurally-driven domains (e.g., first aid), but not as well for ill-defined domains (e.g., exercises in moral judgment) where there might be more than one correct answer and these answers vary only by their level of effectiveness. It should be a goal of the ITS community to develop an ontology for use developing and analyzing tutors for ill-defined domains.

Traditional tutors have also been generally limited to static interaction modes where a single learner is seated in front of a computer workstation and interaction is through a keyboard, mouse, or voice interface. Methods to increase the learner's interaction and range of motion are needed to move ITS from cognitive and affective domains to psychomotor and social interaction domains. It should be a goal of the ITS community to develop additional interaction modes to support increasingly natural training environments for both individuals and teams as shown in Table 1.



Interaction Mode	Environment	Learner Position	Learner Motion	Sensors	Sensory Interaction	Individual /Team
static	indoor	seated	head motion, posture changes, gestures	desktop sensors (e.g., eye tracker, head pose estimation)	visual, aural, olfactory	individuals and network-enabled teams
limited kinetic	indoor in confined instrumented spaces	standing, crouching, kneeling, laying	same as static mode plus limited locomotion	same as static mode plus motion capture	visual, aural, olfactory, haptic	individuals and co-located teams
enhanced kinetic	indoor/outdoor in confined instrumented spaces	standing, crouching, kneeling, laying	same as static mode plus full locomotion	same as static mode plus motion capture	visual, aural, olfactory, haptic	individuals and co-located teams
in the wild	outdoor in unrestricted, uninstrumented spaces	standing, crouching, kneeling, laying	unrestricted natural movement	portable sensor suites including motion capture	visual, aural, olfactory, haptic	individuals and co-located teams

Table 1. ITS interaction modes

Automation processes should be developed to support authoring of expert models, domain models, and classification models for various learner states (cognitive, affective, and physical). Data mining techniques should be optimized to define not only expert performance, but also levels of proficiency and expectations based on a persistent (long-term) learner model. Again, data mining techniques are needed to reduce the time and cost to author domain models including automated path planning for courses based on the hierarchical relationship of concepts, the development of misconception libraries based on course profiles, feedback libraries (e.g., questions, prompts) based on readily available documentation on the internet and from other sources .

6 References

1. Sottilare, R.A., Brawner, K.W., Goldberg, B.S., & and Holden, H.K. (2012). The Generalized Intelligent Framework for Tutoring (GIFT). Orlando, FL: U.S. Army Research Laboratory – Human Research & Engineering Directorate (ARL-HRED).
2. Sottilare, R. (2012). Considerations in the development of an ontology for a Generalized Intelligent Framework for Tutoring. *International Defense & Homeland Security Simulation Workshop in Proceedings of the I3M Conference*. Vienna, Austria, September 2012.
3. 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.

4. 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.
5. 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.
6. Lepper, M. R., Drake, M., & O'Donnell-Johnson, T. M. (1997). Scaffolding techniques of expert human tutors. In K. Hogan & M. Pressley (Eds), *Scaffolding student learning: Instructional approaches and issues* (pp. 108-144). New York: Brookline Books.
7. Lepper, M. and Woolverton, M. (2002). The Wisdom of Practice: Lessons Learned from the Study of Highly Effective Tutors. In J. Aronson (Ed.), Improving academic achievement: impact of psychological factors on education (pp. 135-158). New York: Academic Press.
8. Person, N. K., & Graesser, A. C., & The Tutoring Research Group (2003). Fourteen facts about human tutoring: Food for thought for ITS developers. Artificial Intelligence in Education 2003 Workshop Proceedings on *Tutorial Dialogue Systems: With a View Toward the Classroom* (pp. 335-344). Sydney, Australia.
9. Person, N. K., Kreuz, R. J., Zwaan, R. A., & Graesser, A. C. (1995). Pragmatics and pedagogy: Conversational rules and politeness strategies may inhibit effective tutoring. *Cognition and Instruction*, 13(2), 161–188.
10. Merrill, M. D. (1983). Component Display Theory. In C. M. Reigeluth (Ed). *Instructional-design theories and models: An overview of their current status*, pp.279-333. Hillsdale, NJ: Lawrence Erlbaum Associates.
11. Hanks, S., Pollack, M.E. and Cohen, P.R. (1993). Benchmarks, Test Beds, Controlled Experimentation, and the Design of Agent Architectures. *AI Magazine*, Volume 14 Number 4.
12. Churches, A. 2009. Retrieved April 29, 2013 from http://edorigami.wikispaces.com/Bloom's+Digital+Taxonomy.
13. King-Sears, M. (2009). Universal Design for Learning: Technology and Pedagogy, *Learning Disability Quarterly*, 32(4), 199-201.
14. Rose, D.H. and Meyer, Anne. (2002). Teaching Every Student in the Digital Age: Universal Design for Learning. Association for Supervision and Curriculum Development, ISBN-0-87120-599-8.

7 Acknowledgment

Research was sponsored and conducted by the Army Research Laboratory. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of the 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. Addition information about GIFT can be found at **www.GIFTtutoring.org**.

Authors

Robert A. Sottilare, PhD serves as the Chief Technology Officer (CTO) of the Simulation & Training Technology Center (STTC) within the Army Research Laboratory's Human Research and Engineering Directorate (ARL-HRED). He also leads adaptive tutoring research within ARL's Learning in Intelligent Tutoring Environments (LITE) Laboratory where the focus of his research is in automated authoring, instructional management, and analysis tools and methods for intelligent tutoring systems. His work is widely published and includes recent articles in the *Cognitive Technology* and the *Educational Technology* Journals. Dr. Sottilare is a co-creator of the Generalized Intelligent Framework for Tutoring (GIFT). He received his doctorate in Modeling & Simulation from the University of Central Florida with a focus in intelligent systems. In January 2012, he was honored as the inaugural recipient of the U.S. Army Research Development & Engineering Command's Modeling & Simulation Lifetime Achievement Award.

Heather K. Holden, Ph.D. is a researcher in the Learning in Intelligent Tutoring Environments (LITE) Lab within the U.S. Army Research Laboratory – Human Research and Engineering Directorate (ARL-HRED). The focus of her research is in artificial intelligence and its application to education and training; technology acceptance and Human-Computer Interaction. Dr. Holden's doctoral research evaluated the relationship between teachers' technology acceptance and usage behaviors to better understand the perceived usability and utilization of job-related technologies. Her work has been published in the Journal of Research on Technology in Education, the International Journal of Mobile Learning and Organization, the Interactive Technology and Smart Education Journal, and several relevant conference proceedings. Her PhD and MS were earned in Information Systems from the University of Maryland, Baltimore County. Dr. Holden also possesses a BS in Computer Science from the University of Maryland, Eastern Shore.

Unwrapping GIFT

A Primer on Developing with the Generalized Intelligent Framework for Tutoring

Charles Ragusa, Michael Hoffman, and Jon Leonard

Dignitas Technologies, LLC, Orlando, Florida, USA
{cragusa,mhoffman,jleonard}@dignitastechnologies.com

Abstract. The Generalized Intelligent Framework for Tutoring (GIFT) is an open-source, modular, service-oriented framework which provides tools, methods and services designed to augment third-party training applications for the purpose of creating intelligent and adaptive tutoring systems. In this paper we provide a high-level overview of GIFT from the technical perspective, and describe the key tasks required to integrate a new training application. The paper will be most helpful for software developers using GIFT, but may also be of interest to instructional designers, and others involved in course development.

Keywords: Adaptive Tutoring, Intelligent Tutoring, Framework, Pedagogy

1 Introduction

The Generalized Intelligent Framework for Tutoring (GIFT) is a framework and tool set for the creation of intelligent and adaptive tutoring systems[1-3]. In its current form GIFT is largely an 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 fielded training systems.

Generally speaking, GIFT is domain and training application agnostic. And, while it can present generic content such as documents, multi-media content, etc.; specialized content is typically presented via an external software system, which we will refer to as a training application (TA). GIFT provides a standardized way to integrate training applications and includes many tools and services required to transform the TA into an intelligent and/or adaptive tutoring system. Services and standards include:

- Standard approach for interfacing training applications
- Domain knowledge representation (including authoring tool)
- Performance assessment
- Course flow (including authoring tool)
- Pedagogical model including micro and macro adaptation
- Learner modeling

- Survey support (with authoring tools)
- Learning management system
- Standardized approach for integrating physiological (and other) sensors

Another key aspect of GIFT is that it is an open source project[1]. Baseline development is currently performed by Dignitas Technologies; however, where appropriate, community developed capabilities will be rolled back into the baseline. In addition, results from current and upcoming experiments, such as pedagogical models, learner models, etc. may eventually be incorporated into future releases. Thus, GIFT is an evolving and ever-improving system, where individual contributions are re-integrated into the baseline for the mutual benefit of all users in the community.

2 Architecture

The GIFT runtime environment uses a service-oriented architecture and consists of several loosely coupled modules, communicating via asynchronous message passing across a shared message bus. Key modules and their primary functions are:

- **Gateway Module**: Connects GIFT to third-party training applications.
- **Sensor Module**: Connects GIFT to physiological sensors in a standardized way.
- **Learner Module**: Models the cognitive, affective, and performance state of the learner [4].
- **Pedagogical Module**: Responsible for making domain-independent pedagogical decisions, using an internal pedagogical model based on learner state.
- **Domain Module**: Performs performance assessment, based on domain-expert authored rules, carries out domain specific implementations of pedagogical actions based on domain-independent pedagogical requests, and (together with the pedagogical module) orchestrates course flow.
- **Tutor Module**: Presents the user interface for tasks such as presentation of surveys, providing feedback, engaging in two-way dialogues, etc.
- **Learning Management System (LMS) Module**: GIFT connection to an external learning management system, for the storage and maintenance of learner records, biographical data, course material etc.
- **User Management System (UMS) Module**: Manages users of the GIFT system, manages surveys and survey data, and provides logging functions.
- **Monitor Module**: Non-critical module, used as control panel for starting and stopping other GIFT modules, and monitoring the state of active GIFT sessions.

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

3 Getting Started with the GIFT Framework

3.1 GIFT Messages

Message Classes. GIFT messages are the sole means of communication between GIFT modules. The Message class hierarchy consists of three classes. The Message base class includes all boiler-plate message fields such as the time stamp, the payload type, an object reference for the optional payload, identification of the source and destination modules, etc. Two subclasses add additional fields appropriate for the GIFT context, such as User Session ID and Domain Session ID.

Message Payloads. Many message types transport data in the optional payload. To support inter-process communication (IPC), the messages and their payloads must comply with an agreed upon encoding and decoding scheme. In GIFT 3.0 the default scheme is Java Script Object Notation (JSON).

Message Types. Every GIFT message has an associated type. The various message types are enumerated in the class mil.arl.gift.common.enums.MessageTypeEnum.

3.2 Interfacing a Training Application using the GIFT Gateway Module

Training Application Considerations. There are two basic requirements that a TA must meet for a satisfactory integration with GIFT. The first is a means to transmit game state from the TA to GIFT. The second is a way for GIFT to exercise some degree of control over the TA. Basic controls such as launching the TA, loading specific content, and shutting down the TA, are very helpful in making a seamless training solution, even though they are not strictly required.

The requirement to communicate game state is immediately met if the TA includes a facility for communicating via a standardized network protocol such as Distributed Interactive Simulation (DIS) protocol. In the absence of such a capability, the TA must be augmented either by leveraging an existing API or by modifying the TA's source code to allow communication of the game state to GIFT via IPC.

Control of the TA by GIFT follows a similar pattern. If an existing protocol exists, it should be used. If not, then custom development will be required. In addition to basic start, load, stop-type control messages, some use cases may require more advanced interactions, discussion of which is beyond the scope of this document.

Creating the Gateway Module Plugin. The process of adapting a TA to the GIFT gateway module involves creating a gateway module plugin. When faced with integrating a new TA, a developer should first ask if one of the existing plugins is suitable for reuse. GIFT 3.0 includes plugins for: DIS, Power Point, TC3Sim, and VBS2. Even if a new plugin is required, these will serve as excellent references.

When developing a new plugin, the primary objective is to implement a concrete subclass of mil.arl.gift.gateway.interop.AbstractInteropInterface. The essential requirements of a new subclass are minimal, but by providing concrete implementations for each of the abstract methods, the plugin will seamlessly operate within the gateway module context. Beyond that, the plugin should implement whatever additional functionality it requires, such as receiving game state messages from the TA and converting them to GIFT messages, and/or receiving GIFT messages (e.g. SIMAN messages) and passing them on to the TA in a way that the TA will understand.

GIFT Messaging. To complete the integration of the TA with the gateway module, at least one GIFT message payload class is needed to represent the game state of the TA. Existing message payload classes that have been used with previously integrated TA's include: TC3GameStateJSON, EntityStateJSON, and PowerPointJSON. If any of these satisfactorily represents the game state from the new TA, then reusing the existing message is advised. However, if none of them are suitable, then a new message will be required. Any new message payload types should be added to the mil.arl.gift.common.enums.MessageTypeEnum class and the appropriate payload class(es) added to the mil.arl.gift.net.api.message.codec.json package.

3.3 Domain Module Modifications and Programming

Overview. At the appropriate time(s) during the execution of a GIFT course, the domain module loads a domain-specific file called the domain knowledge file (DKF). This XML input file contains the domain specific information required by the domain module to carry out several of its key tasks during the learner's interaction with the TA. The first is assessments of the learner's performance on various training tasks encountered during the TA session. It also includes micro-pedagogical mappings of learner state (affective, cognitive, and performance) transitions to named instructional strategies as well as implementation details of those strategies**.**

Integration of any new TA, or even developing a new training course using a previously integrated TA, will typically require DKF authoring as a primary task. In some cases, new custom java coding may also be required, as discussed below.

Domain Knowledge File Authoring. Given that DKF files are XML, they can be edited with any number of text or XML editors, but the preferred method is to use the GIFT-supplied DKF authoring tool (DAT). Using the DAT will enforce the DKF schema as well as perform other validation such as checks against external references.

Before creating a new DKF the user should become familiar with the DKF file format, which is described in the file GIFTDomainKnowledgeFile.htm[2]. In addition, a GIFT release may include one or more test documents (spreadsheets), one of which will contain a step-by-step procedure for authoring a DKF from scratch.

Performance Assessment Authoring. Performance assessment authoring is done within the assessment tag of the DKF file. The basic structure is a task/concept/condition

[2] This and many other documents are contained in the GIFT/docs folder within the GIFT source distribution, which is available for download at http://gifttutoring.org

hierarchy. Tasks have start and end triggers and a set of concepts. Each concept, in turn, will have a set of conditions[3]. It is at the condition level that computation takes place. In fact, you'll notice that each condition tag will contain a conditionImpl tag that refers to a java class responsible for carrying out the performance computation based upon game state received from the TA and inputs encoded in the DKF. Currently, performance values are limited to: unknown, below expectation, at expectation, and above expectation. Beyond the runtime performance assessment, each condition also supports a set of authorable scoring rules and evaluators that together determine the final score for that condition. When scoring rules are present, learners are presented with an after-action review of their performance at appropriate times and scores are written to the LMS.

State Transition Authoring. State transition authoring is performed within the actions tag of the DKF. The basic structure is a list of state transitions, each of which represent a state change in the learner, to which the tutor should react, along with a list of strategy choices (options) that may be used when that particular state change is encountered. In cases where state transitions refer to the learner's performance state, the state transition will have a reference back to a performance node in the assessment section of the DKF.

Instructional Strategy Authoring. Instructional strategy (IS) authoring is also performed within the actions tag of the DKF. Implemented strategies currently include learner feedback, scenario adaptations (changes to the currently executing TA scenario), and request for performance assessment by the domain module. Each strategy entry references a StrategyHandler, which is a specification of the java class responsible for handling authored input contained in the DKF file. The linkage to java code allows substantial flexibility as will be discussed in the next section.

Custom Programming. As described above, the domain module supports a built-in scheme for extending its capabilities for both performance assessment and for instructional strategies. To augment the performance assessment capabilities, a developer codes an implementation of the AbstractCondition interface and then references the implementation class in the appropriate section of the DKF. The key abstract method to be implemented is the handleSimulationMessage[4] method, which takes in a Message as the sole argument, and returns an AssessmentLevelEnum. The message argument is, of course, a representation of the game state that originates in the TA. Developers of new condition implementations should strive to make their code as abstract as possible to allow for the broadest possible reuse[5].

[3] This is a simplified description for the sake of readability. In actuality, concepts support arbitrarily deep nesting of other concepts (i.e., sub-concepts).

[4] The method name reflects GIFT's early development focus on integration with simulations such as VBS2. In future releases the name will be likely be changed to something more generic, such as, "handleGameStateMessage".

[5] Reuse across different TA's, scenarios, domains, etc.

Implementing new instructional strategies is done similarly. Developers provide a concrete implementation of the StrategyHandlerInterface and then reference the implementation class within the DKF. A good example of this is seen with providing feedback to a learner. In the DefaultStrategyHandler, feedback is presented to the learner using the GIFT Tutor User Interface (TUI). However, in a recent experiment, alternative presentations of feedback were required. To satisfy this requirement the TC3StrategyHandler was developed, which allowed feedback strings to be communicated back to TC3Sim for presentation to the learner directly by TC3Sim.

3.4 Surveys and Survey Authoring

GIFT uses the term "survey" to refer generically to any number of interrogative forms presented to the learner via the TUI. GIFT supports survey authoring through its Survey Authoring System (SAS) web application as well as runtime presentation and processing of surveys during execution of a GIFT course. GIFT surveys can be used for a variety of purposes including pre, mid, and post lesson competency assessment; acquiring biographical and demographic information; psychological and learner profiling; and even for user satisfaction surveys. A variety of useful question and response types are supported. Further discussion of the SAS is beyond the scope of this document, but interested readers can consult the GIFTSurveyAuthoringSystemInstructions.htm for additional information.

3.5 Course Authoring

Currently in GIFT, the top-level unit of instruction that learners interact with is a called a course, the specification of which is contained in a dedicated course.xml file. Prior to GIFT 3.0 a course specified a fixed linear flow through a series of course elements; however, with GIFT 3.0 we have introduced support for dynamic flow through course elements, based on macroadaptation strategies.

The primary course elements are surveys, lesson material, and TA sessions. Survey elements administer GIFT surveys that have been previously authored using the SAS. Lesson material elements present browser compatible instructional content such as PDF documents, html pages, or other media files. TA sessions support interactive sessions with a TA such as VBS2, PowerPoint, or other specialized software systems. A fourth course element called "Guidance", which presents textual messages to the learner, exists to support making user-friendly transitions between other course elements. For example at the conclusion of a survey a guidance element might be used to introduce an upcoming TA session.

Course.xml files are authored using the Course Authoring Tool (CAT). For linear flow, the author uses the CAT to specify the various elements of the course along with any necessary inputs. For dynamic flow, authoring involves selecting when in the course flow a branch point is appropriate. The branch point specifies that the macro pedagogical model should gather a list of metadata attributes based on the current learner state when deemed necessary. This collection of metadata attributes is then provided to the domain module as search criteria over the domain content resources for the current course. As the search discovers domain content matching the metadata attributes of interest, paradata files are used to drill down the list of possible content to display based on usage data. The end result is that the domain module is able to present content based on the learner state and pedagogy recommendations.

3.6 Learner Module

The Learner Module is responsible for managing learner state, which can include short term and predicted measures of cognitive and affective state as well as other long term traits. Inputs used to compute state can originate from multiple sources including TA performance assessments sent from the domain module, sensor data from the sensor module, survey responses, and long term traits stored in the LMS.

To date, the GIFT team has focused on computing learner state from sensor data received from the sensor module. The processing framework employs a pipeline architecture which allows the developer to chain concrete implementations of abstract data translators, classifiers, and predictors. Customized pipelines can be created for each sensor type and/or groups of sensors.

Creation of pipelines using existing java implementation classes is performed using the Learner Configuration Authoring Tool, which is launched using scripts/ launchLCAT.bat. Currently defined pipelines can be found in GIFT/config/learner/LearnerConfiguration.xml.

Developers requiring customized implementation classes are referred to the API docs and source code in the mil.arl.gift.learner package. Key abstract classes include AbstractClassifier, AbstractBasePredictor, and AbstractSensorTranslator.

Measurement, representation, and application of learner state are areas of active research and future version of the GIFT learner module will incorporate relevant research outcomes to enhance its capabilities.

3.7 Pedagogical Module

The pedagogical module is responsible for making pedagogical decisions based on learner performance and state. Its primary objective is to reason on the available information, and then influence the training environment to maximize the learning effectiveness for each individual learner using the system. The rules, algorithms, and heuristics that provide the basis for making pedagogical decisions in a domain-independent way are generally referred to as the pedagogical model. One near term goal of GIFT is to provide a framework upon which intelligent tutoring researchers can easily integrate, test and validate a variety of pedagogical models.

In GIFT 3.0, there are two pedagogical models in place: a micro and a macro model. The micro model uses the state transitions information authored in the DKFs as described in previous sections. The macro model is based on research gathered by IST on macro adaptive pedagogy findings [5] which has been encoded as an XML file in GIFT. This XML file is used to configure the macro adaptive pedagogical model when the pedagogical module is started. The information contains a tree-like structure specifying useful metadata for different types of learner state characteristics. This model will continue to be developed after GIFT 3.0.

3.8 Learning Management System (LMS) Module

The GIFT LMS module is a surrogate for an external LMS. In the future, a commercial grade LMS system may be integrated to maintain a variety of data, including student records, course material, and other learning resources. However, in the current version of GIFT, the LMS implementation is an SQL database, designed simply to store and maintain learner records for GIFT courses that have been completed. Aside from developers engaged in integration of GIFT with a production LMS system, very few developers will have a need to modify the LMS module**.**

3.9 User Management System (UMS) Module

The UMS module is supports three major functions: management of users; storage and maintenance of the surveys, survey questions and learner responses to surveys; and message logging. None of these functions are likely targets for development for new GIFT users; however, the logging feature is very important for researchers.

The UMS-managed log files contain every message sent between the various modules during each GIFT session. Using the GIFT Event Reporting Tool (ERT) researchers can apply filters to the log files to isolate messages of interest and perform analysis and data mining that can be used to construct new models.

3.10 Tutor Module: User Interface Considerations

Users interact with GIFT via the TUI, which is a web application that connects to GIFT on the back end. As of GIFT 3.0, Internet Explorer 9.0 is the browser of choice, in accordance with the current U.S. Army mandate [6]. Learner interactions with the TUI include: user login, surveys, feedback, after-action review, interactive dialogues, learning material presentation, etc.

3.11 Monitor Module

As of GIFT 3.0 the monitor module is largely a tool used to launch various GIFT modules and serve as a monitor of a running GIFT session. It is an unlikely development target for new GIFT users. Use of the Monitor Module is described in GIFTMonitor(IOS-EOS)Instructions.htm.

3.12 Sensor Module and Sensor Configuration

The Sensor Module provides a standardized approach to acquiring data from sensors measuring some aspect of Learner State. Currently integrated sensors include: EEG (Emotiv), Electro Dermal Activity (QSensor), Palm temperature and humidity (via instrumented mouse), Zephyr-Technology BioHarness, Inertial Labs Weapon Orientation Module (WOM), USC/ICT Multisense, and Microsoft Kinect.

Sensor data are sent to the learner module to become part of the learner state and potentially used by the pedagogical module. Time-stamped sensor data are also written to log files making them available for post-run analysis by researchers.

The sensor module is configured pre-runtime by editing the SensorConfig.xml file using the Sensor Configuration Authoring Tool (SCAT). The SensorConfig.xml file specifies which sensors should be activated by the sensor module, which plugin (java class) to load to access the sensor hardware, as well as any specialized configuration data. In addition, the SensorConfig.xml includes specification of Filters and Writers, which control the filtering of raw sensor data and writing of sensor data to log files. Users can specify which sensor configuration file is used by editing the GIFT/config/sensor/sensor.properties file.

Developers using one of the previously integrated sensors can, in most cases, limit their focus to editing of the SensorConfig.xml file using the SCAT. Developers integrating new sensors will need to write java code. The key coding task for required for creating a new sensor plugin is to implement a concrete subclass of AbstractSensor. Developers may also want to subclass AbstractSensorFilter and/or AbstractFileWriter, though there are default implementations of these classes that will suffice for many applications.

4 Conclusion

GIFT is a highly configurable and extensible open-source framework designed to support a wide range of intelligent and adaptive tutoring applications. Its modularity and configurability make it well suited for a variety of research efforts.

Configuration and customization opportunities are available at a number of levels ranging from minor editing of text-based configuration files to creation of new java classes. Basic module settings are configurable in dedicated java properties files located in GIFT/config subfolders. More sophisticated configurations reside in XML files, which, depending on the purpose, may reside in a GIFT/config subfolder (e.g. SensorConfig.xml) or alongside the domain content (e.g., course.xml and dkf.xml). GIFT includes specialized editors/authoring tools for many of these files.

As an open-source project, users also have the ability to extend GIFT by modifying source code. In key areas where user extensions are anticipated, GIFT uses appropriate object oriented abstractions. Developers are then able to create their own customized implementation classes, and specify their use at runtime by edits made to the corresponding XML file.

Interested parties are encouraged to register on the GIFT Portal at (http://gifttutoring.org).

5 References

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

ing Systems (CBTS). In *The Interservice/Industry Training, Simulation & Education Conference (I/ITSEC)*(Vol. 2012, No. 1). National Training Systems Association.
4. 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. In *The Interservice/Industry Training, Simulation & Education Conference (I/ITSEC)* (Vol. 2012, No. 1). National Training Systems Association.
5. Goldberg, B., Brawner, K., Sottilare, R., Tarr, R., Billings, D. R., & Malone, N. (2012). Use of Evidence-based Strategies to Enhance the Extensibility of Adaptive Tutoring Technologies. In *The Interservice/Industry Training, Simulation & Education Conference (I/ITSEC)* (Vol. 2012, No. 1). National Training Systems Association.
6. Information Assurance Vulnerability Alert 2012-A-0198 ARMY IAVA - Revised-2013-A-5001 {Release Jan 11 2013}.

Authors

Charles Ragusa is a senior software engineer at Dignitas Technologies with over thirteen years of software development experience. After graduating from University of Central Florida with a B.S. in computer science, Mr. Ragusa spent several years at SAIC working on a variety of R&D projects in roles ranging from software engineer and technical/integration lead to project manager. Noteworthy projects include the 2006 DARPA Grand Challenge as an embedded engineer with the Carnegie Mellon Red Team, program manager of the SAIC CDT/MRAP IR&D project, and lead engineer for Psychosocial Performance Factors in Space Dwelling Groups. Since joining Dignitas Technologies in 2009, he has held technical leadership roles on multiple projects, including his current role as the principal investigator for the GIFT project.

Michael Hoffman is a software engineer at Dignitas Technologies with over seven years of experience in software development. Upon graduating from the University of Central Florida with a B.S. in Computer Science, he spent a majority of his time on various OneSAF development activities for SAIC. He worked on the DARPA Urban Challenge, where he provided a training environment for the robot by simulating AI traffic and the various sensors located on Georgia Tech's Porsche Cayenne in a OneSAF environment. Soon after earning a Master of Science degree from the University of Central Florida, Michael found himself working at Dignitas. Both at Dignitas and on his own time, Michael has created several iPhone applications. One application, called the Tactical Terrain Analysis app, provides mobile situation awareness and can be used as a training tool for various real world scenarios. More recently he has worked to determine if unobtrusive sensors can be used to detect an individual’s mood during a series of computer interactions. Michael excels in integrating both software and hardware systems such as third party simulations and sensors. Michael has been the lead software engineer on GIFT since its inception over two years ago.

Jon Leonard is a junior software engineer at Dignitas Technologies. He graduated from the University of Central Florida with a B.S. in Computer Science where he gained experience in concepts such as computer learning and computer graphics by developing an augmented reality Android video game. At Dignitas, among several other projects, he has worked on a simulation environment for M1A1 Abrams and Bradley tank gunnery training. His personal efforts include working on open source procedural content generation algorithms and mobile applications. Jon's interest is in solving interesting problems. Jon has been a developer for GIFT since its inception.

GIFT Research Transition: An Outline of Options

How transition in GIFT moves through phases of idea, project, and paper, and to real-world use.

Keith W. Brawner[1]

[1]*Army Research Laboratory*
Keith.W.Brawner@us.army.mil

Abstract. This paper describes the use of the Generalized Intelligent Framework for Tutoring (GIFT) to transition research and findings into use beyond publication. A proposal is submitted to use GIFT as a research platform for community development, with examples of how it provides transition opportunities for individual researchers. Several projects which have already transitioned are discussed, while two projects by the author are specifically shown as examples.

Keywords. Intelligent Tutoring Systems, Research Transition

1 Current Transition Path for Research in the ITS Community

The Generalized Intelligent Framework for Tutoring (GIFT) development is currently performed under contract for the Army Research Laboratory. There any many reasons why the military is interested in training technology in general, and adaptive intelligent training technologies in specific [1]. Fundamentally, the end result of research conducted at ARL is technological advancements which are usable by soldiers, or, succinctly, "Technology Driven. Warfighter Focused".

Technology transition is defined as the process of transferring skills, knowledge, or ability from research (typically performed at university or Government labs) to users who can further develop or exploit these items into products, processes, applications, or services. There are many ways for research projects to transition from research to development, to new product, to lifecycle support. While this innovative diffusion may occur solely through technological 'push' of publishing or the 'pull' user adoption, these typically do not occur without a transition partner [2]. Part of the purpose of ARL is to function as a transition partner: leveraging technology advances made in academic laboratories, developing them into usable products, and transitioning them to developmental support roles.

ITS research has historically transitioned directly to the user, which bypasses the developmental and exploitive portions of a traditional transition. One example of this is a project such as the Cognitive Tutor, which bypassed the "external development" phase through marketing to local school districts. Another example includes the Crystal Island program, which has also transitioned through collaboration with the local school districts, rather than an industrial base. Further examples include AutoTutor transition of Operation ARIES through the facilitating intermediary of Pearson Education, or GnuTutor through open source software release.

Researchers generally face competing desires for their project. As a research goal, they desire to perform research, create findings, publish results, and solve interesting problems. A researcher may have a related goal, which competes for their time: the desire for their technology to be useful to a population of users. Given finite resources, the individual or organization must compromise one of these goals to facilitate the other. A two-way facilitating transition partner would allow the researcher to see their creation used and obtain meaningful feedback while maintaining research pursuits.

ARL in general, and the GIFT project in specific, have a goal of facilitating this research transition. This goal is not empty talk, as the repackaging and transition of several research projects has already occurred programmatically. In addition to being an ARL researcher, the author is anticipated to obtain a doctorate at the University of Central Florida in August 2013. Research done at ARL and UCF alike are both transitioning to the field through the GIFT, and will be described in this paper. The author will outline how you can use GIFT to transition your research, give examples of projects which have done so, and describe the benefits of this approach.

2 Proposal for a Community Research Platform

GIFT is intended to be both a community platform and growing standard [3, 4]. This fundamentally offers several advantages, a short selection of which is described below:

- Like any open source software approach, a researcher or developer is able to build upon the work of others. This magnifies the ability of an individual developer to contribute.
- Like any community project, a developer is able to quickly see the use of their work. An individual developer/researcher is able to quickly access a population of users of their research, which magnifies their individual impact.
- ITS technology can be leveraged against a broad amount of training content, while keeping the same core functionality. This magnifies the use of the product.
- The ITS technology can improve through various software versions, which improves learning while costing little or nothing for implementation. Content is used in a more useful fashion, making the use of an incrementally updated project attractive.
- A researcher or developer can use standardized tools to create, modify, or adjust individual items for the purpose of experimentation, evaluation, and validation.
- Experimental comparisons can be conducted fairly at multiple locations, with multiple populations. This allows the research conducted within the framework to be fairly compared.
- A researcher can leverage tools which make the interpretation of data easier. A shared set tools has been of aid to other researchers in Educational Data Mining [5].

3 Re-GIFT-ing: models of transition

There are several models of transition which can be used with varying levels of researcher interaction and levels of opportunity. Transition into GIFT may be through a tool, a compatible software or hardware product, a plug-in, a releasable item, or a piece of software integrated into an official baseline. These differing modes of transition are summarized in **Error! Reference source not found.** alongside the required user interaction, an example of a project which has followed this transition path, and the potential impact that it has to the field.

The first project to discuss is the tool created by the Personalized Assistant for Learning (PAL) for data analysis [6]. During the course of a PAL experiment with GIFT, the developers found it helpful to have a tool to parse through GIFT data. After developing this tool, they provided this it back to the community through simply posting it on the http://www.gifttutoring.org forums. An author following this transition path may host a "**GIFT tool**" on their own site, make it available to only their lab, or other method. To the author's knowledge, no one has used or modified this tool outside of their laboratory. However, others have the opportunity to use this tool and improve on it, and its functionality has directed the development of a more thorough tool available within the GIFT Release: the Event Reporting Tool (ERT).

The next project, and method of transition, to discuss is the eMotive EEG library. The eMotive EEG was found helpful in other research conducted by the author [7], and was incorporated into GIFT as a software library interface. The purchase of an eMotive EEG headset gives the developer access to the library. The fact the GIFT supports easy integration of the sensor makes it so that each GIFT user is a potential eMotive customer, which benefits eMotive. Transition of research as a "**GIFT compatible**" product involved little interaction with the developers, but may be unsupported in future releases. While developer involvement is low, the potential impact is similarly low.

Continuing to use sensors as an example, the next project to discuss is the Q-Sensor project, which transitioned in a way which is different from the previous versions. All software required to integrate an Affectiva Q-Sensor is provided freely to the GIFT community, as part of a **"GIFT plugin"**. Changes made to the Q-Sensor are supported in future versions of GIFT and the plugin is released in the current GIFT 3.0 version. To date, this type of transition has resulted in the use of Q-Sensor technology in a minimum of two different experiments, with three pilot trials. This has occurred with little interaction from the Q-Sensor developers.

There are now several complete programmed packages which are released with the GIFT version. One of these is the medical instruction and assessment game "vMedic", which contains several scenarios which have GIFT tutoring. Another example is the Human Affect Recording Tool (HART), developed by Ryan Baker [8], which enables affective coding of behavior. Both of these programs have reached a wider audience through leveraging **"GIFT releasable"** transition, with some work required by the developer. The developer of each of these programs targeted use within GIFT as part of the model of development. Each of these programs is provided back to the community as downloadable software packages on www.gifttutoring.org. In this fashion, the vMedic program has reached a significantly wider audience and the HART app has seen distribution and citation.

Lastly, one can transition source code directly into the GIFT baseline via a **"GIFT integration"**, in anticipation of the next release. The work required to integrate into the GIFT framework is done by the developer, before giving it back to www.gifttutoring.org. While this requires more work, it is able to reach a wider audience, and is automatically carried forward into each future release. This is the only release path which is thoroughly tested and vetted prior to each version. This allows for the broadest application of the developed technology.

Table 2. Examples of various GIFT transitions, projects which used this transition method, and levels of interaction provided

Type of Transition	**Example of project**	**User interaction**	**Potential Impact**
GIFT tool	PAL Tool	None	Low
GIFT compatible	eMotive EEG	None	Low
GIFT plug-in	Q-Sensor	Low	Medium
GIFT releasable	HART, vMedic	Medium	Medium
GIFT integration	GSR filtering, MultiSense	High	High

4 Two Research Transition Stories: GSR Filtering, realtime modeling

In this section, the author will tell two stories research transition where first-hand experience was obtained. The first of these stories involves the transition of a new GSR sensor filtering method, available in GIFT 2.0, while the second focuses on a larger piece of work which has intended availability in GIFT 4.0. The aim of this section is to give an example of how an idea becomes a deliverable.

4.1 GSR Filtering

The first project idea is that a realtime sensor filter may be able to collect meaningful measures of affective/cognitive state in realtime. The idea behind this project is that the author was unaware of relevant feature extraction techniques, or implementations, for several datastreams of interest. A dataset was used which collected both ECG and GSR measures while participants experienced various training events [7]. It was hypothesized that meaningful measures of cognition and affect could be extracted from these sensor datastreams.

It was found that meaningful measures of cognition and affect could be extracted, including statistical measures and signal power measures, borrowing from the field of digital signal processing. It is possible that these techniques could be leveraged into an intelligent tutoring system. These results were then published [9].

Just because a method has been published to be useful does not mean that industrial or academic partners and collaborators will take it upon themselves to read an academic paper, implement the algorithm, and put it in their system. The more that an individual developer can do to help this process, the quicker transition of the research will be [2]. One way to do this is to merge the work of a researcher with a project which is already transitioning to industry. GIFT represents this possibility.

The idea, project, and paper on GSR filtering has transitioned into GIFT via the "GIFT integration" route. Every researcher which downloads GIFT (which is compatible with a GSR sensor) is able to implement the developed feature extraction, do their own experiment and draw their own conclusions. Furthermore, any ITS constructed from the GIFT framework and tools already has this implementation, and the development of a student model which uses this information progresses a significant step towards reality. The ECG filtering from the same paper is intended to be released GIFT 4.0.

To date, GSR filtering algorithms have now been provided to over 100 researchers and developers. The author hopes that his work will be found valuable. In either case, the developed research has been placed in the hands of numerous users, which is more valuable than publication alone. If the work is not found valuable, the author would hope that the other researchers are able to improve on the technique, and feed the results to other researchers through a similar transition path.

4.2 Realtime modeling

The second project idea is that individualized models of learner affect/cognition may be able to be created in realtime. The idea behind the second project is that generalized models of affect and cognition are difficult to create. Individualized models can be made, but their quality is known to degrade over time [10]. Realtime modeling and adaptive algorithms may present a solution to the problem.

The realtime modeling project used two datasets [11] and constructed seven total classifiers. The approach used four different types of classification techniques, including neural gasses, resonance theory, clustering, and online linear regression. Each of these techniques was developed with three different schemes for labeling data, including unsupervised, semi-supervised, and fully-supervised.

It was found that semi-supervision had significant contribution to the overall accuracy of the problem. It was also found that realtime affective models could be created with reasonable quality, and that realtime cognitive models are a more difficult problem that requires alternate means in conjunction with the methods presented. These results will be published as a doctoral dissertation later in the year.

Realtime student state assessment is anticipated to be available within GIFT 4.0. Targeting GIFT as a research transition allows industry and academia to benefit from the research, and targets a larger and different audience than publication. Once again, transition of research through GIFT allows larger access, experimentation, citation, and overall exposure.

5 Conclusion/Recommendations

GIFT is a functional Intelligent Tutoring System which has been used as part of several experiments. Research which transitions into GIFT has the potential to be used by a population of learners, instructional designers, and experimenters. Each of these user groups is anticipated to have their own user interface, which can make use of the research transitioned into GIFT, in whichever fashion is implemented.

In addition, GIFT is intended as a research platform, and Army Research Laboratory has plans for development out to 2017. A research transition into GIFT, in any fashion, should be able to reach a community of users for the next four years, at a minimum. The project has potential longevity beyond 2017, with funding from the Army, DoD, or others. Even if not supported by the Army, it will remain in the public domain, able to be improved by anyone in the community. Using GIFT as an exit vector for research ideas has more potential than simple publication, or of hosting an open source project.

Furthermore, the licensing agreement on GIFT does not hinder the individual researcher from capitalizing on their ideas. Two for-profit companies have targeted GIFT as a technology which can support the ability to commercialize their ideas, while others have been in conversation. Other research organizations have proposed or used GIFT to widen their audience and to focus their expertise.

This paper has discussed how some research technology has *already* transitioned to the field using the GIFT entry vector, and how other portions are intended. The concept which the author presents in this workshop paper and presentation is that it is possible to use GIFT as a platform to transition research results into the field of use, while minimizing the effort required by the researcher.

6 References

1. Army, D.o.t., *The U.S. Army Learning Concept for 2015*, 2011: TRADOC.
2. Fowler, P. and L. Levine, *A conceptual framework for software technology transition*, 1993, DTIC Document.
3. Sottilare, R.A., et al. *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. NTSA.
4. Sottilare, R.A., et al., *The Generalized Intelligent Framework for Tutoring (GIFT).* 2012.
5. Koedinger, K.R., et al., *A data repository for the EDM community: The PSLC DataShop.* Handbook of Educational Data Mining, 2010: p. 43-55.
6. Regan, D., E.M. Raybourn, and P. Durlach, *Learning modeling consideration for a personalized assistant for learning (PAL)*, in *Design Recommendations for Adaptive Intelligent Tutoring Systems: Learner Modeling (Volume 1).*2013.
7. Goldberg, B.S., et al., *Predicting Learner Engagement during Well-Defined and Ill-Defined Computer-Based Intercultural Interactions*, in *Proceedings of the 4th International Conference on Affective Computing and Intelligent Interaction (ACII 2011), LNCS,* S.D. Mello, et al., Editors. 2011, Springer-Verlag: Berlin Heidelberg. p. 538-547.
8. Baker, R.S., et al. *Towards Sensor-Free Affect Detection in Cognitive Tutor Algebra*. in *Proceedings of the 5th International Conference on Educational Data Mining*. 2012.

9. Brawner, K. and B. Goldberg. *Real-Time Monitoring of ECG and GSR Signals during Computer-Based Training*. 2012. Springer.
10. AlZoubi, O., R. Calvo, and R. Stevens, *Classification of EEG for Affect Recognition: An Adaptive Approach.* AI 2009: Advances in Artificial Intelligence, 2009: p. 52-61.
11. Carroll, M., et al., *Modeling Trainee Affective and Cognitive State Using Low Cost Sensors*, in *Proceedings of the Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC)*2011: Orlando, FL.

Authors

Keith W. Brawner is a researcher for the Learning in Intelligent Tutoring Environments (LITE) Lab within the U. S. Army Research Laboratory's Human Research & Engineering Directorate (ARL-HRED). He has 7 years of experience within U.S. Army and Navy acquisition, development, and research agencies. He holds a Masters degree in Computer Engineering with a focus on Intelligent Systems and Machine Learning from the University of Central Florida, and will obtain a doctoral degree in the same field in Summer 2013. The focus of his current research is in machine learning, active learning, realtime processing, datastream mining, adaptive training, affective computing, and semi/fully automated user tools for adaptive training content.

Experimentation with the Generalized Intelligent Framework for Tutoring (GIFT): A Testbed Use Case

Benjamin Goldberg[1] and Jan Cannon-Bowers[2]

[1]*United States Army Research Laboratory-Human Research & Engineering Directorate-Simulation and Training Technology Center, Orlando, FL 32826*
{benjamin.s.goldberg@us.army.mil}
[2]*Center for Advanced Medical Learning and Simulation (CAMLS) at the University of South Florida, Tampa, FL 33602*
{jcannonb@health.usf.edu}

Abstract. Computer-Based Tutoring Systems (CBTS) are grounded in instructional theory, utilizing tailored pedagogical approaches at the individual level to assist in learning and retention of associated materials. As a result, the effectiveness of such systems is dependent on the application of sound instructional tactics that take into account the strengths and weaknesses of a given learner. Researchers continue to investigate this challenge by identifying factors associated with content and guidance as it pertains to the learning process and the level of understanding an individual has for a particular domain. In terms of experimentation, a major goal is to identify specific tactics that impact an individual's performance and the information that manages their implementation. The Generalized Intelligent Framework for Tutoring (GIFT) is a valuable tool for this avenue of research, as it is a modular, domain-independent framework that enables the authoring of congruent systems that vary in terms of the research questions being addressed. This paper will present GIFT's design considerations for use as an experimental testbed, followed by the description of a use case applied to examine the modality effect of feedback during game-based training events.

Keywords: generalized intelligent framework for tutoring, instructional strategies, testbed, feedback, experimentation

1 Introduction

The overarching goal of Computer-Based Tutoring Systems (CBTSs) is to enable computer-based training applications to better serve a leaner's needs by tailoring and personalizing instruction [1]. Specifically, the goal is to achieve performance benefits within computer-based instruction as seen in Bloom's 1984 study "the 2-Sigma Problem". Though there is recent debate on the validity of these results [2], this classic experiment showed that individuals receiving one-on-one instruction with an expert tutor outperformed their fellow classmates in a traditional one-to-many condition by an average of two standard deviations. The success of this interaction is in the ability of the instructor to tailor the learning experience to the needs of the individual. Interaction is based on the knowledge level of the learner as well as their performance and reaction (i.e., cognitive and affective response) to subsequent problems and communications [3].

With the recent development of the Generalized Intelligent Framework for Tutoring (GIFT; see Figure 1), a fundamental goal is to develop a domain-independent pedagogical model that applies broad instructional strategies identified in the literature. This framework would then be used to author adaptive environments across learning tasks to produce benefits accrued through one-on-one instruction. At the core of GIFT is pedagogical modeling, which is associated with the application of learning theory based on variables empirically proven to influence performance outcomes [4]. According to Beal and Lee [5] the role of a pedagogical model is to balance the level of guidance and challenge during a learning event so as to maintain engagement and motivation. The notion for GIFT is to identify generalized strategies on both a macro- and micro-adaptive level that can be used to author specific instructional tactics for execution in a managed ITS application. The pedagogical model uses data on '*Who*' is being instructed, '*What*' is being instructed, and the '*Content*' available from which to instruct. In an ideal case, GIFT can identify recommended strategies based on this information, and also provide tools to convert those strategies into specific instructional tactics for implementation.

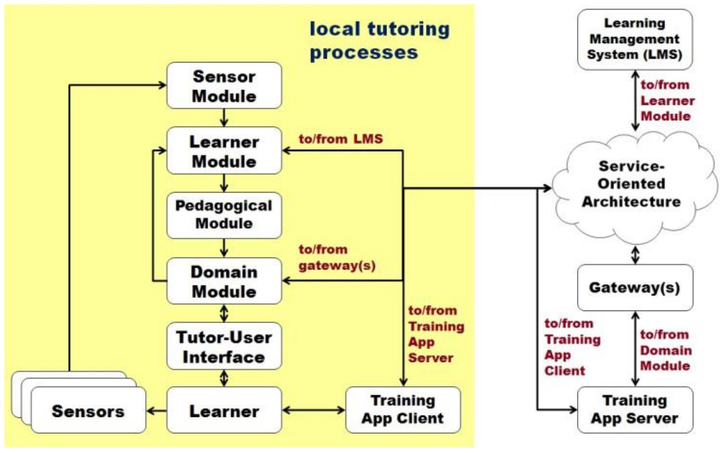


Fig. 3. Generalized Intelligent Framework for Tutoring (GIFT)

Before this conceptual approach of GIFT can be realized, a great deal work needs to be done to identify strategies found to consistently affect learning across multiple domains (codified in the pedagogical model) and the variables that influence the selection of these strategies (expressed in the learner model). In the remainder of this paper, we describe GIFT's functional application as an experimental testbed for conducting empirical research, followed by a descriptive use case of a recent instructional strategy-based experiment examining the effect varying modalities of feedback delivery have on learner performance and engagement within a game-based environment.

1.1 GIFT's Testbed Functionality

For GIFT to be effective across all facets of learning, there are a number of research questions that need to be addressed. These include, but are not limited to: (1) How can GIFT be used to manage the sequence, pace, and difficulty of instructional content before a learning session begins, as well as how to adapt instruction in real-time based on learner model metrics?; (2) What information is required in the learner model to make informed decisions on instructional strategy selection?; (3) How can GIFT best manage guidance and feedback during a learning session based on competency and individual differences?; and (4) What is the optimal approach for delivering GIFT communications to a learner during system interaction?

While GIFT provides the tools necessary to author and deliver adaptive learning applications, an additional function of the framework is to operate as a testbed for the purpose of running empirical evaluations on research questions that will influence future developmental efforts. Empirically evaluating developed models and techniques is essential to ensuring the efficacy of GIFT as a sound instructional tool. To accommodate this requirement, while maintaining domain-independency, GIFT's design is completely modular. This allows for the swapping of specific parts within the framework without affecting other components or models. Modularity enables easy development of comparative systems designed to inform research questions above. The framework is structured to support a variety of experimental design approaches, including ablative tutor studies, tutor vs. traditional classroom training comparisons, intervention vs. non-intervention comparisons, and affect modeling and diagnosis research [6]. The descriptive use case illustrated next is based on an intervention comparison approach.

2 GIFT Experimental Use Case

In this section, we describe in detail the process of using GIFT to design and run a study to evaluate varying methods for communicating information to a learner while they interact with a game-based environment. This experiment was designed to examine varying modality approaches for feedback information delivery during a game-based learning event that is not implicit within the virtual environment (i.e., feedback in the scenario as a result of a player/entity or environmental change). This is influenced by available features present in the GIFT architecture and the benefits associated with research surrounding learning and social cognitive theory [10-11]. The notion is to identify optimal approaches for providing social agent functions to deliver feedback content that is cost effective and not technically intensive to implement. As a result, research questions were generated around the various communication modalities GIFT provides for relaying information back to the learner.

A functional component unique to GIFT is the Tutor-User Interface (TUI). The TUI is a browser-based user-interface designed for collecting inputs (e.g. survey and assessment responses) and for relaying relevant information back to the user (e.g. performance feedback). In terms of providing real-time guided instruction, the TUI can be used as a tool for delivering explicit feedback content (i.e., guidance delivered outside the context of a task that relays information linking scenario performance to training objectives) based on requests generated from the pedagogical model. Because the TUI operates in an internet browser window, it supports multimedia applications and the presence of virtual entities acting as defined tutors. As a potential driver for interfacing with a learner, research is required to evaluate feedback delivery in the TUI and assess its effectiveness in relation to other source modality variations. The overarching purpose of the described research is to determine how Non-Player Characters (NPCs) can be utilized as guidance functions while learning in a virtual world environment and to identify tradeoffs among the varying techniques.

Research questions were generated around the limitation associated with using the TUI during game-based instruction. For a virtual human to be present in the TUI, it requires a windowed display of the interfacing game so the browser can be viewed in addition to the game environment, which may take away from the level of immersion users feel during interaction; thus removing a major benefit with utilizing a game-based approach in education. Specifically, this study will assess whether explicit feedback delivered by NPCs embedded in a scenario environment has a significant effect on identified dependent variables (e.g., knowledge and skill performance, and subjective ratings of flow, workload, and agent perception) when compared to external NPC feedback sources present in the TUI. In terms of serious games, the current research is designed to address how the TUI can be utilized during game-based interactions and determine its effectiveness versus more labor intensive approaches to embedding explicit feedback directly in the game world.

This experiment was the first implemented use of GIFT functioning as a testbed for empirical evaluation. During the process of its development, many components had to be hand authored to accommodate the investigation of the associated research questions. This involved integration with multiple external platforms (e.g., serious game TC3Sim, the Student Information Models for Intelligent Learning Environments (SIMILE) program, and Media Semantics); development of scenarios, training objectives, assessments, and feedback; exploration of available avenues to communicate information; and representing these relationships in the GIFT schema. In the following subsections, we will review the process associated with each phase listed above.

2.1 Testbed Development

GIFT provides the ability to interface with existing learning platforms that don’t have intelligent tutoring functions built within. In these games, learners are dependent on implicit information channels to gauge progress towards objectives. Integrating the game with GIFT offers new real-time assessment capabilities that can be used to provide learner guidance based on actions taken within the environment that map to associated performance objectives.

For the instance of this described use case, the serious game TC3Sim was selected as the learning environment to assess the effect of differing feedback modality approaches. TC3Sim is designed to teach and reinforce the tactics, techniques, and procedures required to successfully perform as an Army Combat Medic and Combat Lifesaver [7]. The game incorporates story-driven scenarios designed within a game-engine based simulation and uses short, goal-oriented exercises to provide a means to train a closely grouped set of related tasks as they fit within the context of a mission [8]. Tasks simulated within TC3Sim include assessing casualties, performing triage, providing initial treatments, and preparing a casualty for evacuation under conditions of conflict (ECS, 2012). For the purpose of the experiment, GIFT had to be embedded within TC3Sim for the function of monitoring performance to trigger feedback that would ultimately influence data associated with the dependent variables of interest.

This required pairing of the two systems so that GIFT could consume game state messages from TC3Sim for assessment on defined objectives, and for TC3Sim to consume and act upon pedagogical requests coming out of GIFT. For this to happen, a Gateway Module had to be authored that serves as a translation layer between the two disparate systems. The Gateway Module was also modified to handle feedback requests that were to be delivered by programs external to the game. This included integration with MediaSemantics, desktop and server software that provides character-based applications and facilitated the presence of a virtual human in the TUI that would act as the tutor entity. Following, enhancements to the Communication Module/TUI had to be employed to support the variations in feedback modalities.

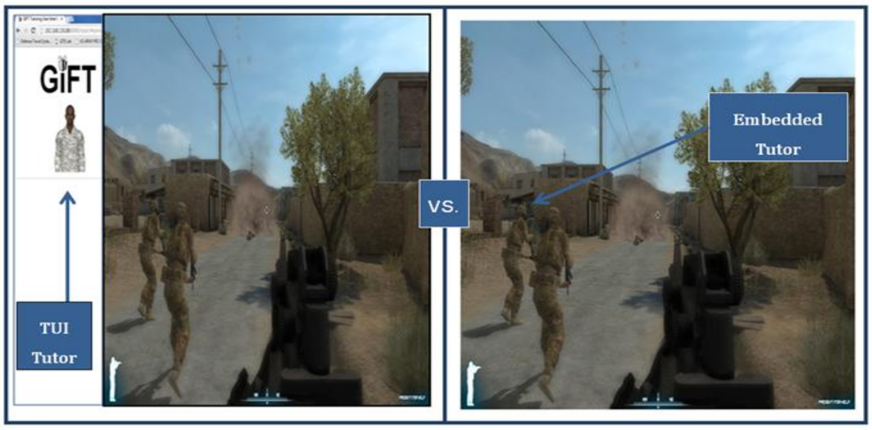


Fig. 4. Feedback Communication Modes

Communication Development. The functional component of GIFT primarily assessed in this research is the Communication Module/TUI, and focused on interfacing approaches for delivering feedback communications during a game-based learning event. For this purpose the major variations associated with the framework took place in GIFT's TUI, as well as identifying approaches for GIFT to manage agent actions within a virtual environment. This required two GIFT/TC3Sim versions with modifications to how the game was visually represented (see Figure 2). With a windowed version of the game, the MediaSemantics character was embedded into the TUI browser and was programmed to respond to feedback requests coming out of the domain module. Furthermore, two additional control conditions were authored to assess whether feedback delivered as audio alone made a difference and a condition with zero feedback to determine whether the guidance had any effect on performance. All participants interacted with the same scenarios, with two conditions including an EPA present in the virtual environment as an NPC. The remaining conditions will receive feedback from external sources to the game. With the functional modifications in place, the next step was designing scenarios, assessments, and feedback scripts.

Scenario Development. With the ability to apply varying techniques of feedback delivery during a game-based learning event, the next step was to design a scenario in TC3Sim to test the effects of all approaches. This requires multiple steps to ensure scenario elements are appropriate so that they lend to accurate inference based on the associated data captured during game interaction. This involved the definition of learning objectives the scenario would entail, associated assessments to gauge performance on objectives, and feedback to apply when performance was deemed poor.

Objectives were informed by competencies identified in ARL-STTC's Medical Training Evaluation Review System (MeTERS) program, which decomposed applied and technical skills for Combat Medics and Combat Lifesavers into their associated tasks, conditions, and standards for assessment purposes (Weible, n.d.). In development of the TC3Sim, the identified competencies were further decomposed into specific learning objectives in terms of enabling learning objectives and terminal learning objectives for each role and task simulated in the game environment. With guiding specifications, a scenario was developed that incorporated decision points for treating a hemorrhage in a combat environment. The scenario was designed to be difficult enough that participants would struggle, resulting in triggered feedback, while not being too difficult that successfully completing the task was impossible.

However, before explicit feedback linked to performance can be delivered in game-based environment, methods for accurately assessing game actions as they relate to objectives is required. The first step to achieve this is properly representing the domain's objectives within GIFT's Domain Knowledge File (DKF) schema by structuring them within the domain and learner model ontology. This creates a domain representation GIFT can make sense of, and results in a hierarchy of concepts that require assessments for determining competency. This association enables the system to track individual enabling objectives based on defined assessments, giving the diagnosis required to provide relevant explicit feedback based on specific actions taken. Following, methods for assessing the defined concepts must be applied that provide information for determining whether an objective has been satisfactorily met. For this purpose, ECS's Student Information Models for Intelligent Learning Environments (SIMILE) was integrated within GIFT.

Student Information Models for Intelligent Learning Environments (SIMILE). An innovative tool used in conjunction with TC3Sim for the purpose standardized assessment is SIMILE (ECS, 2012). In the context of this use case, SIMILE is a rule-engine based application used to monitor participant interaction in game environments and is used to trigger explicit feedback interventions as deemed by GIFT's learner and pedagogical models. In essence, SIMILE established rule-based assessment models built around TC3Sim game-state messages to generate real-time performance metric communication to GIFT. SIMILE monitors game message traffic (i.e., ActiveMQ messaging for this instance) and compares user interaction to pre-established domain expertise defined by procedural rules. As user data from gameplay is collected in SIMILE, specific message types pair with an associated rule authored and look for evidence determining if the rule has been satisfied; that information is then communicated to GIFT, which establishes if there was a transition in performance. Next, that performance state is passed to the learner model. GIFT interprets SIMILE performance metrics for the purpose of tracking progress as it relates to objectives. When errors in performance are detected, causal information is communicated by SIMILE in to GIFT, which then determines the feedback string to deliver.

Feedback Development. Following the completion of linking GIFT's domain representation with SIMILE-based assessments, specific feedback scripts had to be authored that would be presented when the pedagogical model made a 'feedback request'. In the design phase of these prompts, it was recognized that GIFT is dependent on a transition in performance before the pedagogical model can make any decision on what to do next. In the case of the TC3Sim scenario, this requires the player to perform certain actions that denote competency on a concept, but a question is, what information is available to determine they were ignoring steps linked to an objective?

From this perspective, it was recognized that time and entity locations are major performance variables in such dynamic operational environments. Outcomes in hostile environments are context specific, and time to act and location of entities are critical metrics that require monitoring. From there, if a participant had not performed an action in the game or violated a rule that maps to an associated concept, GIFT could provide reflective prompts to assist the individual on what action to perform next. An example applied in the experiment is 'Maintain Cover'. This requires staying out of the streets while walking through a hostile urban environment. For assessment, the player's distance from the street center was monitored, with a defined threshold designating if they maintained appropriate cover. For each concept, rules based on time and locations were identified, and reflective prompts were authored for each concept. Following, audio for each feedback prompt was recorded. This was the final step before the system could be fully developed.

3 Data Collection and Analysis Prep

Data collection was conducted over a five-day period at the United States Military Academy (USMA) at West Point, NY where a total of 131 subjects participated. This resulted in 22 participants for each experimental condition minus the control, which totaled at 21 subjects. The lab space was arranged for running six subjects at a time, with two experimental proctors administering informed consents and handling any technical issues that arose during each session. Once a subject logged in, GIFT managed all experimental procedures and sequencing, allowing the proctors to maintain an experimenter's log for all six machines. This feature shows the true benefit of GIFT in an experimental setting. Once properly configured, GIFT administers all surveys/tests and opens/closes all training applications linked to the procedure, thus reducing the workload on the experimental proctor and enabling multiple data sessions to be administered at a single time. GIFT offers the Course Authoring Tool (CAT) to create the transitions described above. A researcher can author the sequence of materials a participant will interact with, including transition screens presented in the TUI that assist a user in navigating through the materials.

Following the experimental sessions, data must be extracted from associated log files and prepped for analysis. A tool built into GIFT to assist with this process in the Event Reporting Tool (ERT). The ERT enables a researcher to pull out specific pieces of data that are of interest, along with options on how the data is represented (i.e., user can determine if they would like to observe data in relation to time within a learning event or to observe data between users for comparison). The result is a .CSV file containing the selected information, leaving minimal work to prepare for analysis. In this use case, the majority of analysis was conducted in IBM's SPSS statistical software, with the ERT playing a major role in the creation of the master file consumed by the program. This drastically reduced the time required to prep data for analysis, as it removed the need to input instrument responses for all subjects, it structured the data in a format necessary for SPSS consumption (i.e., each row of data represents an individual participant), and produced variable naming conventions listed on the top row.

4 The Way Ahead

GIFT provides a potent testbed in which studies of instructional techniques can be evaluated. Specifically, it allows researchers to investigate how best to implement tenants of intelligent tutoring, including optimal mechanisms for tracking performance, providing feedback and improving outcomes. At the current moment, GIFT provides limited feedback mechanisms that are generally used as formative prompts for correcting errors and reaffirming appropriate actions. New feedback approaches must be explored, such as natural language dialog, to expand the available options for relaying information in game environments. As well, research needs to identify approaches for using environmental cues in the game world to act as feedback functions informed by GIFT. In terms of GIFT as a testbed, advancements need to be applied to the ERT in terms of how data is exported to ease the required post-processing leading to analysis. This includes the ability to segment data in log files based around defined events in the environment that are of interest in analysis. Future research can build on the use case presented and/or conceptualize other investigations that benefit from GIFT.

5 References

1. Heylen, D., Nijholt, A., R., o. d. A., Vissers, M.: Socially Intelligent Tutor Agents. In: T. Rist, R. Aylett, D. Ballin & J. Rickel (Eds.), Proceedings of Intelligent Virtual Agents (IVA 2003), Lecture Notes in Computer Science, 2792: 341-347. Berlin: Springer (2008)
2. VanLehn, K.: The Relative Effectiveness of Human Tutoring, Intelligent Tutoring Systems, and Other Tutoring Systems. Educational Psychologist, 46(4): 197-221 (2011).
3. Porayska-Pomsta, K., Mavrikis, M., Pain, H.: Diagnosing and acting on student affect: the tutor's perspective. User Modeling and User-Adapted Interaction, 18(1): 125-173 (2008).
4. Mayes, T., Freitas, S. d.: Review of e-learning frameworks, models and theories. JISC e-Learning Models Desk Study, from http://www.jisc.ac.uk/epedagogy/ (2004).

5. Beal, C., Lee, H.: Creating a pedagogical model that uses student self reports of motivation and mood to adapt ITS instruction. In: Workshop on Motivation and Affect in Educational Software in conjuctions with the 12th International Conference on Artificial Intelligence in Education (2005).
6. Sottilare, R., Goldberg, B., Brawner, K., Holden, H.: Modular Framework to Support the Authoring and Assessment of Adaptive Computer-Based Tutoring Systems. In: Proceedings of the Inteservice/Industry Training, Simulation, and Education Conference, (2012).
7. S: vMedic. Retrieved from http://www.ecsorl.com/products/vmedic (2012).
8. Fowler, S., Smith, B., & Litteral, C.: *A TC3 game-based simulation for combat medic training* (2005).
9. Weible, J.: Preparing soldiers to respond knowledgeably to battlefield injuries. *MSTC U.S. Army*. (n.d.).
10. Bandura, A.: Social Cognitive Theory: An Agentic Perspective. Annual Review of Psychology, 52: 1-26. (2011).
11. Vygotsky, L.: Zone of Proximal Development. Mind in Society: The Development of Higher Psychological Processes: 52-91. (1987).

Authors

Benjamin Goldberg: is a member of the Learning in Intelligent Tutoring Environments (LITE) Lab at the U.S. Army Research Laboratory's (ARL) Simulation and Training Technology Center (STTC) in Orlando, FL. He has been conducting research in the Modeling and Simulation community for the past five years with a focus on adaptive learning and how to leverage Artificial Intelligence tools and methods for adaptive computer-based instruction. Currently, he is the LITE Lab's lead scientist on instructional strategy research within adaptive training environments. Mr. Goldberg is a Ph.D. Candidate at the University of Central Florida and holds an M.S. in Modeling & Simulation. Prior to employment with ARL, he held a Graduate Research Assistant position for two years in the Applied Cognition and Training in Immersive Virtual Environments (ACTIVE) Lab at the Institute for Simulation and Training. Mr. Goldberg's work has been published across several well-known conferences, with recent contributions to both the Human Factors and Ergonomics Society (HFES) and Intelligent Tutoring Systems (ITS) proceedings.

Jan Cannon-Bowers: is Director of Research at the Center for Advanced Medical Learning and Simulation (CAMLS) at the University of South Florida (USF) in Tampa, FL. She holds MA and Ph.D. degrees in Industrial/ Organizational Psychology from USF. She served as Assistant Director of Simulation-Based Surgical Education for the Department of Education at the American College of Surgeons from 2009 - 2011. Previously, Dr. Cannon-Bowers served as the U.S. Navy's Senior Scientist for Training Systems where she was involved in a number of large-scale R&D projects directed toward improving performance in complex environments. In this capacity she earned a reputation as an international leader in the area of simulation and game-based training, team training, training evaluation, human systems integration and applying the science of learning to real-world problems. Since joining academia, Dr. Cannon-Bowers has continued her work in technology-enabled learning and synthetic learning environments by applying principles from the science of learning to the design of instructional systems in education, healthcare, the military, and other high performance environments. She has been an active researcher, with over 150 publications in scholarly journals, books and technical reports, and numerous professional presentations.

Bringing Authoring Tools for Intelligent Tutoring Systems and Serious Games Closer Together: Integrating GIFT with the Unity Game Engine

Colin Ray, Stephen Gilbert

Iowa State University, Ames, IA, USA
{rcray, gilbert}@iastate.edu
http://www.iastate.edu

Abstract. In an effort to bring intelligent tutoring system (ITS) authoring tools closer to content authoring tools, the authors are working to integrate GIFT with the Unity game engine and editor. The paper begins by describing challenges faced by modern intelligent tutors and the motivation behind the integration effort, with special consideration given to how this work will better meet the needs of future serious games. The next three sections expand on these major hurdles more thoroughly, followed by proposed design enhancements that would allow GIFT to overcome these issues. Finally, an overview is given of the authors' cur- rent progress towards implementing the proposed design. The key contribution of this work is an abstraction of the interface between intelligent tutoring systems and serious games, thus enabling ITS authors to implement more complex training behaviors.

Keywords: intelligent tutoring, serious games, virtual environments, game engines

1 Introduction

Experience with the Generalized Intelligent Framework for Tutoring (GIFT) has shown that authoring new courses, domain knowledge, and learner configurations requires little-to-no programming experience. A basic understanding of XML and how the modules of GIFT interact is sufficient to design and configure a course for one of the supported training applications. When it comes to extending the framework to support new training applications, however, each interface module must be handcrafted. Reducing the amount of effort required to author a tutor and its content is a desirable quality of future authoring tools [1], therefore the task of integrating new training applications should be made as seamless as possible.

Serious games are one example of training applications that are well-suited for integration with ITSs; two such games are already supported by GIFT: Virtual Battlespace 2 (VBS2) and TC3 vMedic. These games encompass a only a subset of the training material that is possible with serious games, however. There are certain aspects of this genre of game are common across all individual applications, meaning that it may be possible to create a single abstraction layer capable of decoupling GIFT from the training application. This approach is recommended by Sottilare and Gilbert, who suggest that such an abstraction layer might be able to translate learning objectives into meaningful objects actions in the game world, and vice versa [2].

In addition to adapting data about the game state to a format that the ITS expects, it is also desirable for the ITS to have a finer degree of control over the scenario itself. These so-called "branching" or "conditional" scenarios [2] are difficult to achieve if the serious game and its plugin API are not designed with such functionality in mind. Therefore, it may also be necessary to "standardize" the ability to branch scenarios in the design of serious games.

To these ends, our proposed solution is to bring the ITS authoring tools closer to the content authoring tools used to create a given serious game. In the case of this paper, we have chosen to work with the popular Unity game engine. In the following sections we will show how integration with Unity and other serious game authoring tools can achieve the functionality that is currently desired in a modern ITS authoring suite.

2 Current Authoring Capabilities

As stated by Sottilare et. al, authoring new intelligent tutors is one of the three primary functions of GIFT [3]. To this end, the framework already contains authoring tools that enable users to create and configure the essential elements of an intelligent tutoring program. The following list gives a brief overview of the current authoring capabilities supported by GIFT:

- Authoring learner models through the Learner Configuration Authoring Tool (LCAT)
- Configuring sensors through the Sensor Configuration Authoring Tool (SCAT)
- Authoring Domain Knowledge Files (DKFs) through the DKF Authoring Tool (DAT)
- Creating and presenting surveys through the Survey Authoring Tool (SAT)

By using good design principles, the authors of GIFT have been able to effectively decouple the authoring of individual tutor components from one another. The decoupling of different program elements is important for improving the maintainability and extensibility of large pieces of software such as GIFT. One area of the framework design that suffers from tight coupling is the integration of third-party training applications, e.g. VBS2, vMedic, etc.

The development of these authoring tools is guided by several design goals, one of which is to "Employ standards to support rapid integration of external training/tutoring environments." [3] In this regard, the current GIFT authoring construct can benefit from design enhancements that standardize this process across a range of training applications. Through the work outlined in this paper, we aim to generalize the process of integrating serious games with GIFT by creating an abstraction layer between GIFT and the game engine itself.

3 Related Work

Prior work in integrating serious games and intelligent tutors has demonstrated that ITS authoring tools can be easily adapted to work with individual games. Research conducted by Gilbert et al. demonstrated interoperation between the Extensible Problem-Specific Tutor (xPST) and a scenario created in the Torque game engine [4].

Devasani et al. built upon this work and demonstrated how an authoring tool for interpreting game state and player actions might be designed [5]. For their work, xPST was integrated with a VBS2 scenario. An important revelation made by the authors was that the author of the tutor need not define a complete state machine with transitions, since these transitions are implicit when the game engine changes state each frame.

Another of the GIFT design goals is to "Develop interfaces/gateways to widely used commercial and academic tools." [2] As previously mentioned, the current GIFT release has support for two serious games, one of which is VBS2, and the other being vMedic. This work and the previous two examples highlight the usefulness of integrating intelligent tutors with serious games, as well as the need for a standardized interface for authoring relationships between the game objects and tutor concepts. There are currently no concrete examples of a standard for quickly integrating serious games and intelligent tutors, although Sottilare and Gilbert make recommendations on how this problem might be approached [2].

4 Design Enhancements

As noted by previous authors [2, 4], one of the key challenges of tutoring in a virtual environment is mapping relevant game states to subgoals defined by the training curriculum. If the learner's goal is to move to a specific location, for example, the tutor author may not be interested in how the learner reached that state (e.g., driving, walking, or running). Thus, the tutor would have to know to filter out information from the game engine about modality of movement, and attend only to the location. If, however, the trainer wants to focus on exactly how best to move to that location (e.g., stealthily), then the tutor does need to monitor movement information. Using this example, we see that the context of the pedagogical goal influences the type of and granularity of tutor monitoring. From here on, we will refer to this challenge as the "observation granularity challenge."

In the process of reaching each pedagogical goal, the learner will build up a history of actions. Similar to the concept of a context attached to goals, there can also be context attached to patterns of actions over time. As an example, there may be cases where a tutor would permit errors in subgoals within a larger pattern of actions that it would still deem "successful." This history is essentially a recording of the virtual environment state over the course of the training. The amount and diversity of data in this history stream is potentially massive, creating a major challenge when attempting to recognize patterns. The problem of recognizing these patterns is crucial for identifying the learner's progress. From here on, we will refer to this challenge as the "history challenge."

In addition, because game environments afford interaction among multiple simultaneous entities, the tutor's reaction to actions and other new game states may be dependent on the actor. This context dependence suggests that it would be a valuable to add game entity attributes to state updates, and for GIFT to be able to process logic such as, "If the gunshot action came from an entity that is unknown or hostile, then take action X. If the gunshot came from a friend entity, take action Y." The additional layer of entity attributes adds complexity to authoring, but will be necessary for modeling team and social interactions. Devasani et al. describes a possible state-based architecture that could be the basis for such an approach, and it could be incorporated into GIFT [4]. From here on, we will refer to this challenge as the "actor-context challenge."

4.1 Abstraction Layer

A core aspect of the design principles behind GIFT is its generalizability to new training applications and scenarios. For this reason it is critical that the representations of data in GIFT and in the training application be allowed to remain independent. It is infeasible to force training applications to adapt to the interfaces that GIFT provides. However, a layer of abstraction that adapts messages from a sender into a form that can be consumed by a receiver is similar to the classic Adapter design pattern in software engineering. This design pattern has the useful property of enabling two otherwise incompatible interfaces to communicate, in addition to decreasing the coupling between them. In the case of GIFT, the abstraction layer would handle the mapping of objects from one service into a representation that makes sense to the other. As an example, this module might receive a message from the game engine containing the new location of the learner in the virtual environment which might then be interpreted for the tutor as "the learner reached the checkpoint."

In addition to mapping game engine concepts to tutor concepts, the abstraction layer can act as a filter in order to solve the observation granularity and history challenges. The scripting language achieves this by affording "do not care" conditions that would then trigger the abstraction layer to interpret only the relevant messages and discard everything else.

One proposed method for implementing this mapping is a scripting language and engine that allows the author to define the mapping themselves. Although it is far from being an automated solution, a scripting language would allow the ITS and content authors to hook into more complex behaviors with very little learning overhead. Scripting languages can be more user-friendly than XML by virtue of their syntactical similarities to written English. Furthermore, within the context of the Unity development environment we can expect users to have familiarity with scripting languages such as JavaScript and Boo (similar to Python). For these reasons, a scripting language is a natural choice for abstracting communication between GIFT and Unity. It is important for the simplicity of tutor authoring that this messaging abstraction layer have the tutor-side representation use language that a trainer would naturally use. If this is the case, the trainer can more easily author feedback and adaptive scenarios.

Although JavaScript and Boo are well-suited as tools to implement complex behaviors for game objects, they overcomplicate the task of describing interactions between the game world and the tutor. Instead of complex behaviors, we seek to enable the tutor author to quickly declare relationships between objects in the game, domain knowledge, and pedagogical goals.

In order to avoid burdening the author with the challenge of authoring different components in different languages, it may be advantageous to use XML for authoring abstraction layer rules. The declarative nature of XML makes it ideal for this role, although as mentioned previously, it suffers from poor readability. An alternative to XML is TutorScript, a scripting language developed for use in ITSs [6]. The design of TutorScript centers around the sequences of goals or contexts called a predicate tree. TutorScript's primary advantage over the previously mentioned alternatives is that it was designed with the goal of relating domain knowledge to learner actions in the training application. Additionally, TutorScript takes inspiration from Apple script in regards to syntax, allowing non-programmers to write scripts that read like English. For our work, TutorScript would allow us to hook into objects in both GIFT and Unity, where we can then create interactions using simple language.

4.2 Unity Editor

One of the main benefits of the Unity editor is that it is extensible to support user-created tools for custom workflows, or to fill in functionality lacked by the default editor. Some examples of editor plugins authored by users have added advanced level building tools, cut-scene editors, and even node-based visual scripting interfaces. The ultimate goal of this project is to completely integrate GIFT's authoring tools with the Unity ecosystem. This entails creating editor plugins for the entire suite of GIFT authoring tools, thereby enabling content authors to generate serious game and tutor content side-by-side using a single development environment.

An added benefit of integration with the Unity editor is its powerful rapid-prototyping abilities. Scenarios in Unity are organized into "Scenes" which can be loaded individually, played, and paused within Unity's built-in player. Current work to develop a proof-of-concept has demonstrated that it is possible to interact with the tutor within this player, thereby enabling the author to perform debugging on the training scenario to an extent.

It is considered good practice when authoring Unity games to "tag" game objects with names that encode the meaningful behavior that the game object performs. Assuming that the author adheres to this practice, the tagging mechanism combined with the abstraction layer will solve the actor-context challenge. Tags can be transmitted with game state updates that pass through the abstraction layer, which will then interpret the tags into context that is meaningful to the tutor. Since the abstraction layer is scripted by the author, it is essential for the abstraction layer script editor to be included in Unity's authoring suite. Making these tools easily accessible from one or the other allows the author to update changes to the scripts as soon as he or she makes changes to game object tags and other metadata.

As stated previously, the scripting languages provided by Unity may not be ideally suited to the task of communicating between the game engine and the tutor. Additional modifications will need to be made to MonoDevelop, the highly extensible IDE distributed as part of Unity, in order to support TutorScript or a variant of it. MonoDevelop greatly simplifies the creation of helpful programming tools such as syntax-highlighting and auto-completion that assist users with no prior programming background. Developing a MonoDevelop add-on for TutorScript also allows the author to more easily manage large or complex scripts needed to address the history and actor-context challenges via the built-in code organization features such as collapsing scopes. Taken together, Unity and MonoDevelop can be used as a suite of tools for authoring not only serious game content, but also advanced tutor behaviors, curriculum, and domain knowledge that will drive the training scenarios.

5 Recommendations

We project that the design enhancements recommended in this paper will assist in improving time savings and reducing cost involved with authoring an intelligent tutor. Specifically, we are aiming to reduce the time required to integrate GIFT with a new serious game by instead integrating it with the game engine itself. Our reasoning is that there are relatively few game engines that would need to be integrated, compared to the number of games with potential for enhancement through tutoring. Additionally, code reuse is facilitated by employing a standard format for describing relationships between game and tutor objects. If successful, this work will introduce a new abstraction layer between GIFT and the game engines that drive serious serious games, enabling a single tutor configuration to be deployed across a wide range of scenarios. For your convenience, the recommendations have been consolidated and figured in the table below.

Table 3. GIFT Design Enhancement Recommendations

Improve decoupling of potential learner actions and other game-specific data from the gateway and other GIFT modules.
Define a new XML schema for constructing game-tutor object relationships.
Develop a new authoring tool capable of authoring and validating these relationships.
Integrate new and existing authoring tools with the Unity editor.

6 Current Work

At this point we have successfully developed a proof-of-concept plugin that demonstrates basic communication between GIFT and Unity-driven games, similar to the interoperation module developed for VBS2. The extent of this functionality encompasses connecting to the Unity plugin from GIFT and then issuing playback commands such as pause and resume to the Unity player. This work has helped to increase our understanding of the inner workings of GIFT with regard to the augmentation required to communicate with our abstraction layer. In particular, the extent to which GIFT is tailored to each training application became apparent. In addition, we were able to leverage support for C# .NET 2.0 in Unity to move a great deal of the supporting code into components attached to game objects. This design allows the three services (Unity, Abstraction Layer, and GIFT) to remain isolated from one another during development, encouraging loose coupling across service boundaries and portability to other serious game authoring tools.

Before any work on the abstraction layer can begin, the language used to define object relationships must first be well-defined. Once this step is completed, we can begin abstracting away the elements of third-party application integration in GIFT that are currently hard-coded. Ultimately, these elements will be encapsulated by the proposed abstraction layer.

7 Conclusion

In this paper we proposed a handful of major design enhancements to GIFT with the overarching goal of bringing the ITS authoring workflow into the game content creation pipeline. The first task in realizing this vision is to create an abstraction layer comprised of a scripting engine tailored for ITSs. The second and final task is to integrate the GIFT authoring tools into Unity, in order to encourage side-by-side development of game and tutor content. The Unity game engine has been chosen for this work due to its ease of use, cross-platform support, and high extensibility. It is our hope that such a comprehensive suite of tools will help to drive a new generation of high-quality serious games.

8 References

1. Brawner, K., Holden, H., Goldberg, B., Sottilare, R.: Recommendations for Modern Tools to Author Tutoring Systems. (2012)
2. Sottilare, R.A., Gilbert, S.: Considerations for adaptive tutoring within serious games: authoring cognitive models and game interfaces. (2011)
3. Sottilare, R.A., Brawner, K.W., Goldberg, B.S., Holden, H.K.,: The Generalized Intelligent Framework for Tutoring (GIFT). Technical report (2013)
4. Gilbert, S., Devasani, S., Kodavali, S., Blessing, S.: Easy Authoring of Intelligent Tutoring Systems for Synthetic Environments. (2011)
5. Devasani, S., Gilbert, S. B., Shetty, S., Ramaswamy, N., Blessing, S.: Authoring Intelligent Tutoring Systems for 3D Game Environments. (2011)
6. Blessing, S.B., Gilbert, S., Ritter, S.: Developing an authoring system for cognitive models within commercial-quality ITSs. (2006) 497–502

Authors:

Colin Ray is a graduate student at Iowa State University, where he is pursuing an M.S. in Human Computer Interaction and Computer Science under the guidance of Stephen Gilbert, Ph.D. He possesses a B.S., also from Iowa State University, in the field of Electrical Engineering. His current research is focused on integrating intelligent tutoring systems with entertainment technology. In addition to ITS research, he is also conducting research and development in the areas of wireless video streaming and mobile surveillance to develop a platform for studying 360-degree video interfaces.

Stephen Gilbert, Ph. D., is the associate director of the virtual reality applications center (VRAC) and human computer interaction (HCI) graduate program at Iowa State University. He is also assistant professor of industrial and manufacturing systems engineering in the human factors division. His research focuses on intelligent tutoring systems. While he has built tutors for engineering education and more traditional classroom environments, his particular interest is their use in whole-body real-world tasks such as training for soldiers and first responders or for machine maintenance. He has supported research integrating four virtual and three live environments in a simultaneous capability demonstration for the Air Force Office of Scientific Research. He is currently leading an effort to develop a next-generation mixed-reality virtual and constructive training environment for the U.S. Army. This environment will allow 20-minute reconfiguration of walls, building textures, and displays in a fully tracked environment to produce radically different scenarios for warfighter training. Dr. Gilbert has over 15 years of experience working with emerging technologies for training and education.

Authoring a Thermodynamics Cycle Tutor Using GIFT

Mostafa Amin-Naseri[1], Enruo Guo[2], Stephen Gilbert[1], John Jackman[1], Mathew Hagge[3], Gloria Starns[3], LeAnn Faidly[4]

[1] *Department of Industrial & Manufacturing Systems Engineering*
[2] *Department of Computer Science*
3 Department of Mechanical Engineering
Iowa State University, Ames, IA, 50011 USA
4 Department of Mechanical Engineering
Wartburg College, Waverly, IA 50677 USA
{aminnas, enruoguo, gilbert, jkj, fforty, gkstarns}@iastate.edu,
leann.faidley@wartburg.edu

Abstract. The main idea of generalized intelligent tutoring system (ITS) development tools like Generalized Intelligent Framework for Tutoring (GIFT) is to provide authors with high-level standards and a readily reusable structure within different domains. Hence, adapting such a tool could be the best way to boost an underdeveloped tutor. In this paper we propose the design for a new GIFT-based tutor for undergraduate thermodynamics. An existing Thermodynamics Cycle Tutor has been designed that is meant to facilitate problem framing for undergraduate students. We describe the advantages of integrating this tutor with GIFT to add student models. Also an approach for evaluating the pedagogical performance of the GIFT-enhanced tutor is described.

Keywords: GIFT, intelligent tutoring system, thermodynamics cycle

1 Introduction

One of the most important challenges for engineering students is problem solving. Complex engineering problems typically contain multiple constraints, require multiple ideas, and may not have clear criteria for deciding the best solution. Beginning students struggle with engineering problem solving, and it has been observed that the initial stage (i.e., framing the problem) often causes the most difficulty. Students find it difficult to frame a complex problem, identify the core components, and brainstorm a possible solution path. These difficulties triggered the idea of building a tutor that can help undergraduate engineering students with their problem framing.

Thermodynamics cycles were our choices of topic to start with. In a National Science Foundation (NSF) funded project, a web-based software was developed to give students the ability to draw some initial sketches of the problem. Their drawing will be evaluated with regard to the expert model provided by the instructor and respectively they will be provided with different types and categories of feedback and instructions.

Regardless of how much effort is devoted to a project, there is always room for improvement. Key advantages of a generalized approach to ITS development (and GIFT in particular) are their standards and their high potential for reuse across educational and training domains. Other advantages that drive efficiency and affordability are GIFT's modular design and standard messaging; its largely domain-independent components; and its reuse of interfaces, methods, and tools for authoring, instruction, and analysis. Given these GIFT characteristics, there are many ways that the tutor could be enhanced being incorporating into GIFT. This will also provide us with an invaluable testbed to examine a GIFT-enhanced tutor with the existing one.

In the following sections, first a brief description of the tutor will be given and then an overview of the ways that the existing tutor can be enhanced by GIFT will be demonstrated. Finally a testing opportunity for the software will be described.

2 Current tutor

We would like to describe our current intelligent thermodynamics cycle tutor for engineering undergraduate courses. For the purpose of conceptualization and design, an ITS is often thought as consisting of several interdependent components: domain model, learner model, expert model, pedagogical module, interface and training media (Beck, Stern & Haugsjaa, 1996; Sottilare & Gilbert, 2011; Sottilare & Proctor, 2012).

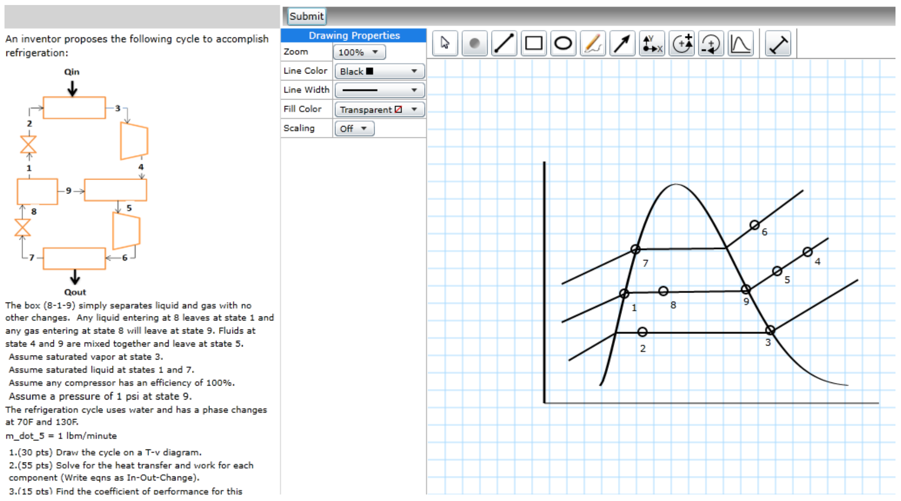
2.1 Domain model

The domain is about thermodynamics cycle problems. The goal is to understand how changes in pressure, temperature, specific volume and entropy interact with some commonly-used components, such as pump, compressor, turbine, expansion value, evaporator, heat exchanger, liquid-gas separator and mixing chamber. Based on the physical and chemical properties, a rule is associated with each component. For example, when an object goes through a pump, the pressure will increase, while the temperature and specific volume will increase slightly. In the final version, the author will have the option to modify the rules (e.g. to assume constant specific volume, or to test a student with a component that doesn't make physical sense). The table below shows the rules associated with other components. The domain model contains these rules.

Table 4. Rules for several components

Component	Pressure	Temperature	Specific Volume	Entropy
expansion valve	decease	decrease	Increase	Increase
evaporator	Same	same, increase	Increase	Increase
compressor	increase	Increase	decrease	same, increase
mixing chamber	same	between	between	between
condenser	same	decrease, same	decrease	decrease
Liquid -gas separator	same	same	between	between

2.2 Interface


An inventor proposes the following cycle to accomplish refrigeration:

The box (8-1-9) simply separates liquid and gas with no other changes. Any liquid entering at 8 leaves at state 1 and any gas entering at state 8 will leave at state 9. Fluids at state 4 and 9 are mixed together and leave at state 5.
Assume saturated vapor at state 3.
Assume saturated liquid at states 1 and 7.
Assume any compressor has an efficiency of 100%.
Assume a pressure of 1 psi at state 9.
The refrigeration cycle uses water and has a phase changes at 70F and 130F.
m_dot_5 = 1 lbm/minute
1.(30 pts) Draw the cycle on a T-v diagram.
2.(55 pts) Solve for the heat transfer and work for each component (Write eqns as In-Out-Change).
3.(15 pts) Find the coefficient of performance for this

Drawing Properties
Zoom: 100%
Line Color: Black ■
Line Width: ———
Fill Color: Transparent
Scaling: Off

Fig. 5. A screenshot of Thermodynamics Cycle Tutor. The student reads the problem at left and solves it by constructing a vapor dome diagram at right.

Thermodynamics Cycle Tutor has been developed as part of a problem framing research project funded by the National Science Foundation. The tutor basically contains two parts. On the left side, it contains system/component diagram, problem description and questions. The right side uses a web-based drawing interface, XDraw, developed internally by author Jackman using the Microsoft Silverlight framework. XDraw supports basic drawing objects such as vapor dome, point, line, rectangle and vector as well as freehand drawing. It also provides facilities to allow students to label the states and insert text on the drawing. A backend database saves students' diagrams. XDraw communicates with tutor server via a TCP socket. Several message types are defined in order to differentiate what information would be checked and the next action should be taken.

When it starts, the left side shows the system diagram and problem description. Students can start problem framing by drawing a vapor dome (T-V diagram in this case) and use lines and points to represent pressure curve and state, respectively, and apply labels according to the system diagram. After clicking submit button, the diagram is sent to the tutor server, which checks a specific part based on the query message. The tutor then sends back the evaluation result and instruction for the next action as a returned message. Students may be directed to another interface based on their performance in the current stage. We will talk about the detailed sequences in the expert model.

2.3 Expert model

The expert model sets standards and compares learner actions to determine the progress. In the thermodynamics cycle domain, the expert model contains the following:

1. Check vapor dome present.
2. Check number of pressures.
3. Check number of states.
4. Check Pressure and Temperature relations in each of the components.

After the student submits the drawing, the tutor will check if the drawing contains the vapor dome. If so, it will continue to the next check: number of pressures. If it is wrong, the students will be asked questions like, “How many pressures are there in the system?” showing on the left panel. If the student’s answer is wrong, the tutor will go through all the components, and ask the pressure change within each of them. Some tutorial videos and illustrations will be provided to help them better understand the concept.

The content on the left panel will be changed based on the student’s activity in a particular problem. For example, in the drawing, the student draws state 4 to the right of state 3. A compressor pushes the gas molecules closer together, so specific volume should decrease. The left panel will show a compressor’s diagram, along with some questions, such as “How does the specific volume change in a compressor?” It contains several choices that students can select. If the student chooses the correct answer, it will ask the student to correct it in the diagram. If the student gets the wrong answer, it will direct the student to some tutorial video files and ask again.

2.4 Training media

In order to help students correct their misconceptions, the tutor provides some video files that include class lectures and illustration videos at a certain stage of the activity. The video files will be loaded automatically to ask students to watch when their answer is wrong. Generally speaking, the training media is domain-dependent and requires the instructor to prepare and pre-define what stage it should appear.

2.5 Learner model

Currently there is no learner model in our Thermodynamics Cycle Tutor. We think it is a good idea to monitor and keep track of students' current progress, save students' previous performances, and perform surveys. An example could be when student starts a new problem, the tutor should be able to select an appropriate problem from the learner model and predict how successful the student will be based on his/her historical data. Also, in the survey part, the tutor could receive feedback on the learner's background knowledge and quality of the pedagogical process. We believe GIFT could allow us to build a learner model easily, and we would like to explore how it may benefit our tutor.

2.6 Pedagogy

As a pedagogical learning tool, the tutor also needs to set up learning goals and pace for the students, so the student can learn each component's P, T, and V behavior one at a time (starting with the easiest one, and increasing difficulty as easier ones are mastered). The ideal tutor would be able to connect with other thermodynamic understanding, using ideas such as rate of heat transfer and rate of work (power) to connect with P, T and V relationships. For students with different performance levels, the problem difficulty should vary. The tutor's feedback has to inspire their thinking, not give them answers directly. The pedagogy module requires much flexibility and should vary based on different problem sets and instructor-student needs.

3 GIFT-enhanced tutor

The existing tutor is expected to demonstrate an acceptable functionality; however, there are limitations in its domain independence and reusability, and it also lacks some desirable features such as a learner model. Mitigating these limitations will require a considerable amount of time and programming effort. GIFT offers many features that can attenuate the level of programming skill and time required. Also, providing standards and well-defined domain independent structures facilitates the tool enhancement. The main benefits of GIFT for our tutor are explained below.

3.1 Learner model

A highly desired feature for intelligent tutors is to provide learners with personalized education (Woolf, 2010). In other words, if we could know the exact skills that learners do and do not have, then we could provide them with the exact resources they need. Learner model is a module that has been developed for this reason. Learner model keeps record of many aspects of the learner, such as the learner's progress toward objectives, actions taken in the interface and historical data (e.g., previous performance) (Sottilare, 2012). There is also a need to define some skill levels with respect to the learner's patterns. Having this valuable information about the learner and their skill, the tutor will be able to provide him or her with specific problems, feedback, instructional content, etc**.**

In our current tutor there are many data streams that are monitored (e.g., the mistakes or feedback types, instructional content provided, etc.). Also, by handing out surveys, some information about knowledge background is available. The problem is they are stored in separate databases and it is hard to put them together. Putting these data together can help us build the student model. GIFT provides the ability to store this data in a well-structured way, as it has the option for sensor data storing. In addition, we can benefit the GIFT survey authoring tool, to conduct our surveys in the same program and store them easily in the proper place. In this way, by defining the skill levels we will be able to build our learner models based on the data we have collected from them.

Another important feature is data reporting. Having collected a considerable amount of data on the learner, an easy-to-access way to extract knowledge out of it is necessary. The GIFT event report tool provides a proper interface to easily let users (instructors) access the data they desire.

For any further research, we might want to use different types of sensors to evaluate a learner's cognitive load or status or stress. GIFT provides the ability to readily acquire that data as well.

3.2 Multiple scenarios

Once skill levels have been assigned to learners, appropriate content must be provided based on those skills. To handle various types of problems and instructional material (i.e., Domain Knowledge Files), a precise structure is needed to store them. For this, GIFT Domain Authoring Tool (DAT) will be used. Since this tool can be used without having to launch the entire GIFT architecture, it enables us to benefit from it earlier in the development process.

In addition, different instructors have different pedagogical strategies and instructional content. Thus, they may want students to go through a different scenario or visit different content. Two of our co-authors, for example, have different pedagogical preferences for teaching the thermodynamic cycle. Based on their preferences, GIFT could enable us to create multiple scenarios appropriately. Without having a perfect match between the knowledge database and the tutor, accommodating multiple approaches would not be possible. However, GIFT has already provided the structured database, so making the linkage between the tutor and GIFT DAT will be helpful.

3.3 Expansion to other domains

The domain-independent structure of GIFT will enable us to simply customize our tool for different fields. Currently, statics problems, e.g., free body diagrams, can also be tutored via our tool, but using the Domain Authoring Tool that will facilitate the deployment of instructional material. The entire process of student model and learner-specific instructions could be implemented with this approach as well.

4 Proposed evaluation experiment

In the Fall 2013 semester, a thermodynamics class will be offered for undergraduate mechanical engineering students in Iowa State University. Early in the semester they will be divided into three groups. One group will work with the GIFT-enhanced tutor, another group with the existing tutor (without student model), and the last group will just join the class and have no tutor. Keeping records of the three groups' performances during the semester with periodic quizzes, as well as gathering data on their skills and solution time, will provide us with a valuable data to evaluate the performance of an intelligent tutor with the student model (GIFT-enhanced). It will also help us examine the effectiveness of the existing tutor.

5 Conclusion

After analyzing the features of our existing tutor and GIFT, they seem to complement each other perfectly and provide a comprehensive ITS. Using GIFT's standards for structuring the tutor, as well as data and file storing, will attenuate the requisite programming skill and effort to accomplish the same objectives. Also, its high domain-independence will create opportunities to expand the tutor to different learning domains. The GIFT-enhanced tutor will be compared with the existing tutor and with traditional class training during the 2013 Fall semester. The results could provide a documented comparison between two different methods of ITS development.

6 References

1. Beck, J., Stern, M., and Haugsjaa, E. (1996). Applications of AI in Education, ACM Crossroads.
2. Woolf, B. P. (2010). A Roadmap for Education Technology. National Science Foundation.
3. Sottilare, R. (2012), Adaptive Tutoring & the Generalized Intelligent Framework for Tutoring (GIFT), Retrieved from: http://www.ist.ucf.edu/grad/forms/2012_10_Sottilare_UCF_GIFT%20brief.pdf.
4. 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, Christchurch, New Zealand, June 2011.
5. Sottilare, R. and Proctor, M. (2012). Passively classifying student mood and performance within intelligent tutoring systems (ITS). Educational Technology Journal & Society. Volume 15, Issue 2.

Authors

Mostafa Amin-Naseri: is a Master of Science student in Industrial and Manufacturing Systems Engineering in Iowa State University. His BS was in industrial engineering with a major in systems engineering. Having had an experience in tutoring high school and undergraduate students he got familiar with common mistakes and issues that students usually face when solving problems and also the necessity for personalized instructional material. This background led him to start working on Intelligent Tutoring Systems (ITS). He is currently working with a team on an ITS that helps undergraduate engineering students with problem framing in Statics and Thermodynamics. Applying statistical analysis and data mining techniques to learners' historical data in order to come up with learner models, evaluate skill levels and to offer customized instructional material and feedback, is one of his fields of interest. Finally, with a systems engineering background, he is also interested in analyzing and simulating the learning process using System Dynamics models.

Enruo Guo: a Ph.D. student in Computer Science co-majoring in Human Computer Interaction in Iowa State University. She got her Master's degree in Computer Science from Iowa State in 2012. She also had a background in pedagogy and psychology in her undergraduate study in Beijing Normal University, which is best-known for training school teachers in China. Her philosophy is to use computers to simplify human's learning process and make everything as simple as possible. She has broad interests in intelligent tutoring system, artificial intelligence, computer vision and virtual reality. She has strong enthusiasm in developing real-world applications to assist undergraduate teaching and administration. She develops Thermodynamics Cycle Tutor and now is working on Free-Body Diagram Tutor for engineering undergraduates. Furthermore, in order to reduce the workload of human inspector of Department of Chemistry, she develops Intelligent Safety Goggle Detector which can automatically detect if a lab user wears safety goggle at the entrance of the lab.

Stephen Gilbert, Ph.D., is the associate director of the virtual reality applications center (VRAC) and human computer interaction (HCI) graduate program at Iowa State University. He is also assistant professor of industrial and manufacturing systems engineering in the human factors division. His research focuses on intelligent tutoring systems. While he has built tutors for engineering education and more traditional classroom environments, his particular interest is their use in whole-body real-world tasks such as training for soldiers and first responders or for machine maintenance. He has supported research integrating four virtual and three live environments in a simultaneous capability demonstration for the Air Force Office of Scientific Research. He is currently leading an effort to develop a next-generation mixed-reality virtual and constructive training environment for the U.S. Army. This environment will allow 20-minute reconfiguration of walls, building textures, and displays in a fully tracked environment to produce radically different scenarios for warfighter training. Dr. Gilbert has over 15 years experience working with emerging technologies for training and education

Dr. John Jackman: an Associate Professor, Industrial and Manufacturing Systems Engineering at Iowa State University, conducts research in manufacturing and engineering education. In manufacturing, he is currently working on wind turbine blade inspection techniques that characterize the variability in blade geometry and detect surface flaws. Dr. Jackman has extensive experience in computer simulation, web-based immersive learning environments, and data acquisition and control. His work in engineering problem solving has appeared in the Journal of Engineering Education and the International Journal of Engineering Education. He is currently investigating how to improve students' problem framing skills using formative assessment.

Dr.Mathew Hagge: has built a teaching style for thermodynamics that simplifies the course into a small set of ideas and skills, and asks students to develop and apply these same ideas to novel situations. The same set of ideas and skills are used for every problem. No equation sheets or similar problems are used. Memorization is not needed, and will actually decrease student performance. Students are asked to make as many decisions as possible, subject to their level of understanding. As student knowledge and expertise increases, so does the problem complexity. Less than a dozen problems will be needed, but each new problem will push the student's understanding. By the end of the course, successful students have the skills to solve any problem in a traditional textbook, and to correctly solve problems much more complex than a traditional textbook. When students need help, Dr. Hagge has developed a set of questions that can identify the specific misunderstanding, and then provide an activity or discussion that will eliminate the misunderstanding.
Dr Hagge's teaching method is ideally suited for implementation with a tutor that focuses on student understanding. The tutor can measure specific skills/understanding and provide feedback unique to that student.

Dr Gloria Starns: received her Ph.D. in Mechanical Engineering from Iowa State University in 1996 and began instructing engineering students as a graduate student at Iowa State in 1990. Dr. Starns' interest in working with a personal tutoring system is related to her past work with concept based learning, as well as understanding the role that use of active and constructive learning has in enabling students to retain and use acquired knowledge; her role in this project has been to provide the research team problems of varying complexity for purposes of collecting data from the tutor as it continues to evolve.

Dr.LeAnn Faidley: is an Assistant Professor of Engineering Science at Wartburg College in Waverly, IA. She has a BS in Engineering Science and Physics from Iowa State University, an MS in Engineering Mechanics, and a MS and PhD in Mechanical Engineering from The Ohio State University. At Wartburg, Dr. Faidley teaches the freshman labs, the Engineering Mechanics sequence, the Design sequence, and Engineering Materials. She is interested in improving student engagement with engineering subjects through active learning, relevant projects, and interactive online tools.

Integrating GIFT and AutoTutor with Sharable Knowledge Objects (SKO)

Benjamin D. Nye

Institute for Intelligent Systems
University of Memphis, Memphis, TN 38111
benjamin.nye@gmail.com

Abstract. AutoTutor and the Generalized Intelligent Framework for Tutoring (GIFT) are two separate projects that have independently recognized the need for greater interoperability and modularity between intelligent tutoring systems. To this end, both projects are moving toward service-oriented delivery of tutoring. A project is currently underway to integrate AutoTutor and GIFT. This paper describes the Sharable Knowledge Object (SKO) framework, a service-oriented, publish and subscribe architecture for natural language tutoring. First, the rationale for breaking an established tutoring system into separate services is discussed. Secondly, a short history of AutoTutor's software design is reviewed. Next, the design principles of the new SKO framework for tutoring are described. Finally, the plans and progress for integration with the GIFT architecture are presented.

Keywords: Intelligent Tutoring Systems, Service Oriented Architectures, Message Passing, Design Patterns, Systems Integration

1 Introduction

Intelligent tutoring systems (ITS), despite effectiveness as instructional technology, have historically suffered from monolithic design patterns (Murray, 2003). Roschelle and Kaput (1996) referred to tutoring systems as "application islands" for their lack of interoperability. A recent systematic literature review by the author of this paper found little evidence of newer tutoring systems sharing components or working toward a common base of components (Nye, 2013). This lack of modular ITS services reduces the availability of ITS software by preventing sharing of ITS components between systems. This problem increases the cost of ITS development and imposes a high barrier to entry for new systems.

An improvement over this design would be a component-based and service-oriented architecture, allowing composability of ITS components. Composability would greatly benefit ITS research, due to the high interdisciplinary skill-set needed to build a full tutoring system. Service oriented design would allow specialists to focus on individual components, while sharing common components. It would also greatly reduce the waste of reimplementing components that could be shared by ITS. However, this concept is not new. Roschelle and Kaput (1996) suggested component-based design over a decade ago, but little meaningful progress has been made toward that end. Part of the problem was the relative novelty of tutoring systems: fewer established examples existed and there was less consensus about the definition and functionality of an ITS.

More recently, central researchers have noted that different ITS tools share many of the same high-level behaviors (VanLehn, 2006; Woolf, 2009). This consensus implies a common ontology for describing the high level functions of ITS components and the meaning of information passed between them. While literature consensus does not constitute a formal ontology, it indicates the possibility of a grammar for talking about the types of information communicated between different parts of an ITS. An argument against the feasibility of this approach might be the disconnected nature of many subfields of ITS research, which come from different theoretical backgrounds that are not easily integrated (Pavlik and Toth, 2010). With that said, regardless of the underlying theory, the external behaviors (e.g., giving a hint) and core assessments (e.g., learning gains) are quite similar. The need to maintain theoretical coherence does not mean that a common ontology is infeasible, but simply indicates that there are limits to its useful granularity. For example, does a user-interface care how a hint is generated? If not, the user interface should be able to display hints from any system capable of generating hints. By taking advantage of the distinct roles and functions within a tutoring system, breaking down an individual tutoring system into distinct, sharable components is possible. Moreover, a significant number of components of the tutoring system are secondary to the tutor's theoretical concerns but pivotal to their operation. Machine learning algorithms, data storage interfaces, facial recognition software, speech synthesis, linguistic analysis, graphical interfaces, and tutoring API hooks for 3D worlds are enabling technologies for tutoring systems (Pavlik et al., 2012; Nye et al., 2013).

AutoTutor and the Generalized Intelligent Framework for Tutoring (GIFT) are two separate tutoring frameworks that have independently recognized the importance of modularity and interoperability in tutoring design. AutoTutor is a highly-effective natural language tutoring system where learners talk through domain concepts with an animated agent (Graesser et al., 2004a). Learning gains for AutoTutor average 0.8σ over reading static text materials on the same topic (Graesser et al., 2012). GIFT is a service-oriented framework for integrating tutoring capabilities into static material, such as a PowerPoint, and interactive environments, such as a simulation or a serious game (Sottilare et al., 2012). This paper describes the process of moving AutoTutor toward a service-oriented paradigm and the progress toward integrating AutoTutor with GIFT.

2 Prior AutoTutor Design Patterns

The original AutoTutor design was implemented as a standalone desktop application to teach computer literacy, which also relied on platform-dependent elements such as the Microsoft Agent (Peter Wiemer-Hastings et al., 1998). Since an installed application made AutoTutor harder to deliver, a subsequent version reimplemented the tutoring system as a web-based application (Graesser et al., 2004a). Since that time, various tutoring systems that followed in AutoTutor's footsteps have used a mixture of desktop and web-based designs. While many of these systems share conceptual principles and some share authoring tools, reuse of components and services between these different tutoring projects has been limited. So then, while Roschelle and Kaput (1996) spoke of "application islands," AutoTutor and related systems have evolved as a sort of "application archipelago" of related but independent tutoring systems. While the principles of AutoTutor have been influential, code reuse has been limited, even in projects that explicitly extend AutoTutor, such as AutoTutor Lite (Hu et al., 2009).

AutoTutor's package that handles linguistic analysis is a counter-example to this pattern. Coh-Metrix provides a suite of linguistic analysis tools, such as latent semantic analysis, regular expression matching, and domain corpora (Graesser et al., 2004b). While this tool started development nearly a decade ago, it remains under active development and is used regularly by AutoTutor and other projects. This longevity may be attributed to its focused scope and purpose as a toolbox for linguistic analysis. Additionally, Coh-Metrix has the advantage that it is primarily algorithmic and algorithms do not tend to change much.

By comparison, the landscape of educational computing has changed greatly over that period: web-based applications replaced many desktop applications, then full-featured Java web applications were replaced by lighter JavaScript and Flash clients with server-side code written in languages such as Python and C#. AutoTutor designs have mirrored these trends fairly closely, with the original AutoTutor written as a desktop application (Peter Wiemer-Hastings et al., 1998), the next iteration being a Java-based web application (Graesser et al., 2004a), and systems such as AutoTutor Lite relying on Flash, JavaScript, and Python (Hu et al., 2009). In the process of changing platforms and programming languages, a great deal of development work has been lost to a cycle of re-implementation to match the needs of a changing technology landscape.

Based on this history, how could design patterns be improved to encourage reuse and interoperability? The first principle, demonstrated by Coh-Metrix, is embodied by the Unix philosophy: "Do one thing and do it well" (Raymond, 2003). This is fundamental to service-oriented design, where boundaries between components are strict. The second principle is that delivery platforms may evolve rapidly. Just as AutoTutor has adapted to web delivery for desktops, mobile applications are becoming an important platform. Tutoring systems need to minimize platform-dependence. Finally, the best programming languages for different platforms vary. Moreover, existing tutoring systems have large investments in their code base. Components need to communicate using language-agnostic standards for different tutoring systems to interoperate. Service-oriented designs, while not yet common in tutoring systems, offer significant advantages for the next generation of ITS.

3 Sharable Knowledge Objects

AutoTutor is moving in this direction with Sharable Knowledge Objects (SKO), which allow creating tutoring modules by composing a mixture of components: local static media, remote static media, local components, and web services. These components are categorized in terms of two questions: 1. Is the component local? and 2. Is the component static or interactive? While the current focus of this work is on service-oriented web delivery, the design is also intended to support communication between components in the same process. By using a uniform messaging pattern, components can be developed without consideration of whether they will be used on a local device or accessed as a remote service.

In design pattern terms, SKO's are being developed to follow the service composition principle. In service composition, a composition of multiple services can be considered a single service when creating a new composition of services. Service-oriented design is largely the same concept as component-based design, except with the added complexity that the components may be distributed across time and space as part of a distributed network. So then, what is a SKO? A SKO declares a composition of services intended to deliver knowledge to a user, with the expected use case being tutoring in natural language. In this context, the SKO framework is not a re-implementation of AutoTutor but a framework for breaking AutoTutor down into minimal components that can be composed to create tutoring modules that may or may not rely on the traditional AutoTutor modules. These minimal components are intended to be used as part of a service-oriented design.

Figure 1 shows an overview of the new SKO framework. The core of the new SKO framework relies on a publish-and-subscribe architecture based entirely on passing messages that convey semantically-tagged information. These patterns significantly improve the flexibility of service composition for tutoring. Publish and subscribe frees individual components from explicit knowledge of any other services. The component knows only its own state, the messages that it has received, and the messages that it has transmitted. SKO is viewed as a way to split AutoTutor into separate, easily-reusable components. Secondly, SKO is intended to unify components from different systems that have evolved from AutoTutor along divergent paths by adding their unique functionality as services.

Exploring the details of each of these services is outside the scope of this paper. Instead, this section will focus on how different users would interact with and benefit from a SKO. While certain features of SKO are still under development, these examples describe how different users will interact with the completed SKO framework. To the learner, a SKO acts as a single module of instructional content focusing on a single lesson (e.g., learning how to complete a given math problem). For AutoTutor Lite, a web page loads a talking head and a user-input box, often with a button to begin a tutoring session. The SKO module does not specify any rules or functions. Instead, it relies on components to send messages. So then, user input triggers on the tutoring button generates a message from the user interface component. The tutoring engine reads that message and selects tutoring dialog, which is sent off as a new message. The animated agent and text-to-speech services read this message and cause the talking head to speak the message to the learner. By sharing a student model in a learning management system, multiple SKO can be combined into larger lesson units.

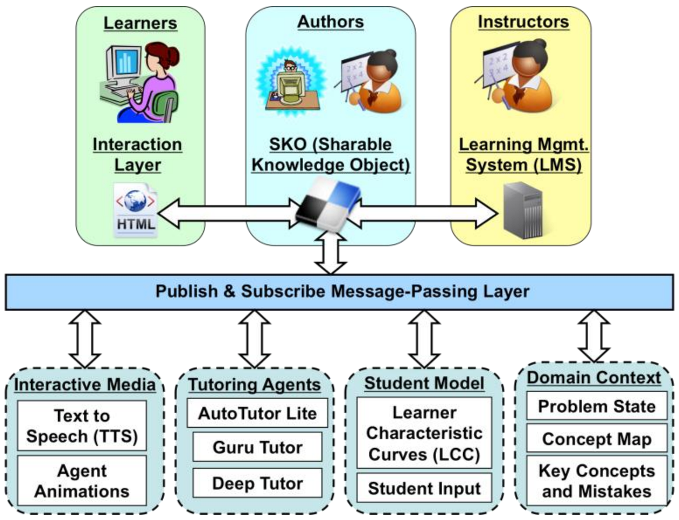


Fig. 6. Sharable Knowledge Object Framework for AutoTutor

To an advanced developer, a SKO is a collection of services. Advanced developers design new services and create SKO templates that can be filled in by instructors. These designers can create a SKO template using an advanced interface, where they would define the set of services within a SKO template and how these tie into the user interface. However, the advanced developer is not expected to add any domain content. Instead, they merely specify placeholders for content that is required or allowed. Based on these placeholders, a form-based authoring wizard would be created to allow instructors and domain experts to create specific SKO based on the template.

To an instructor, a SKO is a series of forms where they enter their expert data and produce working tutoring modules that they can test immediately. For example, an advanced developer could make a SKO template for guiding a student through solving an Algebra problem. From this, a form would be generated to allow an instructor to specify solution steps and tutoring dialogs associated with each step. An instructor could complete this form multiple times to enter content for different problems. This development is intended to be collaborative. By storing SKO in cloud hosts, different authors can edit or test each module. This also greatly facilitates SKO delivery, as a web-based SKO can be directly tested after creation.

4 Integration with GIFT

As part of the project to integrate AutoTutor with GIFT, AutoTutor Lite is being broken down into distinct services to fit into the SKO framework. Rather than focus on the low-level details of how AutoTutor and GIFT are integrating, the high-level process will be outlined. There is no canonical set of services that a given tutoring system should be broken down into so that it can be integrated into GIFT. However, the general integration process followed by AutoTutor might serve as a model for other systems considering GIFT integration. This integration has five phases: 1. identifying complementary functionality, 2. determining distinct “parts” of the AutoTutor Lite system, 3. specifying the functionality, inputs, and outputs of each part, 4. building web services, and 5. working with GIFT developers to add these to the GIFT distribution.

In the first phase, to identify complementary functionality between GIFT and AutoTutor, a large table of various key features for each system was created. This table helped identify the tools that GIFT had already implemented and those that AutoTutor Lite could contribute. This process identified that AutoTutor’s main contributions were conversational pedagogical agents, interactive tutoring, improved student modeling, and semantic analysis tools to compare sentence similarity. In the second phase, the full AutoTutor Lite system was examined to find distinct parts: sets of functionality that could be meaningfully split into distinct components. GIFT is meant to be a generalized system, so re-usable components offer more value to the system. To find these divisions, we looked for parts that only needed and returned small, well-formed information from other parts (e.g., the semantic service can compare any two sets of words and return a similarity value). In the next phase, the functions, inputs, and outputs of each part were determined. After that, we started building web services for each part. Web services were used because they follow communication standards that mean that AutoTutor code does not need to be in the same language as the GIFT code, nor does it need to run on the same computer. Finally, as versions of these web services have been completed, they have been provided to GIFT for integration into the system. This is an important part of the process, as testing with GIFT has helped uncover hurdles about the scalability and limitations of these new services. As these services are completed, they are being integrated into releases of GIFT.

Overall, integration with GIFT dovetails with a larger movement of AutoTutor toward a service-oriented architecture. This redesign will not only help integration with GIFT, but also with other systems in the future. Figure 2 shows how AutoTutor services are expected to integrate into the GIFT framework. AutoTutor services are shown on the right side of the diagram and include the semantic analysis service (for analyzing user input), learner’s characteristic curve (LCC) service (a simple type of student model), tutoring service for AutoTutor Lite, a service for text-to-speech, and an animated agent service. Some of these components are already available as web services. Once these services are available, GIFT will be able to incorporate basic AutoTutor Lite tutoring as part of its framework. The message-passing SKO framework will then standardize how AutoTutor communicates with GIFT. Additional services not displayed are also anticipated, such as a persistent student model, authentication service, and services for wrapping assessments such as multiple choice tests.

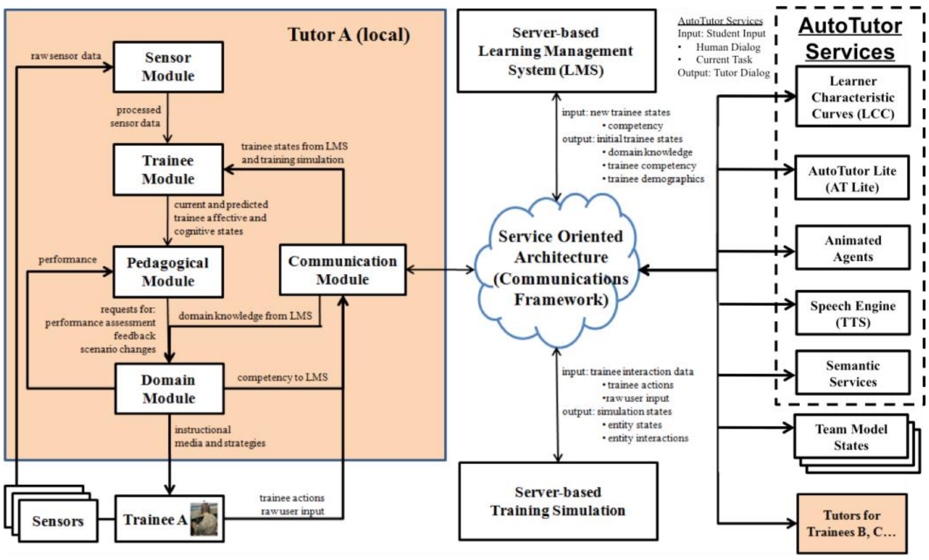

Tutor A (local)
raw sensor data
Sensor Module
processed sensor data
Trainee Module
trainee states from LMS and training simulation
current and predicted trainee affective and cognitive states
performance
Pedagogical Module
Communication Module
requests for: performance assessment feedback scenario changes
domain knowledge from LMS
Domain Module
competency to LMS
instructional media and strategies
Sensors
Trainee A
trainee actions raw user input
Server-based Learning Management System (LMS)
input: new trainee states
• competency
output: initial trainee states
• domain knowledge
• trainee competency
• trainee demographics
Service Oriented Architecture (Communications Framework)
input: trainee interaction data
• trainee actions
• raw user input
output: simulation states
• entity states
• entity interactions
Server-based Training Simulation
AutoTutor Services
Input: Student Input
• Human Dialog
• Current Task
Output: Tutor Dialog
AutoTutor Services
Learner Characteristic Curves (LCC)
AutoTutor Lite (AT Lite)
Animated Agents
Speech Engine (TTS)
Semantic Services
Team Model States
Tutors for Trainees B, C…

Fig. 7. Integration of AutoTutor and GIFT

5 Limitations and Future Directions

The SKO framework is intended to separate components based on the knowledge transferred between them, represented as semantic messages. This process will greatly improve modularity, enable AutoTutor to be implemented using a service-oriented design, and support interoperability with GIFT. However, modularity is limited by the information each component must share. Certain functions of the tutoring system are more easily separated into distinct components than others. For interoperating with additional tutoring systems, agreeing on a common set of messages may also be a challenge.

Currently, the publish-and-subscribe version Sharable Knowledge Object framework is under active development. In parallel with this work, AutoTutor Lite is being broken down into services and consumed by GIFT using traditional API's. Work in this area is focused on converting the semantic analysis services and AutoTutor Lite tutoring interpreter into services. Message-passing interfaces will then be incorporated into each service and they will be composed using the publish-and-subscribe SKO framework.

Acknowledgements The core SKO architecture is sponsored by the Office of Naval Research STEM Grand Challenge. Integration of AutoTutor Lite services with GIFT is sponsored by the Army Research Lab.

6 References

1. Graesser, A.C., Conley, M.W., Olney, A.: Intelligent tutoring systems. In: Harris, K.R., Graham, S., Urdan, T., Bus, A.G., Major, S., Swanson, H.L. (eds.) APA Educational psychology handbook, Vol 3: Application to learning and teaching, pp. 451–473. APA, Washington, DC (2012)
2. Graesser, A.C., Lu, S., Jackson, G.T., Mitchell, H.H., Ventura, M., Olney, A., Louwerse, M.M.: AutoTutor: A tutor with dialogue in natural language. Behavior Research Methods, Instruments, and Computers 36(2), 180–192 (2004a)
3. Graesser, A.C., McNamara, D.S., Louwerse, M.M., Cai, Z.: Coh-Metrix: Analysis of text on cohesion and language. Behavior Research Methods, Instruments, and Computers 36(2), 193–202 (May 2004b)
4. Hu, X., Cai, Z., Han, L., Craig, S.D., Wang, T., Graesser, A.C.: AutoTutor Lite. In: AIED 2009. IOS Press, Amsterdam, The Netherlands (2009)
5. Murray, T.: An overview of intelligent tutoring system authoring tools. In: Authoring Tools for Advanced Technology Learning Environments, pp. 493– 546 (2003)
6. Nye, B.D.: ITS and the digital divide: Trends, challenges, and opportunities. In: AIED 2013 (2013)
7. Nye, B.D., Graesser, A.C., Hu, X.: Multimedia learning in intelligent tutoring systems. In: Mayer, R.E. (ed.) Multimedia Learning (3rd Ed.). Cambridge University Press (2013)
8. Pavlik, P.I., Maass, J., Rus, V., Olney, A.M.: Facilitating co-adaptation of technology and education through the creation of an open-source repository of interoperable code. In: ITS 2012. pp. 677–678. Springer, Berlin (2012)
9. Pavlik, P.I., Toth, J.: How to build bridges between intelligent tutoring system subfields of research. In: Aleven, V and Kay, J and Mostow, J. (ed.) ITS 2010. LNCS, vol. 6095, pp. 103–112 (2010)
10. Peter Wiemer-Hastings, Arthur C. Graesser, Derek Harter: The foundations and architecture of AutoTutor. In: Goettl, B.P., Halff, H.M., Redfield, C.L., Shute, V.J. (eds.) ITS 1998. LNCS, vol. 1452, pp. 334–343. Springer, Berlin (Sep 1998)
11. Raymond, E.S.: The Art of UNIX Programming. Addison-Wesley (2003)
12. Roschelle, J., Kaput, J.: Educational software architecture and systemic impact: The promise of component software. Journal of Educational Computing Research 14(3), 217–228 (1996)
13. Sottilare, R.A., Goldberg, B.S., Brawner, K.W., Holden, H.K.: A modular framework to support the authoring and assessment of adaptive computer-based tutoring systems (CBTS). In: I/ITSEC (2012)
14. VanLehn, K.: The behavior of tutoring systems. International Journal of Artificial Intelligence in Education 16(3), 227–265 (2006)
15. Woolf, B.: Building intelligent interactive tutors: Student-centered strategies for revolutionizing e-learning (2009)

Authors:

Benjamin D. Nye is a post-doctoral fellow at the University of Memphis, working on tutoring systems architectures as part of the ONR STEM Grand Challenge. Ben received his Ph.D. from the University of Pennsylvania and is interested in ITS architectures, educational technology for development, and cognitive agents.

Leveraging a Generalized Tutoring Framework in Exploratory Simulations of Ill-Defined Domains

James M. Thomas[1], Ajay Divakaran[2], Saad Khan[2]

[1]*Soar Technology, Inc., Ann Arbor, MI*
jim.thomas@soartech.com

[2]*SRI International Sarnoff, Princeton NJ*
{ajay.divakaran, saad.khan}@sri.com

Abstract. Generalized frameworks for constructing intelligent tutors, such as GIFT promise many benefits. These benefits should be extended to systems that work in ill-defined domains, especially in simulation environments. This paper presents ideas for understanding how ill-defined domains change the tutoring dynamic in simulated environments and proposes some initial extensions to GIFT that accommodate these challenges.

Keywords. Intelligent Tutoring Systems, Computer-Based Tutoring Systems, framework, GIFT, ITS, CBTS, student models, learner models, task models, ill-defined domains, simulated environments, exploratory learning.

1 Introduction

Intelligent Tutoring System (ITS) have been shown to enhance learning effectiveness in a wide variety of academic domains [1]. The ITS field has long drawn inspiration from studying strategies employed by human tutors in one-on-one engagement with students [2]. Success has spurred the research community to extend its aspirations into more complex, ill-defined domains. Ill-defined domains are those that lack clearly defined procedures to determine the correctness of proposed solutions to specific problems [3]. Our interest lies in exploratory training simulations of those domains.

To address the difficulty of guiding effective learning in these complex environments, it seems useful to develop and leverage generalized techniques. The GIFT architecture represents a comprehensive approach to facilitate reuse of common tools and methods for building ITS. Although much of the initial focus of GIFT has been directed toward well-defined domains, it we would like to consider how it could be extended to ill-defined domains as well [4], especially those rendered through exploratory simulations.

The authors' motivating example is a system we are building called "Master Trainer – Individualized" (MT-I). The goal for this system is to intelligently guide new military squad leaders in simulations that combine intercultural communication and negotiation skills with tactical challenges. This system integrates stand-off assessments of student affect to modulate the intensity of the simulation to optimize student challenge. One of the questions we are investigating is how to drive the relationship between the student and a simulated human to achieve pedagogically useful levels of anger, conflict or cooperation. We are interested in applying what we are learning toward the generation of useful domain-independent strategies that could be incorporated into GIFT.

2 Motivations for GIFT

Although the field of ITS research is imbued with a strong collaborative spirit, the field lacks common computational infrastructure. GIFT is a particularly promising approach toward a general reusable framework for intelligent tutoring could benefit the entire field.

Scientific research largely presumes the capability to make apples-to-apples comparisons of competing theories. Although they share some common concepts and goals, the majority of ITS research systems share little common architecture or code [1]. To make broad contributions to this field often requires a fairly full-featured ITS on which to perform analyses, yet bespoke software development is both time consuming and expensive. Shared platforms and plug-ins amortize development costs and grow communities of professionals who can more effectively collaborate and relocate between projects and organizations, accelerating the productivity of the field as a whole [5].

GIFT proposes common frameworks for alternative implementations of a broad set of ITS capabilities. Built on solid design principles and a comprehensive understanding of the work of the ITS community, GIFT promises to serve an increasingly useful role in accelerating the scientific and commercial success of the field. Three common challenges faced by the field: authoring, instructional management, and analysis form the core constructs of GIFT. Successful evolution of these constructs promises to accelerate scientific progress by sharing common evaluation methodologies, reducing the time and expense for reused software components, and promoting a more tightly integrated and collaborative community.

GIFT may help accelerate commercialization of scientific progress by facilitating the production of a common currency of evidence of learning effectiveness that can be used to sell the benefits of implemented systems. It can help provide a platform for rapid prototyping to more quickly cycle through alternative approaches to find those that work best. Much as Eclipse™ has accelerated software development [5], and Unity3D™ has democratized high fidelity game development [6], GIFT has the potential to grow into a common workbench that builds-in the ability to package and deploy new work to a full breadth of possible platforms.

3 Characteristics of Ill-Defined Domains

The current GIFT vision accommodates a wide range of ITS capabilities. However, ill-defined domains have not been a primary component of that vision [4]. This section begins with an irony-free definition of ill-defined domains, describes some of the challenges encountered by human tutors in a subset of these domains, and then considers issues and opportunities they present for ITS designers working with immersive simulation environments.

3.1 Defining Ill-Defined Domains

Much of the historical grounding of ITS research is focused on guiding students through well-structured discrete learning tasks, to impart deeply decomposable knowledge [5] from well-defined domains. Fournier-Viger et al. [8] declare ill-defined domains to be those "where traditional approaches for building tutoring systems are not applicable or do not work well". Lynch and Pinkus [9] characterize problems in ill-defined domains as lacking definitive answers, having answers heavily dependent upon the problem's conception, and requiring students to both retrieve relevant concepts and map them to the task at hand. Mitrovic [10] underscores the important distinction between ill-defined domains and ill-defined tasks, anticipating Sottilare's [4] observation that ITS authoring in ill-defined domains is complicated by the multiplicity of "paths to success" compared to the more well-defined domains in which of ITS research has been situated.

3.2 Tutoring Challenges Posed by Ill-Defined Domains

Human tutors have served as both an inspiration for ITS behavior and benchmark and a benchmark for ITS performance [1]. Because no one has yet made a comprehensive study comparing human tutor behaviors in traditional domains with those in ill-defined domains to identify the most necessary extensions to tutorial reasoning, our work on the MT-I system is inspired by specific analogues of human tutors in the domains of live military training for tactics and intercultural effectiveness.

Live training in environments that combine military tactics and interpersonal challenges often spans many hours or days, ranges through confined indoor and expansive outdoor spaces, and requires dozens or hundreds of live role players. Interactions with these role players are often guided by scripted prompts, but involve a lot of improvisation as well. Examples include resolving disputes between armed civilians, negotiating with civic or spiritual leaders, as well neutralizing threats posed by snipers or potential ambushes. Trainers are usually embedded within the environment and have the ability to provide guidance to students during the simulation.

When comparing the behavior of the trainers/tutors in these exercises to that of academic tutors, a striking contrast is immediately evident. Feedback is often deferred over much longer intervals than what one would typically see in one-on-one tutoring in well-defined academic domains. Because is often unsafe or impossible to suspend exercises involving moving/flying vehicles and timed explosions, most incorporate extensive after-action review (AAR) as the primary conduit for feedback and guidance. To some extent, the tutors may elect to integrate feedback within the broader context of a scheduled AAR. In other cases, immediate feedback cannot be given on an individual student action choice because multiple student actions are required before a judgment can be made. Some of this deferral is linked to the interplay between student and role players, as it is difficult to guide the student without impacting the on-going social exchange. Finally, unlike many academic tasks, it is difficult to reset the problem state after an incorrect student action, as the training is situated in a narrative context with a fixed rate of flow to coordinate the many moving parts.

The immediate feedback tutors do provide in these simulations is often constrained to ensuring that the student is engaged and taking actions that move the implicit narrative forward. The deferred feedback is often a holistic reflection involving multiple learning objectives, student affect and metacognitive guidance on productive application of the feedback to future performance.

3.3 Tutoring in Computer Simulations of Ill-Defined Domains

Many of the challenges encountered by humans in ill-defined domains carry over into computer-based tutoring. The assessment granularity sometimes spans multiple actions, can sometimes be entwined in social interactions, and can sometimes be entwined in narrative. Each of these specific constraints can be viewed more generally.

What we commonly describe as narrative in simulation environments is more generally described as a meaningful continuity of state over time. Narrative-centered learning environments [11, 12] can vary in the extent to which they support alternative branches toward "success" or even emergent run-time generation. Yet they share the constraint that the continuity associated with the progression of states cannot be broken or the reversed without consequence, which in turn places limitations on the action choices available to both student and tutor.

Similarly, what we commonly perceive as social interaction between students and non-player characters (NPCs) in simulations is one particular case of an addition of simulation-based elements to tutorial state. In this case, it is the game-state data associated with the NPCs attitudes toward the student that persistent over sequences of tutorial actions. Other examples of game/simulation state variables that influence tutorial state include consumable or non-replenishable resources in the simulation which may affect the span of future tutorial choices.

Finally, the dependency on multiple student actions for student assessment is a specific manifestation of the well-recognized and more general problem of assessing student correctness at all in ill-defined domain. Yet while these challenges complicate the job of intelligent tutoring, they also introduce new tools. Narrative continuity can be exploited both to scaffold instruction and provide context for interpretation of actions. NPCs and other simulation based entities can be manipulated for pedagogical purposes, providing implicit guidance or challenge to the student. The complexity of interpretation of student action affords the intelligent tutor the opportunity for more nuanced and complex forms of guidance that may have more profound and lasting effects on learning.

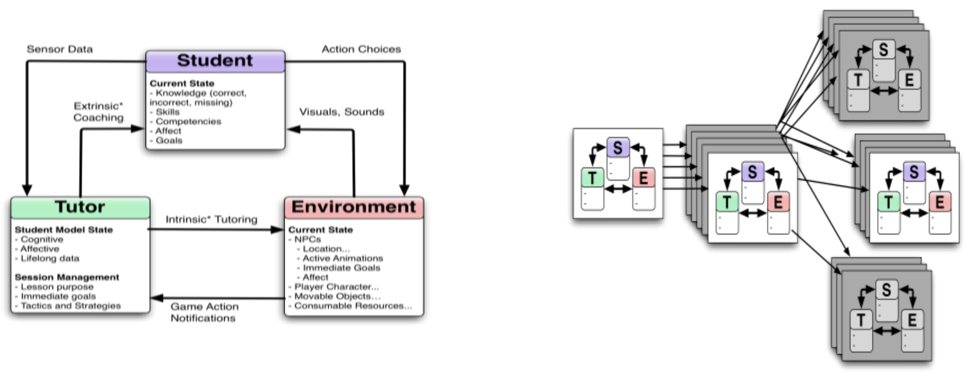


Fig. 8. Model of Tutoring Dynamic in Simulated Ill-Defined Domains

To best confront the challenges and make use of the opportunities of learning based in simulation environments over ill-defined domains, we need models that understand the tutoring dynamic as more than a one-on-one exchange. Rather, the model must recognize that the persistent state, continuity constraints, and assessment ambiguity of the simulation environment continually shape the interactions between tutor and student. Figure 1 is a depiction of such a model. At any point in time, the state of an ITS can be described as a combination of the state data associated with the student, the tutor <u>and</u> the simulation environment. Arrows depict the flow of state-changing actions between these three components. Note that some of these actions may proceed in parallel and may last for human-perceptible durations; perhaps with sufficient frequency that the overall state of the system may be more often in flux than it is quiescent.

This expanded interaction model complicates the prescription of the "learning effect chain" [4]. Because any change to student, tutor, or environment is represented as a new state, the progression toward learning gains involves navigating through a broad space of potential alternative paths. As shown in the rightmost half of Figure 1, one particular progression (the sequence of colored tutorial state snapshots depicted against white backgrounds) is merely one path through a rapidly expanding profusion of alternatives.

This tableau of interwoven learning progressions and alternatives gives an ambitious tutorial agent a lot to think about. Sufficiently inspired agents may perform plan-based reasoning to map the possibilities and nudge the learning experience in the most fruitful directions. In fact, tutorial agents have been constructed that mine the space of alternative actions sequences [12] to devise remediation strategies. Advanced agents might even consider choreographing multiple sessions, altering emphasis and tactics as it varies the pedagogical purpose of each session.

Alternatively, the profusion of possibilities can influence time-sensitive developers to move in the other direction, building "knowledge-lean" [13tutorial agents. As a consequence, ITS developers in these environments often eschew deep knowledge-tracing expert models in favor of less precise, but more easily authored constraint-based approaches [8]. This suggests that a generalized intelligent framework, such as

GIFT, should consider supporting a variety of modeling approaches. In fact, our current implementation of MT-I, which features ill-defined tasks within overlapping ill-defined domains, we have found it useful to author constraint-based models to characterize the correctness of individual student tasks in a wide range of potential contexts, where that model feeds a higher-level knowledge tracing model of higher-level, more abstract learning objectives that operates over longer time spans.

4 Enhanced Knowledge Representations and Reasoning

Not surprisingly, some of the challenges posed by ill-defined domains in simulated environments can be addressed by providing tools to create better definitions. Flexible and knowledge representations (KR) can serve as the definitional "handles" that tutorial agents can use to enhance reasoning about the state of the student and simulation. That reasoning can be converted to action if the simulation is instrumented with "levers", software hooks that cause pedagogically useful changes expressed through those handles. This section proposes three levers that use non-traditional extensions to tutorial knowledge representations to provide enhance tutorial reasoning and more effective student guidance.

Lever #1: Enriched Definitions of Learning Objectives. Trainers in the sophisticated simulations involving role players discussed earlier are often trying to steer their students toward states of mind that go beyond a prescribed set of factual knowledge to include social, narrative and affective dimensions, as shown in Figures 1 and 2. To achieve similar results in simulated environments, tutorial agents must reason about those dimensions of learning objectives as well. The KR should be able to qualify, for example, not only that the trainee know how to greet respectfully a village leader, but that the student can perform that greeting is accomplished while in a highly agitated state.

Lever #2: Enriched Definitions of Tutorial Purpose. Sophisticated simulations can be used in a broader set of pedagogical contexts that traditional systems, ranging from direct instruction of material to which the student has not previously been exposed, to consolidation of previously taught material, to transfer of knowledge to new domains, to assessment of knowledge and performance, to building confidence, teamwork or skills that apply acquired knowledge. Thus, the purpose of a given tutorial session can vary more widely than in traditional instruction, which demands that tutorial strategies and tactics be labeled according to their relevance for these various purposes. For example, a particular tutorial action may have a stronger positive effect on student self-efficacy that an alternative which may have a stronger positive effect on didactic specificity. An enhanced KR enables the tutorial agent to choose between these alternatives based on whether the purpose of the current session is to build confidence or impart knowledge.

Lever #3: Persistently-labeled Learner Data. To maximally leverage the opportunities of sophisticated learning environments, in which multiple learning sessions for varying learning purposes may span arbitrary time periods, individual student data must be persistent and pervasive: accessible and publishable at any level by any component of the tutorial framework. This allows tutorial agents running at various levels with various time horizons to tie together data collected on individual students across multiple sessions. For example, it could prove useful to know how quick a student is to anger, or which immediately reachable emotional states are most conducive to learning for a particular student, where that data may have been collected and stored in an earlier tutorial session by an agent using the same generalize frame work. All student model data should be tagged with its expected lifespan: step, task, session, application, or beyond. This enhances the ability of any particular tutorial agent to perform macro-level adaptation [14] to evolve learning across multiple domains that enhance domain-independent competencies.

5 Conclusions

A generalized framework like GIFT holds significant promise for accelerating scientific and commercial success of ITS. Yet one of the areas in which that acceleration is most desperately needed, ill-defined domains in simulated environments, are not addressed in depth by the current approach to GIFT. We suggest that a first step in this direction would to explore several extensions to the knowledge representations in GIFT to meet the demands of those environments.

6 Acknowledgements

This research effort was supported in part by contract W911QX-12-C-0149 from the US Army Research Laboratory. We gratefully acknowledge the interest and encouragement from Dr. Bob Sottilare at ARL. In addition, Scott Flanagan of Sophia Speira, LLC, has provided invaluable understanding of military training practices and domains.

7 References

1. VanLehn, K. (2011). The relative effectiveness of human tutoring, intelligent tutoring systems, and other tutoring systems. *Educational Psychologist*, *46*(4), 197-221.
2. Wood, D., Bruner, J. S., & Ross, G. (1976). The role of tutoring in problem solving*. *Journal of child psychology and psychiatry*, *17*(2), 89-100.
3. Minsky, M. (1995). Steps to artificial intelligence. In Luger, G. F., editor, *Computation and Intelligence, Collected Readings*, chapter 3, pages 47–90. AAAI, Menlo Park CA, and MIT Press.
4. Sottilare, B. (2012). Considerations in the development of ontology for a generalized intelligent framework for tutoring. *Proceedings of the International Defense and Homeland Security Simulation Workshop.*
5. Kidane, Yared H., and Peter A. Gloor. "Correlating temporal communication patterns of the Eclipse open source community with performance and creativity." *Computational and mathematical organization theory* 13.1 (2007): 17-27.
6. Torrente, Javier, et al. "Game-like simulations for online adaptive learning: A case study." *Learning by Playing. Game-based Education System Design and Development*. Springer Berlin Heidelberg, 2009. 162-173.VanLehn, K. (1990). Mind bugs: the origins of procedural misconception. MIT press.

7. VanLehn, K. (1990). Mind bugs: the origins of procedural misconception. MIT press.
8. Fournier-Viger, P., Nkambou, R., & Nguifo, E. M. (2010). Building Intelligent Tutoring Systems for Ill-Defined Domains. In *Advances in Intelligent Tutoring Systems* (pp. 81-101). Springer Berlin Heidelberg.
9. Lynch, C., Ashley, K., Aleven, V., & Pinkwart, N. (2006). Defining ill-defined domains; a literature survey. In Proceedings of the Workshop on Intelligent Tutoring Systems for Ill-Defined Domains at the 8th International Conference on Intelligent Tutoring Systems (pp. 1-10).
10. Mitrovic, A., & Weerasinghe, A. (2009). Revisiting Ill-Definedness and the Consequences for ITSs. Artificial Intelligence in Education: Building Learning Systems That Care: from Knowledge Representation to Affective Modelling, 200, 375.
11. Mott, B., McQuiggan, S., Lee, S., Lee, S. Y., & Lester, J. C. (2006). Narrative-centered environments for guided exploratory learning. In *Proceedings of the AAMAS 2006 Workshop on Agent-Based Systems for Human Learning* (pp. 22-28).
12. Thomas, J. M., & Young, R. M. (2009, July). Using Task-Based Modeling to Generate Scaffolding in Narrative-Guided Exploratory Learning Environments. In *Proceeding of the 2009 conference on Artificial Intelligence in Education: Building Learning Systems that Care: From Knowledge Representation to Affective Modeling* (pp. 107-114).
13. Lane, H. C., Core, M. G., Gomboc, D., Karnavat, A., & Rosenberg, M. (2007, January). Intelligent tutoring for interpersonal and intercultural skills. In *The Interservice/Industry Training, Simulation & Education Conference (I/ITSEC)*(Vol. 2007, No. 1). National Training Systems Association.
14. 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.

Authors

James M. Thomas: James M. (Jim) Thomas is a Research Scientist at Soar Technology. His current work includes intelligent training and assessment systems that aid children on the autism spectrum and guide soldiers to integrate socio-cultural and tactical skills. He received his Ph.D.degree in Computer Science from North Carolina State University in 2011. His dissertation entitled “Automated Scaffolding of Task-Based Learning in Non-Linear Game Environments”, explored general mechanisms to generate intelligent tutorial planning within exploratory learning environments. Jim has authored more than a dozen papers in the area of intelligent tutoring systems, automated planning, and computer games. His graduate studies were supported by a National Science Foundation Graduate Research Fellowship in Artificial Intelligence. Concurrent with his doctoral studies, Jim developed game-based learning systems designed to improve children’s social skills at the 3-C Institute for Social Development, including work which was published in the journal *Child Development*. Jim also benefits from 15 years of experience in the computer and telecommunications industries, including software development, management, and senior marketing management with IBM, BNR and Nortel Networks.

Ajay Divakaran: Ajay Divakaran, PhD is a Technical Manager at SRI International Sarnoff. He has developed several innovative technologies for multimodal systems for both commercial and government programs over the past 16 years. He currently leads SRI Sarnoff’s projects on Modeling and Analysis of Human Behavior for the DARPA SSIM project, ONR Stress Resiliency project, Army "Master Trainer" Intelligent Tutoring project among others. He worked at Mitsubishi Electric Research Labs for ten years where he was the lead inventor of the world’s first sports highlights playback enabled DVR, as well as a manager overseeing a wide variety of product applications of machine learning. He was elevated to Fellow of the IEEE in 2011 for his contributions to multimedia content analysis. He established a sound experimental and theoretical framework for human perception of action in video sequences, as lead-inventor of the MPEG-7 video standard motion activity descriptor. He serves on TPC’s of key multimedia conferences and served as an associate editor of the IEEE transactions on Multimedia from 2007 to 2011 and has two books and over 100 publications to his credit as well as over 40 issued patents. He received his Ph.D. degree in Electrical Engineering from Rensselaer Polytechnic Institute in 1993.

Saad Khan: Saad Khan is a Senior Computer Scientist at SRI International with 10 years of experience developing computer vision algorithms. He has led the design and development of advanced military training systems that can adapt to both training scenarios and learners’ behavior. He serves as PI/ Co-PI and technical lead on programs in multimodal sensing algorithms for immersive training for DARPA, ONR and PMTRASYS. He led the development and transition of APELL (Automated Performance Evaluation and Lessons Learned) training system. APELL is an immersive, interactive, Mixed Reality training system that has been successfully deployed at the Marines Camp Pendleton training facility. Prior to joining SRI Sarnoff, Dr. Khan conducted research on 3D model based object tracking and human activity analysis in the IARPA VACE program. His work in automated image based localization earned an Honorable Mention award at the International Conference of Computer Vision 2005. He has authored over 20 papers and has 2 issued patents. His work on multiple view human tracking is one of the most highly cited works in recent tracking literature. He received his PhD in Computer Science from University of Central Florida in 2008.

Toward "Hyper-Personalized" Cognitive Tutors

Non-Cognitive Personalization in the Generalized Intelligent Framework for Tutoring

Stephen E. Fancsali[1], Steven Ritter[1], John Stamper[2], Tristan Nixon[1]

[1]*Carnegie Learning, Inc.*
437 Grant Street, Suite 918
Pittsburgh, PA 15219, USA
{sfancsali, sritter, tnixon}@carnegielearning.com

[2]*Human-Computer Interaction Institute*
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213, USA
john@stamper.org

Abstract. We are starting to integrate Carnegie Learning's Cognitive Tutor (CT) into the Army Research Laboratory's Generalized Intelligent Framework for Tutoring (GIFT), with the aim of extending the tutoring systems to understand the impact of integrating non-cognitive factors into our tutoring. As part of this integration, we focus on ways in which non-cognitive factors can be assessed, measured, and/or "detected." This research provides the groundwork for an Office of the Secretary of Defense (OSD) Advanced Distributed Learning (ADL)-funded project on developing a "Hyper-Personalized" Intelligent Tutor (HPIT). We discuss the integration of the HPIT project with GIFT, highlighting several important questions that such integration raises for the GIFT architecture and explore several possible resolutions.

Keywords: Cognitive Tutors, intelligent tutoring systems, student modeling, affect, personalization, non-cognitive factors, gaming the system, off-task behavior, Generalized Intelligent Framework for Tutoring, GIFT

1 Introduction

Our goal in developing a "Hyper-Personalized" Intelligent Tutor (HPIT) is to bring learning systems to the next level of user/student adaptation. In addition to traditional features of systems like Carnegie Learning's Cognitive Tutor (CT), HPIT includes non-cognitive factors to provide a more personalized experience for users of the system. In this paper, we discuss features of HPIT and situate the work in the context of the Generalized Intelligent Framework for Tutoring (GIFT) architecture.

1.1 Cognitive Tutors

Carnegie Learning's Cognitive Tutor (CT) [1] is an adaptive, computer-based tutoring system (CBTS) or intelligent tutoring system (ITS) based on the Adaptive Control of Thought—Rational (ACT-R) theory of cognition [2] used every year by hundreds of thousands of learners, from middle school students through college undergraduates. To date, Carnegie Learning's development of the CT has focused primarily on mathematics.

1.2 Generalized Intelligent Framework for Tutoring (GIFT)

The Army Research Laboratory (ARL) is working to develop the Generalized Intelligent Framework for Tutoring (GIFT). The GIFT project aims to provide a "modular CBTS framework and standards [that] could enhance reuse, support authoring and optimization of CBTS strategies for learning, and lower the cost and skillset needed for users to adopt CBTS solutions for military training and education" [3]. Given substantial efforts in both academia and industry to develop ITSs, integrating aspects of this work with ARL's GIFT is important for future development. We briefly provide an overview of GIFT before describing a particular project that will integrate architecture for "hyper-personalized" versions of ITSs, like Carnegie Learning's CT, with GIFT.

GIFT provides a modular framework to achieve and support three goals or "constructs" [3]: (1) affordable, easy authoring of CBTS components, (2) instructional management for integrating pedagogical best practices, and (3) experimental analysis of effectiveness.

GIFT's service-oriented architecture (SOA) currently provides four modules, among other functional elements, around which CBTSs can be implemented and into which existing ITSs can be integrated. Three modules are domain-independent: the *Sensor Module*, *User Module*, and *Pedagogical Module*. The *Domain Module* contains all domain-specific content, including problems sets, hints, misconceptions, etc.

One functional element outside of "local tutoring processes" in the GIFT architecture is important for the present discussion: *Persistent Learner Models*. These models are intended to "maintain a long term view of the learner's states, traits, demographics, preference, and historical data (e.g., survey results, performance, competencies)" [3]. As we review several key, non-cognitive factors upon which we seek to base a "hyper-personalized" CT, the importance of data intended to be tracked by *Persistent Learner Models* will be clear. However, the notion of "persistence" for this data becomes less clear.

2 Non-Cognitive Factors

While the CT and other ITSs adapt content presented to students based on cognitive factors such as skill mastery, there are many other (cognitive and non-cognitive) factors for which the student learning experience might be adapted and personalized. We present several examples of recent research focusing on the impact of non-cognitive factors on student learning in ITSs.

2.1 Gaming the System and Off-Task Behavior

A wide variety of behaviors in an ITS or CBTS like CT may be associated with learning. Two specific behaviors that have been widely studied in the recent literature include "gaming the system" behavior and off-task behavior [4] [5]. This research has not only studied the association of these behaviors with learning but has also led to the development of software "detectors" of such behavior from ITS log data.

Sometimes students attempt to advance through material in ITSs like the CT without actually learning the content and developing mastery of appropriate skills. Such behavior is generally referred to as "gaming the system." Examples of such behavior include rapid or systematic guessing and "hint-abuse." "Hint-abuse" refers to repeated student hint requests, sometimes until a final or "bottom-out" hint (essentially) provides the answer to a problem or problem step [10].

Software "detectors" of gaming the system behavior have been developed (e.g., [7]) and correlated with field-observations of student behavior. Such software detectors rely on various features that are "distilled" from CT log files [8]. Studies find an association [4] [9] [10] and evidence for a causal link [11] [12] between gaming the system behavior and decreased student learning. Similar software has also been developed, and validated via field-observations, to detect off-task behavior [5].

Other types of behavior, of course, may also be important for learning in CBTSs and ITSs. While some behaviors may be "sensed" via physical, tactile, and/or physiological sensors, we emphasize that state-of-the-art research attempts to detect different types of behavior from logs generated by CBTSs and ITSs.

2.2 Affect

Building on success in developing detectors of student behavior, current research seeks to detect student affect (e.g., boredom, engaged concentration, frustration, etc.) without sensors (i.e., without physical, tactile, and/or physiological sensors) [13]. Such detectors have also been validated by field-observations of students using ITS in the classroom. Further, these detectors have been successfully deployed to predict student learning via standardized test scores [14].

While student affect and behavior might also be physically "sensed", inferred, or measured via survey instruments (e.g., mood via survey [15]), data-driven detection of student affect and behavior is a promising approach to achieve the GIFT design goal of supporting "low-cost, unobtrusive (passive) methods… to inform models to classify (in near real-time) the learner's states (e.g., cognitive and affective)" [3].

2.3 Preferences

Carnegie Learning's middle school mathematics CT product, MATHia, allows students to set preferences for various interest areas (e.g., sports, art) and probabilistically tailors problem presentation based on those preferences. On-going research aims to determine if and how presenting students with problems related to their preference areas is associated with engagement and benefits student learning (e.g., [16-17]). Other student preferences might be ascertained via surveys, configuration settings, or inferred from data, at different levels of granularity and time scales.

2.4 Personality and Other Learner Characteristics

Other characteristics of learners may prove important for learning. We consider two prominent examples that are being considered as we develop HPIT. Investigating other learner characteristics is also a topic for future research.

Grit.

Grit [18-19] is defined as the tendency to persist in tasks over the long term, when reaching the goal is far off in the future. Duckworth et al. [18] found that grit, measured by a survey instrument [19], predicted retention among cadets at the United States Military Academy at West Point, educational attainment among adults, and advancement to the final round among contestants in the Scripps National Spelling Bee.

Educational environments like CT are able to adjust the rate at which difficulty of activities increases. Students high in grit may, for example, benefit more from rapid increases in the difficulty of course material compared to students low in grit, regardless of knowledge-levels.

Motivation and Goals.

Students' motivation and goals are likely to be important for learner adaptation. Recent research [20] considers fine-grained assessment, via frequent surveys (occurring every few minutes) embedded within CT, of student motivation and goal orientation to better understand models self-regulated learning. Elliot's framework for achievement goals provides for two dimensions, definition (mastery vs. performance) and valence (approach vs. avoidance), along which goals are oriented [20-22].

Particular problems or hints (or ways of providing hints) might, for example, be best suited to students with a mastery avoidance goal orientation that seek to avoid failure, and so on. In addition to ascertaining the influence of goals and motivation on learning, determining whether students' motivation and goals (at various levels of granularity) are relatively static or dynamic through a course, and possibly influenced by students' experience in a course, remains a topic of active research [20].

3 Hyper-Personalizing Cognitive Tutors

One particularly important aspect of CTs from an architectural perspective is that they are driven by user inputs (called "transactions" [23]). From a system perspective, an update to the learner model happens only when the student takes some action within the system (e.g., attempting to answer a question or asking for a hint). Other student-initiated inputs might include, for example, student ratings of whether particular problems are interesting (e.g., an ever-present 5-star ranking system attached to each problem). Student-initiated inputs range in time from the nearly continuous to being separated by significant amounts of time.

In a more general system like GIFT, updates to the student model happen, not only at different timescales, but can also be initiated by actors (or factors) other than the learner. Examples include: acquiring data to update the student model through surveys given to the student at times determined by the system (e.g., only at course-beginning and end vs. periodically between problems or units), through real-time sensors (e.g., an eye-tracker), through student-determined inputs, etc. Furthermore, in some learning environments, the student model might be updated by factors linked to the passage of time (e.g., inferring that a skill has been "forgotten" because the student does not use a tutor for a substantial amount of time or updating students' knowledge state after a chemical reaction occurs following some time-lapse in a simulation-based chemistry tutor). The mode and frequency of data collection, in part, determine the kinds of pedagogical moves that the ITS can take.

The ADL-funded Hyper-Personalized Intelligent Tutor (HPIT) project seeks to develop a modular, plug-in-like architecture using various data collection and processing elements to inform CT's provision of problems, feedback, hints, etc. Each factor (whether cognitive or non-cognitive) may contribute to varying degrees to the decision-making process, as data are collected and inferences drawn about learner "state." A plug-in architecture allows for "voting" schemes to drive the personalization process (e.g., perhaps two non-cognitive factors and one cognitive factor are all equally weighted, or not). Methods will be developed to resolve conflicts (i.e., break "ties") when multiple recommendations are appropriate given a student's "state."

While cognitive factors are crucial for adapting educational content for disparate users, HPIT's primary innovation is the creation of a platform and framework for adapting content based on non-cognitive factors. To do so, HPIT will draw on data from software detectors, surveys, and possibly physical sensors. Perhaps more important from an architectural perspective, however, is the fact that the measurement, inference, or assessment of various cognitive and non-cognitive factors may occur on different time scales and at different levels of granularity.

For example, if a student is both bored (as, for example, inferred from a software detector applied to real-time log data) and uninterested in material currently being presented (as inferred from survey results), material similar in difficulty, but providing examples better suited to student preferences, might be presented. However, a different strategy might be required if we lack data about their interests. Adapting pedagogical strategies based on data that is currently available is a virtue of the flexibility of the HPIT architecture we are developing.

4 Implications for GIFT Architecture

The GIFT architecture and recent research (e.g., [15]) focuses on using physical sensors and surveys to gather information about a learner's non-cognitive state. The HPIT framework builds on work to infer/measure student state with surveys and software detectors that use data from tutor logs. These software detectors rely on data generated by the ITS following student-initiated input to the ITS. We discuss several implications for the GIFT architecture and the integration of existing ITSs into GIFT.

4.1 Surveys

In GIFT, *Persistent Learner Models* store survey results and communicate with the *User/Learner Module* via the SOA. However, HPIT requires that surveys be deployable at nearly any point in the learning experience, rather than simply before and after a "chunk" (e.g., unit) of course material. Furthermore, surveys/polls might be conducted that assess momentary characteristics of the student experience, rather than the persistent state of a student[6].

Some survey-like elements may be deployed nearly continuously. Thus, it might be initially attractive to conceptualize surveys are as a particular type of sensor. However, the processing of the type of survey data we have in mind seems fundamentally different than processing sensor data (e.g., an eye-tracker). Consider, for example, the previously noted five-star rating system for problems. While the rating system may be deployed for near-continuous collection of data, frequently students may not choose to rate many problems. Perhaps we find that a student who rates problems infrequently assigns two particular problems a 1-star (low) rating. Given the lack of input from this student, these data may be especially salient and require special consideration compared to a student who frequently rates problems, and with high variability. Such possibilities seem to suggest that we treat discrete survey data (even with high-polling rates) differently than sensors that continuously provide data.

4.2 (Sensor-Free) Detectors in the GIFT Architecture

For purposes of software implementation, detectors are essentially sensors (i.e., both process, filter, and/or aggregate streams of data to make inferences about student state); "detector processing" would be nearly identical to "sensor processing" within the *Sensor Module*. However, the input characteristics of software detectors are much different than those of sensors in the GIFT architecture, as the notion of a sensor within GIFT, to date, focuses on physical sensors. Detectors generally rely on student/user-initiated input mediated by the learning environment, but detectors might also be developed that do not rely on user-initiated input (e.g., a detector of "forgetting" based on time-lapse in usage of the ITS).

One possible resolution would have the *Domain Module* (and/or the *Tutor-User Interface*) as input to the *Sensors* element, so that software-based detectors that rely on tutor log data are also conceptualized as *Sensors*. This proposal may stretch the notion of *Sensors* too far. In response, one might include a new type of *Detector/Analysis Module* that would take *Domain Module* (and possibly *Pedagogical Module* or *Tutor-User Interface*) data as input and provide information to the *User Module* about learners' affective and cognitive states via software detectors. This achieves the goal of keeping the relatively domain-independent detectors outside of the *Domain Module*. This requires that *Domain Module* output is sufficiently rich for use by detectors; as currently conceptualized, this is not clear.

[6] The HPIT architecture maintains such flexibility so that the investigator is free to make (or not make) distinctions about persistent versus non-persistent student characteristics (and concomitant timing decisions about assessment, measurement, or detection).

5 Discussion

Overall, we suggest that the GIFT architecture is well-served by considering the consequences of integrating a broader range of input and output relationships among its component modules (or possibly new types of modules) and other functional elements, including considerations of the presence, timing, granularity, and content of data passed between components.

Current research provides for data-driven means to use CT (and other CBTS) logs to classify and "detect" student behavior and affect without physical sensors, whether transactions and inputs are student-initiated or system-initiated. Integrating capabilities necessary for HPIT will be a fruitful extension of GIFT.

Furthermore, detectors rely on data from the ITS to determine whether students are off-task, gaming, bored, frustrated, etc. Such detectors require relatively rich log data and would not be served by the impoverished (i.e., abstract) assessment categories of "above standard," "below standard," etc., provided by the *Domain Module*. This suggests that detectors are a part of the *Domain Module*, but they are also (relatively) domain independent. Thus, it is not clear that they should be included in the *Domain Module*. Requiring detectors be a part of the *Domain Module* would also incur costs in terms of reusability and modularity. Alternatively, richer data might be provided to an enhanced *Learner Module* that subsumes (aspects of) the *Sensor Module* and our proposed detectors (i.e., the *Detector/Analysis Module*) to better infer characteristics of a learner's state. Further, other open questions remain as to the proper placement of other components of CTs within the GIFT architecture.

6 References

1. Ritter, S., Anderson, J.R., Koedinger, K.R., Corbett, A.T.: Cognitive Tutor: Applied Research in Mathematics Education. Psychonomic Bulleting & Review 14, 249-255 (2007)
2. Anderson, J.R.: Rules of the Mind. Erlbaum, Hillsdale, NJ (1993)
3. Sottilare, R.A., Brawner, K.W., Goldberg, B.S., Holden, H.K.: The Generalized Intelligent Framework for Tutoring (GIFT). (2012), http://www.gifttutoring.org/
4. Baker, R. S., Corbett, A. T., Koedinger, K .R., & Wagner, A. Z.: Off-Task Behavior in the Cognitive Tutor Classroom: When Students "Game the System". In: Proceedings of ACM CHI 2004: Computer-Human Interaction, pp. 383-390 (2004)
5. Baker, R.S.J.d.: 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 (2007)
6. Aleven, V., & Koedinger, K. R.: Limitations of Student Control: Do Students Know When They Need Help? In: Proceedings of the 5th International Conference on Intelligent Tutoring Systems, pp. 292-303 (2000)
7. Baker, R.S.J.d., de Carvalho, A. M. J. A.: Labeling Student Behavior Faster and More Precisely with Text Replays. In: Proceedings of the 1st International Conference on Educational Data Mining, pp. 38-47 (2008)
8. Baker, R.S.J.d., Corbett, A.T., Roll, I., Koedinger, K.R.: Developing a Generalizable Detector of When Students Game the System. User Modeling & User-Adapted Interaction 18, 287-314 (2008)

9. Walonoski, J.A., Heffernan, N.T.: Detection and Analysis of Off-Task Behavior in Intelligent Tutoring Systems. In: Proceedings of the 8th International Conference on Intelligent Tutoring Systems, pp. 382-391 (2006)
10. Cocea, M., Hershkovitz, A., Baker, R.S.J.d.: The Impact of Off-Task and Gaming Behavior on Learning: Immediate or Aggregate? In: Proceedings of the 14th International Conference on Artificial Intelligence in Education, pp. 507-514 (2009)
11. Fancsali, S.E.: Variable Construction and Causal Discovery for Cognitive Tutor Log Data: Initial Results. In: Proceedings of the Fifth International Conference on Educational Data Mining, pp. 238-239 (2012)
12. Fancsali, S.E.: Constructing Variables that Support Causal Inference. Ph.D. Thesis, Department of Philosophy, Carnegie Mellon University, Pittsburgh, PA, USA (2013)
13. 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.: Sensor-free Automated Detection of Affect in a Cognitive Tutor for Algebra. In: Proceedings of the 5th International Conference on Educational Data Mining, pp. 126-133 (2012)
14. Pardos, Z.A., Baker, R.S.J.d., San Pedro, M.O.C.Z., Gowda, S.M., Gowda, S.M.: Affective States and State Tests: Investigating How Affect Throughout the School Year Predicts End of Year Learning Outcomes. In: Proceedings of the 3rd International Conference on Learning Analytics and Knowledge (2013)
15. Sottilare, R., Proctor, M.: Passively Classifying Student Mood and Performance within Intelligent Tutors. Educational Technology & Society 15, 101-114 (2012)
16. Walkington, C., Sherman, M.: Using Adaptive Learning Technologies to Personalize Instruction: The Impact of Interest-Based Scenarios on Performance in Algebra. In: Proceedings of the 10th International Conference of the Learning Sciences, pp. 80-87 (2012)
17. Walkington, C.: Using Learning Technologies to Personalize Instruction to Student Interests: The Impact of Relevant Contexts on Performance and Learning Outcomes. Journal of Educational Psychology (forthcoming)
18. Duckworth, A.L., Peterson, C., Matthews, M.D., Kelly, D.R.: Grit: Perseverance and Passion for Long-Term Goals. Journal of Personality and Social Psychology 92, 1087-1101 (2007)
19. Duckworth, A.L., Quinn, P.D.: Development and Validation of the Short Grit Scales (Grit-S). Journal of Personality Assessment 91, 166-174 (2009)
20. Bernacki, M. L., Nokes-Malach, T.J., Aleven, V.: Fine-Grained Assessment of Motivation Over Long Periods of Learning with an Intelligent Tutoring System: Methodology, Advantages, and Preliminary Results. In: Azevedo, R., Aleven, V. (eds.) International Handbook of Metacognition and Learning Technologies. Berlin: Springer (forthcoming)
21. Elliot, A. J., & McGregor, H. A.: A 2 X 2 Achievement Goal Framework. Journal of Personality and Social Psychology 80, 501-519 (2001)
22. Elliot, A. J., & Murayama, K.: On the Measurement of Achievement Goals: Critique, Illustration, and Application. Journal of Educational Psychology 100, 613-628 (2008)
23. Koedinger, K.R., Baker, R.S.J.d., Cunningham, K., Skogsholm, A., Leber, B., Stamper, J.: A Data Repository for the EDM Community: The PSLC DataShop. In: Romero, C., Ventura, S., Pechenizkiy, M., Baker, R.S.J.d. (eds.) Handbook of Educational Data Mining, pp. 43-55 Boca Raton, FL: CRC Press (2011)

Authors

Stephen E. Fancsali is a Research Scientist at Carnegie Learning, Inc. He received a Ph.D. in Logic, Computation, and Methodology from the Philosophy Department at Carnegie Mellon University in May 2013. His doctoral research centered on the construction of variables from fine-grained data (e.g., intelligent tutoring system log files) to support causal inference and discovery from observational data sets. At Carnegie Learning, he focuses on a variety of issues in educational data mining, including student modeling, providing interpretable ways to quantify improvements in cognitive models and other tutoring system components, and statistical and causal modeling of student affect, behavior, and other phenomena as they relate to learning and other education outcomes, especially in intelligent tutoring systems.

Steven Ritter is Co-Founder and Chief Scientist at Carnegie Learning, Inc. He received a Ph.D. in Psychology from Carnegie Mellon University.

John Stamper is Technical Director of the Pittsburgh Science of Learning Center (PSLC) and a faculty member at the Human-Computer Interaction Institute at Carnegie Mellon University. He received a Ph.D. in Information Technology from the University of North Carolina at Charlotte.

Tristan Nixon is a Research Programmer at Carnegie Learning, Inc. He earned a B.S. in Computer Science from the University of Toronto.

Using GIFT to Support an Empirical Study on the Impact of the Self-Reference Effect on Learning

Anne M. Sinatra, Ph.D.

Army Research Laboratory/Oak Ridge Associated Universities
anne.m.sinatra.ctr@us.army.mil

Abstract. A study is reported in which participants gained experience with deductive reasoning and learned how to complete logic grid puzzles through a computerized tutorial. The names included in the clues and content of the puzzle varied by condition. The names present throughout the learning experience were either the participant's own name, and the names of two friends; the names of characters from a popular movie/book series (*Harry Potter*); or names that were expected to have no relationship to the individual participant (which served as a baseline). The experiment was administered using the Generalized Intelligent Framework for Tutoring (GIFT). GIFT was used to provide surveys, open the experimental programs in PowerPoint, open external web-sites, synchronize a Q-sensor, and extract experimental data. The current paper details the study that was conducted, discusses the benefits of using GIFT, and offers recommendations for future improvements to GIFT.

1 Introduction

The Generalized Intelligent Framework for Tutoring (GIFT) provides an efficient and cost effective way to run a study (Sottilare, Brawner, Goldberg, & Holden, 2012). In Psychology research, in-person experiments usually require the effort of research assistants who engage in opening and closing computer windows and guiding participants through the experimental session. GIFT provides an opportunity to automate this process, and requires a minimal knowledge of programming, which makes it an ideal tool for students and researchers in the field of Psychology. GIFT was utilized in the current pilot study, which is investigating the impact of the self-reference effect on learning to use deductive reasoning to solve logic grid puzzles.

1.1 The Self-Reference Effect and Tutoring

Thinking of the self in relation to a topic can have a positive impact on learning and retention. This finding has been consistently found in Cognitive Psychology research, and is known as the self-reference effect (Symons & Johnson, 1997). In addition, research has suggested that linking information to a popular fictional character (e.g., *Harry Potter*) can also draw an individual's attention when they are engaged in a difficult task, and can result in similar benefits to the self-reference effect (Lombardo, Barnes, Wheelwright, & Baron-Cohen, 2007; Sinatra, Sims, Najle, & Chin, 2011). The self-reference effect could potentially be utilized to provide benefits in tutoring and learning. Moreno and Mayer (2000) examined the impact of a participant being taught science lessons in a manner consistent with first person speech (self-reference), or in the third person. No difference was found in regard to knowledge gained from the lessons, however, when asked to apply the knowledge in a new and creative way, those that received the first person instruction demonstrated better performance. This suggests that relating information to the self may result in a "deeper" learning or understanding, which allows the individual to easily apply the information in new situations.

It has been suggested that deep learning should be a goal in current instruction (Chow, 2010). This is consistent with findings that including topics of interest (e.g., familiar foods, names of friends) when teaching math can have a positive impact on learning outcomes (Anand & Ross, 1987; Ku & Sullivan, 2002). Many of the domains (e.g., math, science) that have been examined in the literature are "well-defined" and do not transfer skills to additional tasks. There has not been a focus on deductive reasoning or teaching logic, which is a highly transferable skill. Logic grid puzzles are useful learning tools because they allow an individual to practice deductive reasoning by solving the puzzle. In these puzzles, individuals are provided with clues, a grid, and a story. The story sets up the puzzle, the clues provide information that assists the individual in narrowing down or deducing the correct answers and the grid provides a work space to figure out the puzzle. It has been suggested that these puzzles can be helpful in instruction, as they require the individual to think deeply about the clues and have a full understanding of them in order to solve the puzzle (McDonald, 2007). After practicing deductive reasoning with these puzzles, the skill can then potentially be transferred and applied in many other domains and subject areas.

1.2 The Current Study

The current study sets out to examine the self-reference effect in the domain of deductive reasoning, by teaching individuals how to complete logic grid puzzles. It is a pilot study, which will later be developed into a large scale study. During the learning phase of the study, there were three different conditions: Self-Reference, Popular Culture, and Generic. The study was administered on a computer using GIFT 2.5. The interactive logic puzzle tutorial was developed using Microsoft PowerPoint 2007 and Visual Basic for Applications (VBA). In the Self-Reference condition, participants entered their own name and the names of two of their close friends into the program, in the Popular Culture condition, the participant was instructed to enter names from the *Harry Potter* series ("Harry", "Ron", and "Hermione") into the program, in the Generic condition, participants were instructed to enter names which were not expected to be their own ("Colby", "Russell", and "Denise") into the program. The program then used the entered names throughout the tutorial as part of the clues and the puzzle with which the participants were being taught. Therefore, the participants were actively working with the names throughout their time learning the skill.

After completing the tutorial, participants were asked to recall anything that they could about the content of the puzzle, answer multiple-choice questions about what they learned, answer applied clue questions in which they were asked to draw conclusions based on a story and an individual clue, and complete two additional logic puzzles (one at the same difficulty level as the one in the tutorial, and one more difficult). These different assessments allowed for measures of retention of the learned content, ability to apply the knowledge, and ability to transfer/apply the knowledge in a new situation.

It was hypothesized that there would be a pattern of results such that individuals in the Self-Reference condition would perform better on all assessments than that in the Popular Culture and Generic conditions, and that the Popular Culture condition would perform better on all assessments than the Generic condition. It was also expected that ratings of self-efficacy and logic grid puzzle experience would increase after the tutorial.

1.3 GIFT and the Current Study

The current study required participants to use a computer, and answer survey questions before and after PowerPoint Tutorials and PowerPoint logic grid puzzles. Due to the capabilities of GIFT 2.5 to provide survey authoring and administering, it was an ideal choice for the development of the study. As GIFT has the capability of opening and closing programs (such as PowerPoint), and presenting surveys and instructions in specific orders, it is a highly efficient way to guide participants through a learning environment and a study, without much effort from research assistants.

In Psychology research there are often many different surveys that are administered to participants. An advantage of GIFT is that the Survey Authoring System provides a free and easy to use tool in which to create surveys. A further advantage is that it does not require the individual to be online when answering the survey.

2 Method

2.1 Participants

In the current pilot study, there were 18 participants recruited from a research organization, and a University. Participants did not receive any compensation for their participation. The sample was 55.6% male (10 participants) and 44.4% female (8 participants). Reported participant ages ranged between 18 years and 51 years (*M* = 28.8 years, *SD* = 9.2 years). As there were 3 conditions, there were 6 participants in each condition.

2.2 Design

The current study employed a between subjects design. The independent variable was the types of names included in the tutorial during the learning phase of the study. There were three conditions: Self-Reference, Popular Culture, and Generic. The dependent variables were ratings of self-efficacy before and after the tutorial, ratings of logic grid puzzle experience after the tutorial, performance on a multiple-choice quiz about the content of the tutorial, performance on applied logic puzzle questions (which asked the participants to apply the skill they learned in a new situation), performance on a logic puzzle of the same difficulty as the tutorial, and on one that was more difficult.

2.3 Apparatus

Laptop and GIFT. The study was conducted on a laptop that was on a docking station, and connected to a monitor. GIFT 2.5 and PowerPoint 2007 were installed on the laptop, and a GIFT course was created for each condition of the experiment.

Q-sensor. Participants wore a Q-sensor on their left wrists. It is a small band approximately the size of a watch, which measures electrodermal activity (EDA).

2.4 Procedure

Upon arriving, participants were given an informed consent form, and the opportunity to ask questions. For this pilot study, participation occurred individually. After signing the form, participants were randomly assigned to a condition. The experimenter launched ActiveMQ and the GIFT Monitor on the computer. Participants were then fitted with the Q-sensor on their left wrists. The experimenter clicked "Launch all Modules" and then proceeded to synchronize the Q-sensor with the computer. If synchronization was unsuccessful after three tries, the experimenter edited the GIFT sensor configuration file and changed the sensor to the Self Assessment Monitor as a placeholder (the data from it was not used). Next, the "Launch Tutor Window" button was clicked, and the experiment was launched in Google Chrome. The experimenter created a new UserID for the participant, and then logged in. The correct condition was then selected from the available courses. The participants were then instructed that they should interact with the computer and let the experimenter know if they had any questions.

Participants were first asked to answer a few brief demographics questions (e.g., age/gender) and filled out Compeau and Higgins' (1995) Self Efficacy Questionnaire (SEQ) with regard to their beliefs in their ability to solve a logic grid puzzle in a computer program and rated their logic grid puzzle experience. They then began the Tutorial. Depending on the condition they were in, they received different instructions in regard to the names to enter (their own name and the name of friends, *Harry Potter* related names, or General names). They then worked through the tutorial that walked them through completing a logic grid puzzle and explained the different types of clues.

After completing the tutorial, they filled in the SEQ again, rated their experience again, and were asked to report any information they remembered from the content of the puzzle. Next, they answered 20 multiple choice questions about the material they learned about in the tutorial. Then, they answered 12 applied clue questions, which provided a story and an individual clue, then asked the participants to select all of the conclusions that could be drawn from that clue. Next, participants had 5 minutes in which to complete an interactive PowerPoint logic grid puzzle at the same level of difficulty as the one that they worked through in the tutorial, and 10 minutes to complete a more difficult puzzle. Finally, they were directed to an external web-site to complete a personality test. They wrote their scores on a piece of paper, and entered them back into GIFT. Afterward, they were given a debriefing form and the study was explained to them.

2.5 GIFT and the Procedure

The Survey Authoring System in GIFT was used to collect survey answers from the participants. While it was a fairly easy to use tool to enter the questions initially, there was some difficulty in the export function. Instead of exporting all the entered questions, there appeared to also be previously deleted questions within the files that were exported. This made it impossible to simply import the questions into an instance of GIFT on an additional computer (in order to have an identical experiment on more than one computer). As a work around, the questions had to be manually typed in and added to each additional computer that was used for the study.

A course file was generated using the Course Authoring Tool. The tool was also fairly easy to use. It provided the ability to author messages that the participant would see between surveys and training applications, determine the specific surveys and PowerPoint applications that would be run, and the order in which they would run. Further, it could send participants to an external web-site; however, while the participants were on the site there was no ability to keep instructions on the screen. Participants only saw a “Continue” button at the bottom of the screen – which may have led to some participants in the current study clicking “Continue” before filling out the surveys they needed to on the web-site. A solution to this was employed by creating a PowerPoint to explain what the participants would be doing on the web-site. However, having the ability to author comments that are seen by the participant while they are on the external web-site would be beneficial.

3 Results

3.1 Pilot Study Results

Performance Results. A series of One Way ANOVAs were run for the percentages correct on the multiple choice questions [$F(2,15) = .389$, $p = .684$], applied clue questions [$F(2,15) = 2.061$, $p = .162$], the easier assessment logic puzzle [$F(2,15) = 3.424$, $p = .060$] and the more difficult logic puzzle [$F(2,15) = 1.080$, $p = .365$]. However, there were no significant differences found between conditions for any of the assessments. See Table 1 for the means and standard deviations for each condition and DV.

	Self-Reference	Popular Culture	Generic
Multiple Choice	*M* = 96.67%, *SD* = 2.58%	*M* = 95.83%, *SD* = 6.65%	*M* = 94.17%, *SD* = 4.92%
Applied Clue	*M* = 80.55%, *SD* = 16.38%	*M* = 87.50%, *SD* = 11.48%	*M* = 69.44%, *SD* = 18.00%
Easy Logic Puzzle	*M* = 51.95%, *SD* = 37.47%	*M* = 93.21%, *SD* = 16.63%	*M* = 74.07%, *SD* = 23.89%
Difficult Logic Puzzle	*M* = 69.78%, *SD* = 24.61%	*M* = 76.89%, *SD* = 16.49%	*M* = 86.89%, *SD* = 19.31%

Table 5. Means and Standard Deviations for Performance Variables for each condition

Logic Grid Puzzle Experience. A 3 (Condition) x 2 (Time of Logic Puzzle Experience) Mixed ANOVA was run comparing the conditions and participant's self rating of their logic grid puzzle experience. Overall, participants indicated that they had significantly higher logic grid puzzle experience after the tutorial (*M* = 3.78, *SD* = 1.215) than before (*M* = 2.00, *SD* = 1.085), *F*(1,15) = 28.764, *p*<.001. However, there was no significant interaction between condition and logic grid puzzle experience ratings, *F*(2, 15) = .365, *p* = .700.

Self Efficacy Questionnaire. A 3 (Condition) x 2 (Time of SEQ score) Mixed ANOVA was run comparing the conditions and the scores on the logic grid puzzle self-efficacy questionnaire. There were significantly higher scores of self-efficacy after tutoring regardless of condition (*M* = 5.583, *SD* = .6564) than before tutoring (*M* = 5.117, *SD* = .7618), *F*(1,15) = 9.037, *p* = .009. However, the condition did not seem to matter, as there was not a significant interaction between condition and time of SEQ score, *F*(2,15) = .661, *p* = .531.

3.2 Using GIFT to extract the information and results

The Event Reporting Tool was used to export survey data from GIFT. However, in the initial GIFT 2.5 version, data from only one participant would export at a time. These files were manually copied and pasted together into one Excel file for analysis. An updated version of GIFT 2.5 offered the ability to export multiple participant files at once. However, if using multiple instances of GIFT on separate computers, it is important to name the questions identically. Combining the outputs of questions that have different names in the survey system may result in the data for those columns not being reported for certain participants.

4 Discussion

4.1 Pilot Results Discussion

The results indicate that the tutorial was successful in teaching participants the skill of completing logic grid puzzles, and made them feel more confident in their abilities than before tutoring. However, the manipulation of the names present in the puzzle during tutoring did not impact performance. As this is a small pilot study, it likely did not have enough power to find results. Currently there are only 6 participants in each condition. The full study is expected to have at least 40 participants in each condition. Individual differences in the ability of individuals to solve the puzzles and the wide variety of ages may also have played a role in the results. Based on the experience with this pilot study, some changes have been made to the full-scale study. First, a pre-test of applied clue questions will be given. Secondly, as not all the participants were able to finish the easier logic puzzle in 5 minutes, the amount of time given for this task will be increased. It is also possible that the current "tests" are not sensitive enough to differences. Further, the sample population for the pilot is different than the intended population for the full-scale study (college students), therefore, those with less research and logic training may show different results.

4.2 GIFT Discussion and Recommendations

GIFT was extremely useful in the current study. During this pilot, participants were able to easily understand and interact with the courses developed with GIFT. All of the survey data was recorded and able to be cleaned up for analysis. One improvement that could be made would be to change the UserID system. Currently, it is set up such that UserIDs are created one by one and in order. It would be beneficial to be able to assign a specific participant number as the User ID in order to reduce confusion when exporting the results (e.g. "P10" rather than "1"). Further, improvements could be made to the ability to launch an external web-site. Currently, there is no ability to provide on-screen directions to individuals while they are on the page. While the Survey Authoring System is useful, it could be greatly improved by having a more reliable import/export option for questions and entire surveys. By doing so, it would be easier to set up identical instances of GIFT on multiple computers.

Overall, GIFT is a useful, cost effective tool which is an asset in running a study. It has a wide variety of helpful functions, and with each release the improvements will likely make it even more valuable to researchers who adopt it.

5 References

1. 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.
2. Compeau, D.R., & Higgins, C.A. (1995). Computer self-efficacy: Development of a measure and initial test. *MIS Quarterly, 19* (2), 189 – 211.

3. Chow B. (October 6th, 2010). The quest for deeper learning. *Education Week*, Retrieved from http://www.hewlett.org/newsroom/quest-deeper-learning
4. Ku, H-Y, & Sullivan, H.J. (2002). Student performance and attitudes using personalized mathematics instruction. *ETR&D, 50* (1), 21 – 34.
5. Lombardo, M.V., Barnes, J.L., Wheelwright, S.J., & Baron-Cohen, S. (2007). Self-referential cognition and empathy in autism. *PLOS one, 2* (9), e883.
6. McDonald, K. (2007). Teaching L2 vocabulary through logic puzzles. *Estudios de Linguistica Inglesa Aplicada, 7,* 149 – 163.
7. Moreno, R., & Mayer, R.E. (2000). Engaging students in active learning: The case for personalized multimedia messages: *Journal of Educational Psychology, 92* (4), 724 – 733.
8. Sinatra, A.M., Sims, V.K., Najle, M.B., & Chin, M.G. (2011, September). An examination of the impact of synthetic speech on unattended recall in a dichotic listening task. *Proceedings of the Human Factors and Ergonomics Society, 55*, 1245 – 1249.
9. 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).
10. Symons, C.S., & Johnson, B.T. (1997). The self-reference effect in memory: A meta-analysis. *Psychological Bulletin, 121* (3), 371 – 394.

6 Acknowledgment

Research was sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-12-2-0019. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of the 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.

Authors

Anne M. Sinatra: Anne M. Sinatra, Ph.D. is an Oak Ridge Associated Universities Post Doctoral Fellow in the Learning in Intelligent Tutoring Environments (LITE) Lab at the U.S. Army Research Laboratory's (ARL) Simulation and Training Technology Center (STTC) in Orlando, FL. The focus of her research is in cognitive and human factors psychology. She has specific interest in how information relating to the self and about those that one is familiar with can aid in memory, recall, and tutoring. Her dissertation research evaluated the impact of using degraded speech and a familiar story on attention/recall in a dichotic listening task. Prior to becoming a Post Doc, Dr. Sinatra was a Graduate Research Associate with the University of Central Florida's Applied Cognition and Technology (ACAT) Lab, and taught a variety of undergraduate Psychology courses. Dr. Sinatra received her Ph.D. and M.A. in Applied Experimental and Human Factors Psychology, as well as her B.S. in Psychology from the University of Central Florida.

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

Jeanine A. DeFalco[1], Ryan S.J.d.Baker[1]

[1]*Teachers College, Columbia University, New York, NY*
jad2234@tc.columbia.edu, baker2@exchange.tc.columbia.edu

Abstract. Developing intelligent tutoring systems that respond effectively to trainee or student affect is a key part of education, particularly in domains where learning to regulate one's emotion is key. Effective affect response relies upon effective affect detection. This paper discusses an upcoming cooperative study between the Army Research Laboratory, Teachers College, Columbia University, and North Carolina State University, with the goal of developing automated detectors that can infer trainee affect as trainees learn by interacting with the vMedic system, which trains learners in combat medicine. In this project, trainee interactions with vMedic will be synchronized with observations of engagement and affect; and physical sensor data on learners, obtained through GIFT's Sensor Module. The result will be models of trainee affect, ready for integration into the GIFT platform, which can leverage sensor information when available, but which can make reasonably accurate inference even without sensor data.

Keywords: GIFT, vMedic, affect, tutoring, intelligent tutoring systems, learning, automated detectors, game-based training

1 Introduction

In recent years, there has been increasing interest in modeling affect within intelligent tutoring systems [7, 11] and using these models to drive affect-sensitive interventions [2]. In this paper, we describe an ongoing collaborative project between the Army Research Laboratory, Teachers College Columbia University, and North Carolina State University, which has the goal of developing automated detection of trainee affect that can leverage sensors when they are available, but which can function robustly even when sensors are not available.

Within this research, trainee affect will be studied in the context of the vMedic, (a.k.a. TC3Sim), a game developed for the U.S. Army by Engineering and Computer Simulations (ECS) in Orlando, Florida, to train combat medics and combat lifesavers on providing care under fire and tactical field care. Trainees will also complete material on hemorrhage control within the auspices of the GIFT framework [12], the Army Research Laboratory's modular framework for Computer-Based Training Systems, with the goal of integrating eventual affect detection into the GIFT framework's User

Module (realized as necessary within the Sensor Module). In turn, the affect detectors will be built into the pedagogies realized through the GIFT Framework's Pedagogical Module, for instance to realize interventions through the embedded instructor and other non-player characters.

In this fashion, this project will contribute not just to the assessment of affect within vMedic, but also to the GIFT framework's broader goal of integrating a range of types of models and detectors into the GIFT framework. By serving as a test case for incorporating two types of detection into GIFT (sensor-free affect detection, and sensor-based affect detection), this project will assist in understanding how GIFT needs to be enhanced to incorporate the full range of models currently being developed by this research community.

Using these detectors, further work will be conducted to study student affective trajectories within vMedic, which affective and engagement states influence learning of the key material within vMedic, and how trainee affect can best be supported based on the results of affect detection. The work to study the relationship between affect, engagement, and outcome variables will provide important evidence on which affective states and engagement variables need to be responded to in a suite of optimally effective computer-based tutoring systems for Army use. Also, 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 training systems used by the U.S. Army.

2 Previous Research: Theoretical Grounding

Affect influences learning in at least three ways: memory, attention, and strategy use [16, 18]. Overly strong affect can contribute to cognitive load in working memory, reducing the cognitive resources available to students in learning tasks [13]. Beyond this, negative 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 [20]. These high-intensity negative affective states can be particularly important for trainees learning content that is emotionally affecting or relevant to their future goals. Combat medicine training for soldiers has each of these components; it is relevant to future situations where they or their fellow soldiers may be in physical danger, and the training in vMedic is designed to be realistic and to involve scenarios where soldiers scream in pain, for example.

However, boredom and disengagement are also relevant to trainees engaging in a task that is not immediately relevant, even if it is relevant to a trainee's longer-term goals. Boredom results in several disengaged behaviors, including off-task behavior [8] and gaming the system [5], when a student intentionally misuses the learning software's help or feedback in order to complete materials without learning. Both gaming the system and off-task behavior have been found to be associated with poorer learning in online learning environments [cf. 4].

However, automated systems that infer and respond to differences in student affect can have a positive impact on students, both in terms of improved affect and improved learning [2, 13]. Similarly, automated interventions based on engagement detection can improve both engagement and learning [2].

A key aspect of automated intervention is the need to detect differences in student affect and engagement, in order to intervene effectively. Recent work has detected these constructs, both using sensors [15], and solely from the student's interactions with the learning system [5, 7, 8]. In recent years, sensor-free models have been developed of a range of behaviors associated with engagement or disengagement: gaming the system [3, 4], off-task behavior [3], self-explanation – an engaged behavior [6], carelessness [18], and WTF ("without thinking fastidiously") behavior, actions within a learning system not targeted towards learning or successful performance [24, 34], among other constructs.

Similarly, automated detectors have been developed that can infer affect solely from student interactions with educational software [7, 10, 11]. However, better performance has typically been achieved by systems that infer affect not only from student interactions, but also from information obtained by physiological sensors. These recognition models use a broad range of physical cues ensuing from affective change. Observable physical cues include body and head posture, facial expressions, and posture, and changes in physiological signals such as heart rate, skin conductivity, temperature, and respiration [1]. In particular, galvanic skin response (GSR) has been correlated with cognitive load and stresses [15], frustration [9], and detecting multiple user emotions in an educational game [10].

3 Project Design

The first step towards developing automated detectors of student affect is to obtain "ground truth" training labels of student affect and engagement. Two approaches are typically chosen to obtain these labels: expert human coding, and self-report [11]. In this project, we rely upon expert human coding, as self-report can be intrusive to the processes we want to study, and self-presentation and demand effects are also likely to be of concern with the population being studied (military cadets are unlikely to want to report that they are frustrated or anxious).

These training labels will be collected in a study to be conducted at West Point, the United States Military Academy. Each trainee will use vMedic for one hour in a computer laboratory, in groups of ten at a time. The following sources of student data will be collected: field observations of trainee affect and engagement; the Immersive Tendencies Questionnaire (ITQ), an instrument to gauge an individual's propensity to experience presence in mediated environments a priori to system interaction; the Sense of Presence questionnaire, a 44-item questionnaire that rates subjective levels of presence on four separate factors: (1) Sense of Physical Space (19 items); (2) Engagement (13 items); (3) Ecological Validity/Naturalness (5 items); and (4) Negative Effects (6 items) [19];, a pre-and post test on hemorrhage control (a total of 12 questions, same questions used in pre-and post-test, though ordered differently), and physical sensor data for students as they play the game. The following physical sensors will be used: Q-sensors, and Kinect depth sensors. Q-sensors track skin conductance data, a measure of arousal, while Kinect depth sensors record depth-map images to support recognition of postural positions.

Within this study, expert codes of trainee affect and engagement will be collected by a trained coder (the first author) using the BROMP 1.0 field observation protocol [16]. The field observations will be conducted in a pre-chosen order to balance observation across trainees and avoid bias towards more noteworthy behaviors or affect. Observations will be conducted using quick side glances in order to make it less clear when a specific trainee is being observed. Coding includes recording the first behavior and affect displayed by the trainee within 20 seconds of the observation, choosing from a predetermined coding scheme. The affect to be coded includes: frustration, confusion, engaged concentration [5], boredom, anxiety, and surprise. Affect will be coded according to a holistic coding scheme. Behavior coding includes: on-task behavior, off-task behavior, gaming the system, "psychopath" behavior (friendly fire, killing bystanders), and WTF ("without thinking fastidiously") behavior, where the trainee's actions have no relation to the scenario [17]. In order to be BROMP-certified, a coder must achieve inter-rater reliability of 0.6 or higher to another BROMP-certified coder; two coders are currently trained at Teachers College, and are available for the project.

Field observation coding will be conducted within a handheld Android app, HART, designed for this purpose [7]. The field observations will be synchronized to the other data sources, based on use of an internet time server. Synchronization will be with reference to several data sources, including trainee interactions with vMedic, provided through the GIFT framework's Trainee Module, and physical sensor data on learners, obtained through GIFT's Sensor Module. We anticipate synchronization to involve a skew of 1-2 seconds, based on the time required to enter observations. The GIFT platform includes a synchronization library, which connects to an Internet time-server so that a precise time-stamp can be added to the logs of trainee interactions with vMedic, and the corresponding sensor data. By connecting to the exact same timeserver, the interactions with vMedic, field observations of engagement and affect, and physical sensor data on learners, three data sources can be precisely synchronized.

Automated detectors will be developed using the interaction logs alone, for use when physiological sensors are not available, and using the sensors as well, for situations where they are. A standard approach of conducting feature engineering and then developing classifiers, and validating the classifiers using student-level cross-validation, will be used.

4 Conclusion

The current project has the goal of enhancing the GIFT framework through the creation of models that can infer trainee engagement and affect. This project is expected to both enhance the capacities of the vMedic software, and to provide a model for how this type of detection can be integrated into the GIFT framework more generally. As such, this project is just one small component of the larger efforts that are currently being pursued by the Army Research Lab, to make the GIFT framework a general and extensible platform to achieve the US Army's overall objective of applying learning theory and state-of-the-art learning technology to achieve superior training results for warfighters [14]. We anticipate that this collaborative effort will provide useful information on the future enhancement of the GIFT platform; as such, this project represents a step towards the vision of adaptable and scalable Computer-Based Training Systems helping to enhance the training of U.S. Army military personnel and prepare U.S. soldiers for the conflicts of the future.

Acknowledgments. We thank our research colleagues at the Army Research Lab, Dr. Robert Sottilare, Benjamin Goldberg and Keith Brawner, as well as at North Carolina State University, Dr. James Lester, Dr. Bradford Mott, and Jonathan Rowe. This research is supported by a grant by the Army Research Lab. 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.

5 References

1. Allanson, J., Fairclough, S. H. A research agenda for physiological computing. In Interacting with Computers, 16 (2004), 857–878.
2. Arroyo, I., Ferguson, K., Johns, J., Dragon, T., Meheranian, H., Fisher, D., Barto, A., Mahadevan, S.,Woolf. B. Preparing disengagement with non-invasive interventions. In Proceedings of the 13th International Conference on Artificial Intelligence in Education,.(2007) 195–202.
3. Baker, R.S.J.d. Is Gaming the System State-or-Trait? Educational Data Mining Through the Multi-Contextual Application of a Validated Behavioral Model. In Proceedings of the Workshop on Data Mining for User Modeling at the 11th International Conference on User Modeling 2007, 76-80.
4. Baker, R.S., Corbett, A.T., Koedinger, K.R., Wagner, A.Z. Off-Task Behavior in the Cognitive Tutor Classroom: When Students "Game The System." In Proceedings of ACM CHI 2004: Computer-Human Interaction 2004, 383-390.
5. Baker, R.S.J.d., D'Mello, S.K., Rodrigo, .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, 48 (2010), 223-241.
6. Baker, R.S.J.d., Gowda, S.M., Corbett, A.T. Automatically Detecting a Student's Preparation for Future Learning: Help Use is Key. In Proceedings of the 4th International Conference on Educational Data Mining 2011, 9-188.
7. 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. Sensor-free automated detection of affect in a Cognitive Tutor for Algebra. In Proceedings of the 5th International Conference on Educational Data Mining 2012, 126-133.
8. Baker, R.S.J.d., Moore, G., Wagner, A., Kalka, J., Karabinos, M., Ashe, C., Yaron, D. The Dynamics Between Student Affect and Behavior Occuring Outside of Educational Software. In Proceedings of the 4th bi-annual International Conference on Affective Computing and Intelligent Interaction 2011.
9. Burleson, W. Affective learning companions: strategies for empathic agents with real-time multimodal affect sensing to foster meta-cognitive and meta-affective approaches to learning, motivation, and perseverance. In unpublished Doctoral Dissertation. Cambridge, MA: Massachusetts Institute of Technology (2006).
10. Conati, C. Probabilistic Assessment of User’s Emotions in Educational Games. In Journal of Applied Artificial Intelligence, 16 (2002), 555-575.

11. D’Mello, S.K., Craig, S.D., Witherspoon, A. W., McDaniel, B. T., Graesser, A. C. Automatic Detection of Learner’s Affect from Conversational Cues. In User Modeling and User Adapted Interaction, 18 (2008), 45-80.
12. Goldberg, B., Brawner, K., Sottilare, R, Tarr, R., Billings, D., & Malone, N. Use of Evidence-based Strategies to Enhance the Extensibility of Adaptive Tutoring Technologies. In Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC) 2012.
13. 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.
14. McQuiggan, S., Rowe, J., Lee, S., Lester, J. Story-based Learning: The Impact of Narrative on Learning Experiences and Outcomes. In Proceedings of the Ninth International Conference on Intelligent Tutoring Systems (2008), 530-539.
15. Mohammad, Y., & Nishida, T. Using physiological signals to detect natural interactive behavior. In Spring Science & Business Media, 33(2010) 79-92.
16. 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. EdLab: New York, NY, Manila, Philippines: Ateneo Laboratory for the Learning Sciences (2012).
17. Sabourin, J., Rowe, J., Mott, B., and Lester, J. When Off-Task in On-Task: The Affective Role of Off-Task Behavior in Narrative-Centered Learning Environments. Proceedings of the 15th International Conference on Artificial Intelligence in Education, pp. 534-536, 2011.
18. San Pedro, M.O.C., Rodrigo, M.M., Baker, R.S.J.d. The Relationship between Carelessness and Affect in a Cognitive Tutor. In Proceedings of the 4th bi-annual International Conference on Affective Computing and Intelligent Interaction 2011.
19. Witmer, B.G. & Singer, M.J. Measuring presence in virtual environments: A presence questionnaire. In Presence, 7 (1998), 225-240.
20. Zeidner, M. Test Anxiety: The State of the Art. Springer, Berlin, 1998.

Authors

Jeanine A. DeFalco is a Doctoral Research Fellow in Cognitive Studies at Teachers College, Columbia University. Jeanine’s research interests include embodied cognition and role-play as a methodology for improving analogic reasoning and creative problem solving in both live and simulated learning platforms. A member of Kappa Delta Pi, the international honors society for education, Jeanine has a Masters in Educational Theatre, Colleges and Communities, from the Steinhardt School, New York University, and a Masters in Drama Studies from The Johns Hopkins University. Jeanine’s paper, “Cognition, O’Neill, and the Common Core Standards,” has an expected publication in the *Eugene O’Neill Journal* for Fall 2013, and she will be presenting this same paper at the July 2013 American Alliance for Theatre and Education conference in Bethesda, MD. Other conference presentations include “Drama as an epistemology for pre-service teachers” forW the 2012 National Communication Association conference in Orlando, FL, and “Teaching O’Neill” at the 8th International Eugene O’Neill Conference, 2011, in New York, NY.

Ryan S.J.d. Baker is the Julius and Rosa Sachs Distinguished Lecturer at Teachers College, Columbia University. He earned his Ph.D. in Human-Computer Interaction from Carnegie Mellon University. Baker was previously Assistant Professor of Psychology and the Learning Sciences at Worcester Polytechnic Institute, and he served as the first Technical Director of the Pittsburgh Science of Learning Center DataShop, the largest public repository for data on the interaction between learners and educational software. He is currently serving as the founding President of the International Educational Data Mining Society, and as Associate Editor of the Journal of Educational Data Mining. His research combines educational data mining and quantitative field observation methods in order to better understand how students respond to educational software, and how these responses impact their learning. He studies these issues within intelligent tutors, simulations, multi-user virtual environments, and educational games.

Run-Time Affect Modeling in a Serious Game with the Generalized Intelligent Framework for Tutoring

Jonathan P. Rowe, Eleni V. Lobene, Jennifer L. Sabourin,
Bradford W. Mott, and James C. Lester

Department of Computer Science, North Carolina State University, Raleigh, NC 27695
{jprowe, eleni.lobene, jlrobiso, bwmott, lester}@ncsu.edu

Abstract. Affective computing holds significant promise for fostering engaging educational interactions that produce significant learning gains. Serious games are particularly well suited to promoting engagement and creating authentic contexts for learning scenarios. This paper describes an ongoing collaborative project between the Army Research Lab (ARL), Teachers College Columbia University, and North Carolina State University to investigate generalized run-time affect detection models in a serious game for tactical combat casualty care, vMedic. These models are being developed and integrated with ARL's Generalized Intelligent Framework for Tutoring (GIFT). Drawing upon our experience with GIFT, we outline opportunities for enhancing GIFT's support for developing and studying run-time affect modeling, including extensions that enhance affective survey administration, leverage mathematical models for formative assessment, and streamline affect data processing and analysis.

Keywords: Affect Detection, GIFT, Serious Games.

1 Introduction

The past decade has witnessed major advances in research on computational models of affect, endowing software systems with affect-sensitivity and yielding new insights into artificial and human intelligence [1]. Education and training have served as key application areas for computational models of affect, producing intelligent tutoring systems (ITSs) that can model students' affective states [2], model virtual agents' affective states [3], and detect student motivation and engagement [4]. Education-focused work on affective computing has sought to increase the fidelity with which affective and motivational processes are understood and utilized in ITSs in an effort to increase the effectiveness of tutorial interactions and, ultimately, learning.

The rise of affective computing has coincided with growing interest in digital games for learning. Serious games have emerged as an effective vehicle for learning and training experiences [5]. The education community has developed a broad range of serious games that combine pedagogy and interactive problem solving with the salient properties of games (e.g., feedback, challenge, rewards) to foster motivation and engagement [6–8]. Efforts to design serious games for training have also been the subject of increasing interest in the defense community [6, 9].

A notable property of serious games is their potential to serve as virtual laboratories for studying affect in learning and training applications. Serious games are well suited to promoting high levels of learner engagement and providing immersive training experiences. These features can have significant impacts on learners' affective trajectories, as well as the relationships between learners' affect and performance. For example, in training tasks that evoke considerable stress or anxiety it is plausible that serious games may foster affective experiences that differ considerably from non-mission-critical domains, significantly impacting learners' abilities to successfully demonstrate their knowledge. Salient features such as these raise questions about how to most effectively study and model learner affect during interactions with serious games, as well as questions about how these methods and models can be generalized to other training environments and domains.

In this paper we describe a collaborative project with Teacher's College Columbia University (TC) and the Army Research Lab (ARL) that uses the Generalized Intelligent Framework for Tutoring (GIFT) to investigate run-time affect modeling in a serious game for tactical combat casualty care. The project draws on recent advances in five areas: minimally-obtrusive and synchronize-able field observations of learner affect [10], empirical studies of serious games [7], educational data mining of affect logs [11–12], hardware sensor-based measurements of affect [13], and generalized intelligent tutoring frameworks [14]. The project's objectives are two fold: 1) create modular intelligent tutor components for run-time affect modeling that generalize across multiple training environments and scale to alternate hardware configurations, and 2) develop tools and procedures to facilitate future research on affective computing in learning technologies. This paper focuses on North Carolina State University's component of the project, which emphasizes sensor-based affect detection, and it outlines recommendations for future enhancements to GIFT in support of run-time affect modeling. Specifically, we outline several opportunities for extending GIFT, which include incorporating support for temporal models of affect such as affect transitions; expanding GIFT's survey tools to serve as a centralized repository of validated instruments with an integrated web-based infrastructure for administering surveys; taking advantage of item response theory techniques to conduct stealth, formative assessment of trainee attitudes during learning interactions; and incorporating features to streamline affect data post-processing.

2 Investigating Affect in a Serious Game for Tactical Combat Casualty Care

The goal of our collaboration with ARL and TC is to model trainee affect in a serious game for combat medic training, vMedic, using GIFT. The research team will utilize machine-learning techniques to induce models for detecting trainee's affective states and levels of engagement during interactions with the vMedic software. Affect and engagement significantly influence learning, and we hypothesize that this will be especially true for the vMedic training environment due to the time-sensitive, life-or-

Fig. 9. vMedic serious game for tactical combat casualty care.

death decisions inherent in tactical combat casualty care. In combination with field observations of trainee affect and trace data from the vMedic serious game, the North Carolina State University team will investigate data streams produced by a Microsoft Kinect sensor and Affectiva Q-Sensor to develop and validate affect detection models. The research team seeks to produce models that 1) integrate trace data logs, sensor data, and field observations of trainee emotions; 2) predict emotions accurately and efficiently when hardware sensors are available; and 3) scale gracefully to settings where hardware sensors are unavailable. The models will be developed and utilized to improve trainee engagement and affect when using vMedic, and they will be integrated with interaction-based models devised by colleagues at TC.

The curriculum for the study focuses 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 exercises, and post-tests, are managed entirely by GIFT, which supports inter-module communication through its service-oriented architecture. At the onset of training, learners are presented with direct instruction about tactical combat casualty care in the form of a PowerPoint presentation. After completing the PowerPoint, participants play through a series of scenarios in the vMedic serious game. vMedic presents combat medic scenarios from a first-person perspective (Fig. 1). The learner adopts the role of a combat medic faced with a situation where one (or several) of his fellow soldiers has been seriously injured. The learner is responsible for properly treating and evacuating the casualty. The scenarios include the following elements: a tutorial level for trainees to learn the controls and game mechanics of vMedic; a scenario focusing on a lower leg amputation; a vignette about a patrol that leads to several casualties; and the “Kobayashi Maru” scenario where the trainee cannot save the casualty’s life regardless of her course of medical treatment. vMedic is currently being used at scale by the U.S. Army for combat medic training, and it has been integrated with GIFT by ARL.

The focus of North Carolina State University’s part of the project is leveraging hardware sensor data from a Microsoft Kinect for Windows and Affectiva Q-Sensor to generate affect detection models. Both hardware sensors are integrated with GIFT, enabling the sensor data to be automatically synchronized with vMedic and PowerPoint interaction logs. This architecture removes the need to directly integrate hardware sensors with individual learning environments. Whenever a new training environment is integrated with GIFT, no additional work is required to use the hardware sensors with the new environment.

The Microsoft Kinect provides four data channels: skeleton tracking, face tracking, RGB (i.e., color), and depth. The first two channels leverage built-in tracking algorithms (which are included with the Microsoft Kinect for Windows SDK) for recognizing a user's skeleton, represented as a graph with vertices as joints, and a user's face, represented as a three-dimensional polygonal mesh. The skeleton and face models can move, rotate, and deform based on the user's head movements and facial expressions. 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. The latter two channels produce large quantities of uncompressed image data, so configuration options have been added to GIFT to adjust the sample rate (default is 30 Hz), sample resolution, and compression technique. RGB and depth data can be stored in an uncompressed format, in PNG format with zlib compression, or in PNG format with lz4 compression. We intend to utilize data from the Microsoft Kinect to detect user posture, hand gestures, and facial expression. The Affectiva 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 collects data at 32Hz, and will primarily be used for real-time arousal detection.

Since all technology components in the planned study are managed by GIFT, we have leveraged GIFT's built-in authoring tools to specify the study questionnaires and curriculum tests for assessing trainee knowledge and engagement before and after the learning intervention. We have utilized GIFT's Survey Authoring Tool to rapidly integrate standard presence and intrinsic motivation questionnaires. Additionally, we have used GIFT's sizable repository of reusable content assessment items to create a curriculum test for measuring learning gains across the training sequence.

After specifying the required measures, we used GIFT's Course Authoring Tool to encode the sequence of training and assessment materials that will be presented by GIFT. The Course Authoring Tool includes support for authoring web-based messages that provide instructions to participants, specifying the presentation order of pre- and post-intervention questionnaires and content tests, and specifying the sequence of PowerPoint and vMedic learning activities that occur during the study. When participants take the course, each of these steps is automatically triggered, monitored, and logged by GIFT. It should be noted that authored courses and questionnaires can be easily exported and shared between groups, consistent with GIFT's objective of fostering reusable components.

Currently, our team has established the initial data collection's study procedure, we have tested the integrated hardware sensors, and ensured the reliability of the study's technology setup. In addition to pilot testing field observation tools from TC with GIFT, we are in the process of planning a study at the U.S. Military Academy to investigate cadets' affective experiences during interactions with vMedic.

3 Extending GIFT's Capabilities for Run-Time Affect Modeling

GIFT consists of a suite of software tools, standards, and resources for creating, deploying, and studying modular intelligent tutoring systems with re-usable components. GIFT provides three primary functions: authoring capabilities for constructing learning technologies, instruction that integrates tutorial principles and strategies, and support for evaluating educational tools and frameworks. These capabilities provide the foundation for our investigation of generalizable run-time affect models. This section discusses several areas for which extensions to GIFT could support research and development of generalized affect models in ITSs.

3.1 Detecting and Understanding Learner Affect

While considerable work remains to identify the precise cognitive and affective mechanisms underlying learning, significant progress has been made in identifying the emotions that students commonly experience and how these affect the learning process. For instance, both D'Mello *et al.* [15] and Baker *et al.* [16] have shown that students are most likely to remain in the same emotion state over time and that certain emotional transitions are more likely than others. Students who are experiencing *boredom* are much more likely to experience *frustration* immediately following the state of *boredom* than they are to enter into positive learning states such as *flow* [15–16]. In this way affect transition analyses reveal underlying relationships between affect and learning, which occur generally across intelligent tutoring systems.

Since existing research has suggested that affect transition analysis is both a useful and generalizable tool for investigating learner emotions, incorporating affect transition models within GIFT is a natural direction for future research and development. This will likely raise questions about how to effectively, and generally, integrate affect transition modeling capabilities with each tutor module in the GIFT architecture: the sensor module, learner module, pedagogical module, and domain module. When designing these components, one must consider how these components communicate with one another, and how the system should be configured to support cases where affect-sensitive components are missing. For example, physiological sensors are highly beneficial for affect recognition, but may not be available in all cases. Consequently, a learner model relying on output from such a sensor would need to be adapted, or gracefully deactivated, in a manner that minimizes negative impacts on other modules. Similarly, different genres of serious games have distinct capabilities and affordances. For example, serious games with believable virtual agents may present different opportunities for affective feedback than serious games without virtual agents. Pedagogical modules should possess mechanisms for handling cases where alternate learning environments support different types of interventions.

3.2 Advances in Survey Administration using GIFT

In the educational research community there is a persistent need for streamlined instrument access, validation, and administration. GIFT currently provides a rich collection of content test items and questionnaires that can be re-used across studies and training environments. This survey repository could be expanded to serve the broader research community by systematically adding validated assessment measures and questionnaires used by the education community. GIFT could serve as a searchable, centralized repository of validated instruments, with an integrated web-based infrastructure for administering surveys before and after learning interventions. The instruments could be submitted and listed by category with their important item information such as validity and reliability, links to published papers that describe how the instruments have been used in prior studies, and specific instructions regarding their appropriate use. This type of integrated resource for obtaining, evaluating, and administering questionnaires, content tests, and surveys could help streamline the community's affective computing research efforts, and serve as an entry point for researchers to begin using GIFT. Researchers commonly spend significant effort trying to locate information about study instruments, and GIFT could serve as a tool to facilitate the survey development and selection process. While other domain specific instrument collections are freely available (e.g., www.IPIPori.org [17]), they do not include features for integrating instruments into surveys, or administering surveys to users. GIFT could also reduce the time allocated to integrating surveys with systems such as SurveyMonkey or Qualtrics while encouraging the use of high quality validated instruments.

3.3 Leveraging Mathematical Models for Formative Assessment in GIFT

Building on the availability of this instrument database, there are opportunities to take advantage of item response theory techniques to conduct stealth, formative assessment during learning interactions. Item response theory (IRT) is a mathematical framework for performing measurement in which the variable is continuous in nature while allowing for an individual person and item to be mapped on the same latent trait continuum [18]. An ideal point response process is an IRT approach based on the idea that an individual only endorses an item if he or she is located close to the item on a latent continuum [19]. In other words, if an item is too extreme in either direction, the individual will respond negatively to the item. It can be used with both dichotomous (e.g., content knowledge test) and polytomous data (e.g., Likert-type attitudes or personality [19]). GIFT is well positioned to integrate ideal point methods within user experiences for stealth and ongoing assessment. To date, little research has investigated embedding adaptive, formative assessment within serious games using intermittent item presentation through ideal point methods with a rich database of instruments from which to select.

GIFT offers the opportunity for assessment of both knowledge and attitudinal (e.g., affective states) variables within immersive training experiences. Using GIFT's capabilities, single items can be "transmitted" as part of the story line within a game experience to the participant. GIFT can run mathematical models in the background to determine the best item to present at the next natural point. Conceptually this approach is similar to computerized adaptive tests designed by major test development companies. For example, if the participant responds negatively to the question "I want to repeat this activity over and over," he or she can be presented with an item lower on the latent trait continuum (e.g., "This activity is interesting for now"). GIFT, having access to all of the item information for each potential question, can strategically present a series of them. By the end of the serious game experience, rich data regarding the individuals' location on a latent trait continuum (e.g., engagement) would be available.

3.4 Streamlined Data Processing and Analysis with GIFT

Another opportunity for GIFT to address practical challenges in affective computing research is in post-processing data. Better solutions are needed for merging files and cases and quickly ascertaining basic information from data sets. GIFT could potentially mitigate some of these challenges by introducing standards for data collected during different stages of research; typically data from different stages is encoded in a variety of formats, and a considerable amount of labor is dedicated to data integration after a study has been completed. GIFT could provide a service that automatically links pre, during, and post data for individual participants, thereby reducing labor in data cleaning and transformation steps. GIFT could also be extended to offer quick summary statistics and perform simple operations such as summarizing demographics, computing composite scores for instruments, and providing general summary results. These tools would be especially helpful with affective instruments that often require reverse scoring and other manipulations prior to analysis.

4 Conclusion

This paper has described a collaborative project between the Army Research Lab, Teachers College Columbia University, and North Carolina State University that aims to investigate run-time affect modeling in a serious game for combat medic training, vMedic. In addition to describing this project, we have outlined a number of ways to extend GIFT's capabilities to improve affective computing research for educational applications. We anticipate that these opportunities could increase GIFT's future impact and usage as a tool for ITS researchers.

Acknowledgments. 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.

5 References

1. Calvo, R. A. & D'Mello, S.K.: Affect Detection: An Interdisciplinary Review of Models, Methods, and their Applications. In: IEEE Transactions on Affective Computing, pp. 18–37. (2010)
2. Conati, C., Maclaren, H.: Empirically Building and Evaluating a Probabilistic Model of User Affect. User Modeling and User-Adapted Interaction. 19, 267–303 (2009)
3. Marsella, S.C., Gratch, J.: EMA: A Process Model of Appraisal Dynamics. Cognitive Systems Research. 10, 70–90 (2009)

4. Forbes-Riley, K., Litman, D.: When Does Disengagement Correlate with Performance in Spoken Dialog Computer Tutoring? International Journal of Artificial Intelligence in Education. (2013)
5. Wouters, P., Van Nimwegen, C., Van Oostendorp, H., & Van der Spek, E.D.: A Meta-Analysis of the Cognitive and Motivational Effects of Serious Games. Journal of Educational Psychology. (2013)
6. Johnson, W.L.: Serious Use of a Serious Game for Language Learning. International Journal of Artificial Intelligence in Education. 20, 175–195 (2010)
7. Rowe, J P, Shores, L. R., Mott, B. W., & Lester, J.C.: Integrating Learning, Problem Solving, and Engagement in Narrative-Centered Learning Environments. International Journal of Artificial Intelligence in Education. 21, 166–177 (2011)
8. Halpern, D., Millis, K., Graesser, A.: Operation ARA: A computerized learning game that teaches critical thinking and scientific reasoning. Thinking Skills and Creativity. 7, 93–100 (2012)
9. Kim, J., Hill, R. W., Durlach, P. J., Lane, H. C., Forbell, E., Core, M., Marsella, S.C., et al.: BiLAT: A game-based environment for practicing negotiation in a cultural context. International Journal of Artificial Intelligence in Education. 19, 289–308 (2009)
10. Rodrigo, M.M.T., Baker, R.S.J.d., Agapito, J., Nabos, J., Repalam, M.C., Reyes, S.S., San Pedro, M.C.Z.: The Effects of an Interactive Software Agent on Student Affective Dynamics while Using an Intelligent Tutoring System. In: IEEE Transactions on Affective Computing, pp. 224–236. (2012)
11. Pardos, Z.A., Baker, R.S.J.d., San Pedro, M.O.C.Z., Gowda, S.M., Gowda, S.M.: Affective states and state tests: Investigating how affect throughout the school year predicts end of year learning outcomes. In: Proceedings of the 3rd International Conference on Learning Analytics and Knowledge, pp. 117–124. (2013)
12. Sabourin, J., Mott, B.W., Lester, J.C.: Modeling Learner Affect with Theoretically Grounded Dynamic Bayesian Networks. In: Proceedings of the 4th International Conference on Affective Computing and Intelligent Interaction, pp. 286–295. (2011)
13. Grafsgaard, J.F., Boyer, K.E., Lester, J.C.: Predicting Facial Indicators of Confusion with Hidden Markov Models. In: Proceedings of the 4th International Conference on Affective Computing and Intelligent Interaction, pp. 97–106. (2011)
14. Sottilare, R. A., Goldberg, B. S., Brawner, K. W., & Holden, H.K.: A modular framework to support the authoring and assessment of adaptive computer-based tutoring systems (CBTS). In: Proceedings of the Interservice/Industry Training, Simulation, and Education Conference, (2012)
15. D'Mello, S., Taylor, R.S., Graesser, A.C.: Monitoring Affective Trajectories during Complex Learning. In: Proceedings of the 29th Annual Meeting of the Cognitive Science Society, pp. 203–208. (2007)
16. Baker, R., Rodrigo, M., Xolocotzin, U.: The dynamics of affective transitions in simulation problem-solving environments. In: Proceedings the 2nd International Conference on Affective Computing and Intelligent Interactions, pp. 666–677. (2007)
17. Goldberg, L. R., Johnson, J. A., Eber, H. W., Hogan, R., Ashton, M. C., Cloninger, C. R., & Gough, H.C.: The International Personality Item Pool and the future of public-domain personality measures. Journal of Research in Personality. 40, 84–96 (2006)
18. de Ayala, R.J.: The Theory and Practice of Item Response Theory. Guilford Press, New York, NY (2009).
19. Stark, S., Chernyshenko, O.S., Drasgow, F., Williams, B.A.: Examining Assumptions About Item Responding in Personality Assessment: Should Ideal Point Methods Be Considered for Scale Development and Scoring? Journal of Applied Psychology. 91, 25–39 (2006)

Authors

Jonathan P. Rowe: Jonathan Rowe is a Research Scientist in the Department of Computer Science at North Carolina State University. He received his Ph.D. in Computer Science from North Carolina State University in 2013. His research is in the areas of artificial intelligence and human-computer interaction for advanced learning technologies, with an emphasis on game-based learning environments. He is particularly interested in intelligent tutoring systems, user modeling, educational data mining, and computational models of interactive narrative. Jonathan has led development efforts on several game-based learning projects, including Crystal Island: Lost Investigation, which was nominated for Best Serious Game at the 2012 Unity Awards and the 2012 I/ITSEC Serious Games Showcase and Challenge. His research has also been recognized with several best paper awards, including best paper at the Seventh International Artificial Intelligence and Interactive Digital Entertainment Conference and best paper at the Second International Conference on Intelligent Technologies for Interactive Entertainment.

Eleni V. Lobene: Eleni Lobene is a Research Psychologist in the Center for Educational Informatics at North Carolina State University. She received her Ph.D. in Industrial/Organizational Psychology from North Carolina State University in 2011, where her research focused on K-12 teacher motivations and perceptions. With a background in psychometrics, including classical test theory and item response theory, Dr. Lobene is interested in study design, instrument validation, assessment, and K-16 education. Prior to joining the IntelliMedia group, she worked as a research assistant at the Friday Institute for Educational Innovation, assisting in the assessment of game-based learning environment effectiveness. She has also served as a primary instructor for undergraduate courses in the Department of Psychology at North Carolina State University and provided consulting services to local and global organizations focusing on improving organizational efficiency and implementing data-based change.

Jennifer L. Sabourin: Jennifer Sabourin is currently a Ph.D. student at North Carolina State University in the Department of Computer Science. She received her B.S. (2008) and M.S. (2012) in Computer Science from North Carolina State University where she graduated as Valedictorian. She is a recipient of the National Science Foundation Graduate Research Fellowship award. Since 2007 she has been engaged in research examining affective and metacognitive issues associated with intelligent learning technologies. Her research efforts have resulted in over 30 published journal articles, book chapters, and refereed conference proceedings. Her work has been recognized with a Best Student Paper Award at the International Conference on Affective Computing and Intelligent Interaction and several additional nominations. In addition to her research on intelligent technologies for education, Sabourin has played an active role in efforts to broaden participation in computing and STEM fields through informal learning activities. She developed and led a year-long middle school program for introducing young students to computing principles. This program has served over 250 middle school students in the last six years and has been successfully disseminated to other schools and districts.

Bradford W. Mott: Bradford Mott is a Senior Research Scientist at the Center for Educational Informatics in the College of Engineering at North Carolina State University. He received his Ph.D. in Computer Science from North Carolina State University in 2006, where his research focused on computational models of interactive narrative. His research interests include intelligent game-based learning environments, computer games, and intelligent tutoring systems. Prior to joining North Carolina State University, he worked in the game industry developing cross-platform middleware solutions for the PlayStation 3, Wii, and Xbox 360. In 2000, he co-founded and was VP of Technology at LiveWire Logic where he led development efforts on the RealDialog™ product suite, an automated natural language customer service solution leveraging corpus-based computational linguistics.

James C. Lester: James Lester is Distinguished Professor of Computer Science at North Carolina State University. His research focuses on transforming education with technology-rich learning environments. Utilizing AI, game technologies, and computational linguistics, he designs, develops, fields, and evaluates next-generation learning technologies for K-12 science, literacy, and computer science education. His work on personalized learning ranges from game-based learning environments and intelligent tutoring systems to affective computing, computational models of narrative, and natural language tutorial dialogue. He has served as Program Chair for the International Conference on Intelligent Tutoring Systems, the International Conference on Intelligent User Interfaces, and the International Conference on Foundations of Digital Games, and as Editor-in-Chief of the International Journal of Artificial Intelligence in Education. He has been recognized with a National Science Foundation CAREER Award and several Best Paper Awards.

Toward a Generalized Framework for Intelligent Teaching and Learning Systems: The Argument for a Lightweight Multiagent Architecture

Benjamin D. Nye and Donald M. Morrison

Institute for Intelligent Systems, The University of Memphis, Memphis, Tennessee
{bdnye and dmmrrson}@memphis.edu

Abstract The U.S. Army's Generalized Intelligent Framework for Tutoring (GIFT) is an important step on the path toward a loosely coupled, service-oriented system that would promote shareable modules and could underpin multiagent architectures. However, the current version of the system may be "heavier" than it needs to be and not ideal for new or veteran ITS developers. We begin our critique with a discussion of general principles of multiagent architecture and provide a simple example. We then look at the needs of ITS developers and consider features of a general-purpose framework which would encourage message-driven, multiagent designs, sharing of services, and porting of modules across systems. Next, we discuss features of the GIFT framework that we believe might encourage or discourage adoption by the growing ITS community. We end by offering three recommendations for improvement.

1 Introduction

As the term is used in a seminal paper on the subject, "Is it an agent, or just a program?" (Franklin & Graesser, 1997), an autonomous agent is

> *..a system situated within and a part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to affect what it senses in the future. (p. 25)*

Because a human is also an agent according to this definition, in a sense any intelligent tutoring system may be considered a multiagent system (MAS), designed to support interactions between two agents—the user and the intelligent tutor. However, recent years have seen an increasing interest in the development of systems with multiagent architectures in the more interesting sense that functionality is decentralized across different software agents. In this paradigm, each agent has its own knowledge base (set of beliefs), and carries out different tasks, either autonomously or at the request of other agents. Agent-oriented services build on component-based approaches by giving each component distinct goals that it works to fulfill. As a result, the intelligent behavior of the system as a whole emerges from the collective behavior of the individual agents—including, of course, the human user—allowing for what has been called "autonomous cooperation" (Hülsmann, Scholz-Reiter, Freitag, Wucisk, & De Beer, 2006; Windt, Böse, & Philipp, 2005). For recent examples of ITSs that employ multiagent architectures, see Bittencourt et al., 2007; Chen & Mizoguchi, 2004; El Mokhtar En-Naimi, Amami, Boukachour, Person, & Bertelle, 2012; Lavendelis & Grundspenkis, 2009; and Zouhair et al., 2012). Although these are for the most part prototypes, they serve as useful demonstrations of the general approach.

Multiagent architectures depend on a shared agent communication language (ACL) such as Knowledge Query and Manipulation Language (Finin, Fritzson, McKay, & McEntire, 1994), FIPA-ACL (O'Brien & Nicol, 1998), or JADE (Bellifemine, Caire, Poggi, & Rimassa, 2008), all of which are based on speech act theory (Austin, 1965; Searle, 1969). The ACL, combined with a shared ontology (semantic concepts, relationships and constraints), allows the agents to exchange information, to request the performance of a task, and, in certain cases—such as when one agent requests access to restricted data—to deny such requests (Chaib-draa & Dignum, 2002; Kone, Shimazu, & Nakajima, 2000). A multiagent architecture therefore consists of a distributed "society" of agents (Bittencourt et al., 2007), each with its own agenda, semantically-organized knowledge base, and ability to send and receive messages. The messages take the form of speech acts, including *requests*, *directives*, *assertions*, and so forth. Here is an example:

```
request
:receiver pedagogical agent
:sender NLP agent
:ontology electronics
:content (define, capacitor)
```

where the message is clearly identified as a *request*, the receiver is a pedagogical agent, and the sender is a natural language processing (NLP) agent that translates utterances from human language into messages the pedagogical agent can understand. In this case the pedagogical agent can fulfill the request because it has access to an ontology in the domain of electronics, and "knows" how to extract a definition from it, by following an algorithm or production rule. Here's another example:

```
tell
:receiver pedagogical agent
:sender emotion sensor
:ontology learner affect
:content (learner, confused)
```

where the receiver is again a pedagogical agent, but in this case the sender is an emotion sensing agent reporting its belief that the learner is currently confused. Again, the pedagogical agent can process the contents of the message because it has access to a "learner affect" ontology. As a final example, consider the following:

```
tell
:receiver LMS agent
:sender pedagogical agent
```

```
:ontology learningExperiences
:content (learner, "passed", "helicopter simulation training")
```

where in this case the pedagogical agent is the sender, and the receiver is an LMS agent, which is being told that a certain learner has passed a training course.

These simple examples illustrate several important principles regarding the nature and behavior of multiagent systems. First, note that all three of the software agents are capable of autonomous action, in accordance with their own agendas, and without the need for supervision. The pedagogical agent need not ask the emotion sensor to report its estimate of the learner's affective state. Rather, the emotion sensor reports its beliefs automatically and autonomously, as it does for any agent that has subscribed to its services. Similarly, when it has judged that a learner has passed a course, the pedagogical agent informs the LMS agent, again without having to be asked, simply because the LMS agent has subscribed to its services.

These agents are "lightweight" in the sense that their power lies in their ability to exchange messages with other agents, and to process the contents of these messages based on ontologies that are shared with the agents they exchange messages with, but not necessarily by all of the agents in the system. For example, the NLP agent and pedagogical agent must both have access to the *electronics* ontology, and the LMS agent and pedagogical agent must both share the ontology of *learner experiences*, but neither the emotion sensing agent nor the LMS agent need to know anything about electronics.

Note also that, assuming that the agents' messages are sent over the Internet, all four agents (including the learner) can be at different, arbitrary locations, whether on servers or local devices. Also, any agent can be replaced by any other agent that performs the same function and uses a compatible ACL and associated ontology. If an emotion-sensing agent comes along that does a better job than the original, then, so long as it reads and writes the same kinds of messages and has a compatible ontology (e.g., terms can be translated meaningfully from one ontology to the other), the other agents don't need to be reconfigured in any way. Most importantly, the functionality and value of membership in the society for all participants can increase incrementally, perhaps even dramatically, by registering new agents with new capabilities, or by upgrading the capabilities of the existing members.

Transforming a monolithic ITS legacy system into one with a distributed, multiagent architecture requires two steps: breaking apart existing components into agents and developing ACLs with ITS-tailored ontologies. By encouraging ITS developers to reorganize their systems as services, the Generalized Framework for Intelligent Tutoring (GIFT) provides strong support for this process (Sottilare, Goldberg, Brawner, & Holden, 2012).

2 Criteria for a MAS ITS Framework

Before discussing GIFT specifically, general criteria required for an effective multi-agent ITS framework will be discussed. To understand the criteria for a development framework, one must understand something about the stakeholders involved. In this case, as we are focusing on the software development practices of an ITS, these stakeholders are the research groups that develop these systems. So then, what do such groups look like? A recently completed systematic literature review of papers including the terms “intelligent tutoring system” or “intelligent tutoring systems” found that the majority of ITS research was split between two types: major ITS families (those with 10 or more papers in a 4-year period) and single-publication projects (Nye, 2013). Together, these account for over 70% of ITS research with each accounting for a fairly equal share. This means two things. First, any generalized framework should be able to accommodate major ITS projects that have a large prior investment in tools. Second, it means that such a framework should also embrace contributions from new developers who are often focused heavily on only a single ITS component (e.g., linguistic analysis, assessment of learning, haptic interfaces). So then, an ideal framework would facilitate breaking down legacy architectures into multiagent systems and would also make it easy for one-off developers to add or replace a single component. The framework should also not be locked-in to a single template for the components included in the system: not all systems can be easily broken down into the same components. However, this walks a fine line: too much structure hinders innovative designs, while too little structure offers little advantage over a generic architecture (e.g., off-the-shelf service-composition frameworks).

Accommodating these different ends of the spectrum requires a lightweight and flexible architecture. However, what do we mean by “lightweight?” There are multiple meanings for a “lightweight framework” and most of them are favorable in this context. The following features can be either lightweight or heavy: (1) hardware requirements, (2) software expertise to design services, (3) software expertise to use existing services, (4) software expertise to stand up the message-passing layer between agents, and (5) minimal working message ontology. The first requirement is that the no special hardware or excessive computational overhead should be required to use the framework. The computational requirements should be light, rather than imposing heavy overhead or unnecessary inter-process or remote service calls. Components requiring significant setup or maintenance (e.g., databases, web-servers) should be optional or, at a minimum, streamlined with default setups that work out of the box.

Assuming self-interest, for both types of developers (veterans and newcomers), the cost of designing or redesigning for the framework would need to be exceeded by the benefits. This means minimizing development overhead to create new services or refit old services for the framework. The generalized framework would need to allow easy wrapping or replacement of existing designs, rather than forcing developers to maintain two parallel versions of their ITS. Researchers and developers are unlikely to develop for a framework that requires extensive additional work to integrate with. This means that new developers should need to know only the minimal amount of information about the framework in order to integrate with it. There should be little to no work to create a simple service that can interoperate with the framework and default wrappers should exist for multiple programming languages to parse raw messages into native objects. Such wrappers or premade interfaces would allow even relatively “heavy” communication between agents, while keeping developers from needing to know these protocols.

The framework must also make it easy to take advantage of services that others have implemented, such as through a repository of publically-available services. At minimum, it should be significantly easier to use existing services than it is to add a module to the system. This means that the minimal use case (e.g., the “Hello world” case) for the system should be very simple. For example, a single installed package should make it possible to author (or copy) a single text file configuring the system to create a basic ITS test system. Anything required to run a basic example beyond these requirements indicates a “heavier” setup requirement to begin using the framework. If this part of the framework is heavy, first-time ITS authors would be unlikely to use the framework. Moreover, without such ease-of-use, established ITS developers would be unlikely to rework their code to fit such a framework unless they were compensated for these efforts. In the long term, the success and survival of a general framework for tutoring relies on its ability to contribute back to the ITS community. If researchers and developers benefit by reusing services in the system, they will use it. Otherwise, it will fall into obscurity.

Standing up message passing coordination must be lightweight as well. This means that developers should need to expend minimal effort to invoke a layer capable of exchanging messages between services. As such, this layer should have a strong set of defaults to handle common cases and should work out of the box. Additionally, it should be possible to invoke this layer as part of a standalone application (message passing in a single process) or as a remote web service. Consideration must also be given to mobile devices, as mobile applications have specific limitations with respect to their installation, sandboxing (access to other applications), and data transmission.

Finally, agent communication relies on specific messaging languages codified explicitly or implicitly. Three major paradigms are possible to control this communication. The oldest and most traditional paradigm defines API function interfaces for various types of agents or agent functionality, where “messages” are technically function calls on agents. This approach, however, is fragile and better-suited for synchronous local communication than for asynchronous distributed agents. The second paradigm is to define a centrally-defined ontology of messages, which each having an agreed-upon meaning. The main advantage of this system is that it imposes consistency: all agents can communicate using this predefined ontology. However, agreeing upon a specific ontology of messages is an extremely hard task in practice. This approach is “heavy” from the perspective of learning and being constrained by the ontology. The ultimate goal of a shared and stable ontology for ITS is valuable, but offers formidable pragmatic challenges. The third paradigm allows ad-hoc ontologies of messages. At face value, this approach seems flimsy: the ontology of messages is not defined by the agent communication language and services can define their own messages that may not be meaningful to other services as a result. However, this approach is actually fairly popular in research on agent communication languages (Li & Kokar, 2013) and in recent standards bodies, such as the Tin Can API associated with SCORM (Poltrack, Hruska, Johnson, & Haag, 2012). These approaches standardize the format of messages (e.g., how they are structured) but not the content. Instead, certain recommendations for tags and messages are presented but not required. This approach is lightweight: only a small ontology is required and developers are free to extend it.

Lightweight ad-hoc message ontologies show the most promise for an ITS framework using agent message passing. By standardizing the message format, any two services can syntactically understand any message passed to it. However, it allows developers to choose any set of messages for their agent communication language. While in theory this could lead to a Babylon of disjointed ontologies, in practice developers will typically attempt to use established formats for messages first, if they are available. Much like the original design of a computer keyboard or choice of which side of the road to drive on, the starting ontology for a framework can provide a powerful self-reinforcing norm that guides influences work. As such, it is possible to define a core set of suggested messages that are used by the initial set of agents designed for the framework. Additional messages could then be added to the "common core ontology" of messages when they became common practice among new agents added to the service.

3 The GIFT Framework as a MAS ITS

Given these five characteristics, we now look at how well the GIFT architecture matches them in its current form. First, it must be noted that the intentions of the GIFT project are both ambitious and admirable: without the general shift toward service-oriented design for ITSs there would be little value in discussing multiagent ITSs that build upon service-oriented principles. However, this analysis finds that the current implementation of GIFT appears heavier than would be ideal for the needs and practices of ITS developers. This does not mean that GIFT is a bad architecture, simply that it is an architecture that is geared toward the needs of stakeholders other than existing ITS developers (e.g., end-users, sponsors, etc). A great deal of emphasis is placed on reliability and stability, which is more reflective of enterprise use rather than rapid development. The current GIFT implementation implies a "consume and curate" service model rather than a "collaborative repository" service model. With help from GIFT experts, it is certainly possible to integrate tutoring services with GIFT and deliver this tutoring effectively using the architecture. However, the architecture does not seem light enough to allow researchers to build it into their own toolchain. This section first examines the strengths of GIFT as a generalized framework for developing tutoring systems and then considers limitations that might be addressed by future releases.

By far, the primary advantage over existing systems is its dedication to service-oriented principles and modular design. GIFT is the first serious attempt to develop a platform intended to inject a common suite of tutoring services into a variety of applications, including web applications and 3D games (Sottilare, Goldberg, Brawner, & Holden, 2012). GIFT also has a strong commitment to standards-based communication protocols, supporting the Java Messaging Service (JMS) for service communication. Finally, GIFT was developed in Java so it can be efficiently interpreted on web servers and has strong cross-platform capabilities. The hardware requirements for the core GIFT system are also light. Modern systems should have no trouble running the GIFT services and communication layer. Overall, GIFT appears to be well-optimized for efficient delivery and hosting of tutoring web services.

However, the current GIFT implementation has significant limitations as a development framework for tutoring systems. First, the current implementation does not offer an easy road for standing up a minimal working example using the GIFT framework. Installing and setting up the core framework for use is a multi-step process with multiple stages and some third-party dependencies. Running the framework also requires setting up a dedicated database, which could never be considered a light feature. While some GIFT ITS may benefit from such a database (e.g., those hosting surveys), many prototype ITS might make do with simpler triple-stores, serial data (e.g., delimited text files), or even no persistent data storage. Additionally, setting up the GIFT framework does not differentiate between the core architecture meant to handle communication between services versus the services that are bundled with the architecture. A barebones version might remedy this limitation. Services also communicate using a classical API paradigm, which does not offer much flexibility compared to a more explicit message-passing approach. This means that a developer would need to inspect individual service interfaces to figure out the appropriate accessors. Effectively, this locks developers into an ontology of how services should *act* (i.e., remote API requests) rather than what they should *know* (e.g., generic beliefs or knowledge). While this may seem like a subtle difference, a service that only needs to broadcast its knowledge can sidestep designing who receives that information and how it should be used. Finally, GIFT lacks service stubs or wrappers in common languages (e.g., Java, Python, C#) that would make it easy to develop a service that conforms to the framework.

Overall, deploying the GIFT architecture and attempting to develop a new service for the system are both heavy tasks rather than lightweight ones. Without support from the GIFT project, this would make developing for the framework quite costly. The software expertise to design services is heavy, since there are few tools to make this process easier. Despite using a service-oriented paradigm, the system does not offer a suite of example services or stub service in common programming languages. Unless developers have expertise in Java and can carefully inspect the available API, they would not be able to integrate a new service into GIFT. The software expertise to use existing services is also heavy. The minimal use-case example currently installs all GIFT services and requires a database. Services are not handled using a repository or package manager approach, but are simply installed with no streamlined method to manage them. Since there is no way to install the service communication layer as a standalone system, the software expertise to stand up any message-passing layer between agents is also heavy. Finally, no message ontology is available because the system messages are invoked to carry API calls between services. While ontologies for GIFT have been discussed, these ontologies are focused on the types of services in the system rather than the types of messages employed (Sottilare, 2012). This forces communication between services to revolve around the API of services rather than on the information they are passing.

In its current form, the GIFT framework would not be well-suited for a multiagent system ITS. It also does not support many of the aspects of such a framework that would aid either of the major classes of ITS developers to base their projects on GIFT. A one-off innovation, such as a PhD candidate's thesis project, would likely be significantly burdened by the effort to stand up the system without help and would need to learn the API for existing services before they could be useful. A large group focusing on an established ITS architecture would be limited by these factors and also by the lack of interfaces and supporting tools for the programming languages used by their legacy projects. Most importantly, since services do not communicate using a more general agent communication language, significant effort will likely be required to tailor communication to the specific API function interfaces. Without the ability to specify a common message ontology for the agent communication language, it would be impractical to develop a multiagent tutoring system using GIFT. Traditional API's based on interfaces are not well-suited to this task, as they conflate process names with the meanings of the data they produce. Traditional API functions are also poorly suited for dynamic function binding and other advanced patterns that could be used by message-passing agents.

4 Discussion and Recommendations

This analysis has explored the potential benefits and requirements related to building an intelligent tutoring system based on multiagent architecture principles and an agent communication language. These requirements were then compared with the GIFT framework's current capabilities. Our finding is that the current implementation of GIFT is not currently well-suited to these advanced design patterns. While hardware requirements are low, software expertise to design new GIFT services and to use the existing GIFT services is fairly high. Additionally, message system of GIFT currently reflects an API pattern with heavy reliance on knowing the other services in the framework. This is unfortunate, as lighter publish-and-subscribe patterns have become increasingly popular in the industry due to their adaptability (Jokela, Zahemszky, Rothenberg, Arianfar, & Nikander, 2009). This said, GIFT represents a project that is far closer to these patterns than any prior ITS project. GIFT has also spurred discussion on patterns for service-oriented tutoring that were not previously at the forefront of ITS design.

Based on this analysis of GIFT, some design recommendations are indicated for future iterations. From the perspective of developing tutoring agents, the first major recommendation is to center communication of services around explicit message passing where agents publish their knowledge using speech acts. To support this goal, feedback should be gathered from major ITS research groups to propose messages for an initial ontology of recommended messages that determine the information passed between components of the system. To add services to the GIFT framework, developers should only need to know this ontology of messages so they can use it or extend it accordingly. Services should not need to know who their messages are received by, only what messages they receive, what messages they produce, and when they wish to produce a message.

The second major recommendation is the need to separate the GIFT services from the GIFT communication layer. If GIFT is truly a general framework, it must ultimately provide a specialized communication layer as its core. Other services should be treated as plug-ins that can be installed or removed using a package-management approach. This includes the core GIFT services that are bundled with the system. Separating the services from the core architecture would greatly simply the ability to provide a minimal working example and would make the system more flexible overall. As the system itself appears to be designed with such boundaries in mind, this should primarily be a matter of how setup packages are structured and installed.

Related to this issue, a very basic installation that works “out of the box” must be available for developers to start working with GIFT.

The third major recommendation is that GIFT should provide small suites of utilities, wrappers, and stubs to help develop services using a variety of common languages. A generalized system must not assume that developers will convert their code to Java or build their own communication wrappers for their native language. While the use of remote procedure API calls has sidestepped this issue slightly, it has not completely removed it. Additionally, a more flexible message-passing paradigm would require such supporting tools to an even greater extent.

Finally, in the present analysis we have focused only on issues of system architecture, as is proper given that GIFT intends to serve as a general purpose framework, not a stand-alone ITS. However, in so doing we have arguably paid insufficient attention to other important issues that GIFT approaches, such as the need for shareable domain models, learner models, and instructional modules. As the developers of GIFT have pointed out, legacy ITSs tend to be built as “unique, one-of-a-kind, largely domain-dependent solutions focused on a single pedagogical strategy” (Sottilare, Brawner, Goldberg & Holden, 2012:1). After some four decades of independent effort, a case can be made that the time has come for a much greater degree of collaboration and sharing among members of the ITS community, including both veterans and newcomers. This means not just the sharing of ideas, but of working software objects and structures. The development of a lightweight, multiagent architecture that supports “autonomous cooperation” among communities of distributed software agents united by an emergent common language offers a first step in the process, but it is by no means the last.

5 References

1. Austin, J. L.: How to do things with words. Oxford University Press, New York (1965)
2. Bellifemine, F., Caire, G., Poggi, A., Rimassa, G.: JADE: A software framework for developing multi-agent applications. Lessons learned, Information and Software Technology, 50(1), 10–21 (2008)
3. Bittencourt, I. I., de Barros Costa, E., Almeida, H. D., Fonseca, B., Maia, G., Calado, I., Silva, A. D.: Towards an ontology-based framework for building multiagent intelligent tutoring systems. In: Simpósio Brasileiro de Engenharia De Software. Workshop on Software Engineering for Agent-oriented Systems, III, João Pessoa, 2007. Proceedings of the Porto Alegre, SBC, pp. 53–64 (2007)
4. Chaib-draa, B., Dignum, F.: Trends in agent communication language. Computational Intelligence, 18(2), 89–101 (2002)
5. Chen, W., Mizoguchi, R.: Learner model ontology and learner model agent. Cognitive Support for Learning-Imagining the Unknown, 189–200 (2004)
6. El Mokhtar En-Naimi, A. Z., Amami, B., Boukachour, H., Person, P., Bertelle, C.: Intelligent Tutoring Systems Based on the Multi-Agent Systems (ITS-MAS): The Dynamic and Incremental Case-Based Reasoning (DICBR) Paradigm. IJCSI International Journal of Computer Science Issues 9(6), 112–121 (2012)

7. Finin, T., Fritzson, R., McKay, D., McEntire, R.: KQML as an agent communication language. In: Proceedings of the third international conference on Information and knowledge management, pp. 456–463. ACM (1994, November)
8. Franklin, S., Graesser, A.: Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents. In: Intelligent agents III agent theories, architectures, and languages, pp. 21–35. Springer Berlin Heidelberg (1997)
9. Hülsmann, M., Scholz-Reiter, B., Freitag, M., Wucisk, C., De Beer, C.: Autonomous cooperation as a method to cope with complexity and dynamics?–A simulation based analyses and measurement concept approach. In: Y. Bar-Yam (ed.), Proceedings of the International Conference on Complex Systems (ICCS 2006), vol. 2006. Boston, MA, USA (2006)
10. Jokela, P., Zahemszky, A., Rothenberg, C., Arianfar, S., Nikander, P.: LIPSIN: Line speed publish/subscribe inter-networking. In: ACM SIGCOMM Computer Communication Review, 39(4), pp. 195–206. ACM Press (2009, August)
11. Kone, M. T., Shimazu, A., Nakajima, T.: The state of the art in agent communication languages. Knowledge and Information Systems, 2(3), 259–284 (2000)
12. Lavendelis, E., Grundspenkis, J.: Design of multi-agent based intelligent tutoring systems. Scientific Journal of Riga Technical University. Computer Sciences, 38(38), 48–59 (2009)
13. Li, S., Kokar, M. M.: Agent Communication Language. In: Flexible Adaptation in Cognitive Radios, pp. 37–44. Springer New York (2013)
14. Nye, B. D.: ITS and the Digital Divide: Trends, Challenges, and Opportunities. In: Artificial Intelligence in Education (In Press).
15. O'Brien, P. D., Nicol, R. C.: FIPA–towards a standard for software agents. BT Technology Journal, 16(3), 51–59 (1998)
16. Poltrack, J., Hruska, N., Johnson, A., Haag, J.: The Next Generation of SCORM: Innovation for the Global Force. In: The Interservice/Industry Training, Simulation & Education Conference (I/ITSEC), vol. 2012, no. 1. National Training Systems Association (2012, January)
17. Searle, J. R.: Speech acts: An essay in the philosophy of language. Cambridge University Press (1969)
18. Sottilare, R. A.: Making a case for machine perception of trainee affect to aid learning and performance in embedded virtual simulations. In: Proceedings of the NATO HFM-169 Research Workshop on the Human Dimensions of Embedded Virtual Simulation. Orlando, Florida (2009, October)
19. Sottilare, R. A.: 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)
20. Sottilare, R. A., Goldberg, B. S., Brawner, K. W., Holden, H. K.: 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 (2012, January)
21. Windt, K., Böse, F., Philipp, T.: Criteria and application of autonomous cooperating logistic processes. In: Gao, J.X., Baxter, D.I., Sackett, P.J. (eds.) Proceedings of the 3rd International Conference on Manufacturing Research. Advances in Manufacturing Technology and Management (2005)
22. Zouhair, A., En-Naimi, E. M., Amami, B., Boukachour, H., Person, P., Bertelle, C.: Intelligent tutoring systems founded on the multi-agent incremental dynamic case based reasoning. In: Information Science and Technology (CIST), 2012 Colloquium, pp. 74–79. IEEE (2012, October)

Authors

Benjamin D. Nye is a post-doctoral fellow at the University of Memphis, working on tutoring systems architectures as part of the ONR STEM Grand Challenge. Ben received his Ph.D. from the University of Pennsylvania and is interested in ITS architectures, educational technology for development, and cognitive agents.

Dr. Chip Morrison is a Faculty Affiliate at IIS. A graduate of Dartmouth, Dr. Morrison holds an M.A. from the University of Hong Kong and an Ed.D. from Harvard. His current research interests include models of human cognition and learning, and the application of these models to conversation-based intelligent learning systems.

Recommendations For The Generalized Intelligent Framework for Tutoring Based On The Development Of The DeepTutor Tutoring Service

VASILE RUS, NOBAL NIRAULA, MIHAI LINTEAN, RAJENDRA BANJADE, DAN STEFANESCU, WILLIAM BAGGETT
The University of Memphis
Department of Computer Science/Institute for Intelligent Systems
Memphis, TN 38138
vrus@memphis.edu

Abstract. We present in this paper the design of DeepTutor, the first dialogue-based intelligent tutoring system based on Learning Progressions, and its implications for developing the Generalized Framework for Intelligent Tutoring. We also present the design of SEMILAR, a semantic similarity toolkit, that helps researchers investigate and author semantic similarity models for evaluating natural language student inputs in conversatioanl ITSs. DeepTutor has been developed as a web service while SEMILAR is a Java library. Based on our experience with developing DeepTutor and SEMILAR, we contrast three different models for developing a standardized architecture for intelligent tutoring systems: (1) a single-entry web service coupled with XML protocols for queries and data, (2) a bundle of web services, and (3) library-API. Based on the analysis of the three models, recommendations are provided.

Keywords: intelligent tutoring systems, computer based tutors, dialogue systems

1 Introduction

The General Framework for Intelligent Tutoring (GIFT; Sottilare et al, 2012) aims at creating a modular ITS/CBTS (intelligent tutoring systems/computer-based tutoring systems) framework and standards to foster "reuse, support authoring and optimization of CBTS strategies for learning, and lower the cost and skillset needed for users to adopt CBTS solutions for military training and education." GIFT has three primary functions: (1) to help with developing components for CBTS and whole tutoring systems; (2) to provide an instructional manager that integrates effective and exploratory tutoring principles and strategies for use in CBTS; and (3) to provide an experimental test bed to analyze the effectiveness and impact of CBTS components, tools, and methods. That is, GIFT is both a software environment and standardization effort. The availability of a GIFT software package suggests that for now the software environment has been given priority to standardization efforts. This paper intends to help make progress towards a GIFT standardization.

To that end, we present the design of DeepTutor (www.deeptutor.org; Rus et al., to appear), the first CBTS based on the emerging framework of Learning Progressions proposed by the science education research community (LPs; Corcoran, Mosher, & Rogat, 2009). LPs can be viewed as incrementally more sophisticated ways to think about an idea that emerge naturally while students move toward expert-level understanding of the idea (Duschl et al., 2007). That is, LPs capture the natural sequence of mental models and mental model shifts students go through while mastering a topic. It is this learner-centric view that differentiates LPs from previous attempts to reform science education. The LPs framework provides a promising way to organize and align content, instruction, and assessment strategies in order to give students the opportunity to develop deep and integrated understanding of science ideas.

DeepTutor is developed as a web service and a first prototype is fully accessible through a browser from any Internet-connected device, including regular desktop computers and mobile devices such as tablets. As of this writing, DeepTutor is designed as a bundle of two web services: (1) the tutoring service itself accessed by learners, and (2) the support service which includes everything else: authoring and content management, experiment management, user management, and instruction management. The latter service is viewed as a single service because there is a single-entry point to access all these functions. The tutoring service exports its functionality through an XML-based protocol. Third party developers can use their own development environments to design custom DeepTutor clients and integrate them with the DeepTutor tutoring service; all they need is to understand and generate an XML-like protocol, which is a query-language for accessing DeepTutor functionality.

We contrast the DeepTutor design with the design of another software environment, SEMILAR (www.semanticsimilarity.org; Rus et al., 2013). SEMILAR can be used to author semantic similarity methods for semantic processing tasks such as the task of assessing students' natural language inputs in dialogue-based CBTSs. SEMILAR, a SEMantic simILARity toolkit, has been designed as a Java library. Access to SEMILAR functionality is already available through a Java API (Application Programming Interface). Users can use the semantic similarity methods in SEMILAR as long as they link the SEMILAR library to their own Java programs. If a developer were to use SEMILAR from non-Java applications, a solution would be for the SEMILAR library to export its functionality through an XML-like protocol which is easily readable from any programming language. This latter integration solution is basically the export of functionality approach available in the DeepTutor tutoring service. SEMILAR has not been developed as a web service because it was initially developed for our own internal use. We have plans to make it available as a web service in the future. A GUI-based Java application has been developed and is currently tested to offer non-programmers easy access to the SEMILAR functionality.

The two designs, DeepTutor and SEMILAR, will help us discuss concretely three models for standardizing and implementing CBTS functionality to meet GIFT's goals: (1) a single-entry web service, e.g. the two DeepTutor services can be collated into one service (a one-stop-shop model); (2) a bundle of web services – the current DeepTutor design in which different functionality is accessed through different service points, and (3) a library of components accessed through an API. The three models share the common requirement of standardizing the communication between a client/user and provider of tutoring components/functions. While all three models have advantages and disadvantages, we favor the web services models for a Generalized Framework for Intelligent Tutoring as these models better suit the emerging world of mobile computing in which users access services in the cloud over the network as opposed to downloading full applications on their local, energy-sensitive mobile devices. Furthermore, the combination of a tutoring service and XML-based protocols for data and commands/queries fits very well with recent standards for representing knowledge proposed by the Semantic Web community, standards for authoring behavior of dialogue systems (see the FLORENCE dialogue manager framework; Fabbrizio & Lewis, 2004), or previous work in the intelligent tutoring community (see CircSim's mark-up language; Freedman et al., 1998).

The rest of the paper is organized as in the followings. The next section provides an overview of the DeepTutor web service. Then, we describe the design of the SEMILAR library. We conclude the paper with Discussion and Conclusions in which we make recommendations for GIFT based on the three models we discussed.

2 The Intelligent Tutoring Web Service DeepTutor

DeepTutor is a conversational ITS that is intended to increase the effectiveness of conversational ITSs beyond the interactivity plateau (VanLehn, 2011) by promoting deep learning of complex science topics through a combination of advanced domain modeling methods (based on LPs), deep language and discourse processing algorithms, and advanced tutorial strategies. DeepTutor currently targets the domain of conceptual Newtonian Physics but it is designed with scalability in mind (cross-topic, cross-domain).

DeepTutor is a problem solving coaching tutor. DeepTutor challenges students to solve problems, called tasks, and scaffolds their deep understanding of complex scientific topics through constructivist dialogue and other elements, e.g. multimedia items. DeepTutor uses the framework of Learning Progressions (LPs) to drive its scaffolding at macro- and micro-level (Rus et al, to appear). There is an interesting interplay among assessment, LPs, instructional tasks, and advanced tutoring strategies that is finely orchestrated by DeepTutor. The LPs are aligned with an initial, pre-tutoring assessment instrument (i.e., pretest) which students must complete before interacting with the system. Based on this first summative assessment, an initial map of students' knowledge level with respect to a topic LP is generated. The LPs encode both knowledge about the domain and knowledge about students' thinking in the form of models that students use to reason about the domain. The student models vary from naïve to weak to strong/mastery models. For each level of understanding in the LP a set of instructional tasks are triggered that are deemed to best help students make progress towards mastery, which coincides with the highest level of understanding modeled by the LP.

The task representation is completely separated from the executable code and therefore DeepTutor is compliant with the principles adopted by GIFT from Patil and Abraham (2010). Also, in accordance with GIFT principles (Sottilare et al., 2012), DeepTutor's pedagogical module interacts with the learner module (the Student) and adapts the scaffolding tasks and dialogue according to the learner's level of knowledge.

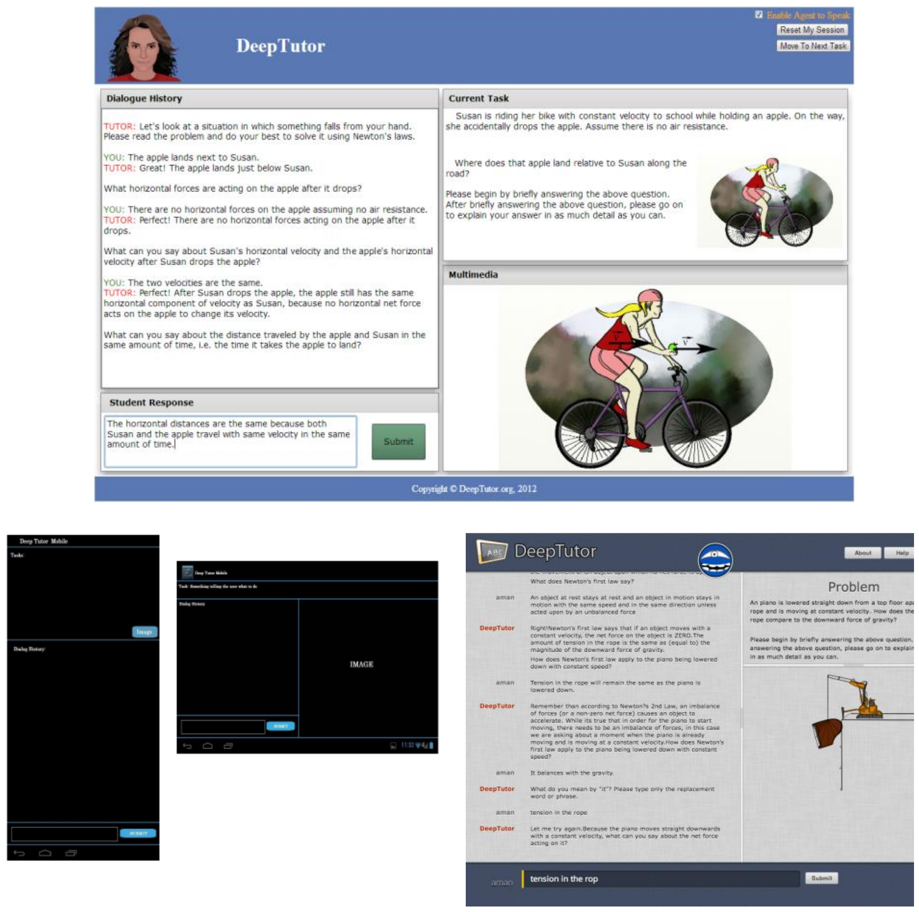
DeepTutor is an ongoing project. As of this writing, different modules are at different stages of maturity. For instance, our LP has been empirically validated based on data collected from 444 high-school student responses. Other components, e.g. the general knowledge module that can handle tasks related to general knowledge such as answering definitional questions ("What does *articulate* mean?"), is still in the works. The system as a whole will be fully validated in the next 6-12 months.

As already mentioned, DeepTutor has been designed as a web service accessible via HTML5-compatible clients, typically web browsers. The familiarity of users with web browsers and eliminating the need to install software packages (except the web browser) on each user's own computer environment makes it extremely convenient for users to access DeepTutor from any Internet-connected device and at the same time opens up unprecedented economies of scale for tutoring research. For instance, during Spring 2013 DeepTutor has been successfully used by more than 300 high-school students[7] from their Internet-device of choice (outside of traditional classroom instruction or experimental lab): home computer, tablet, mobile phones, or library computer.

All communication between the client and the DeepTutor server is handled through an XML-like protocol. The protocol specifies both commands and data that both client and server can interpret. The client communicates user actions and data to the server and the server replies with appropriate responses. Currently, the responses are in the form of display commands and values for various tutoring elements that are visible to the user on screen. That is, the client simply uses the information to update the corresponding interface elements, e.g. the client needs to update the dialogue history box with the most recent DeepTutor feedback response. The protocol contains sufficient information for learner software clients to display the elements of the standard DeepTutor interface. At the same time, the client uses the XML protocol to send the DeepTutor server important information about the user, e.g. user actions such as turning the talking head off, typed responses, time stamps, etc.

There are two major phases for learner clients to connect to the full DeepTutor system: the user authentication and initialization phase and the tutoring phase. In the authentication and initialization phase the user authenticates herself. A set of initialization parameters are sent to the DeepTutor system as well. Currently, the initialization parameters are set from the instructor view of the system, e.g. the researcher/experimenter or instructor/teacher can set a particular instructional strategy to be used by the system for a particular user or groups of learners. We can imagine in the future that these parameters are set dynamically based on the student model retrieved from a persistent database of learner information.

[7] This group of students is different from the 444 student group used for validating the LP.



TUTOR: Let's look at a situation in which something falls from your hand. Please read the problem and do your best to solve it using Newton's laws.

YOU: The apple lands next to Susan.
TUTOR: Great! The apple lands just below Susan.

What horizontal forces are acting on the apple after it drops?

YOU: There are no horizontal forces on the apple assuming no air resistance.
TUTOR: Perfect! There are no horizontal forces acting on the apple after it drops.

What can you say about Susan's horizontal velocity and the apple's horizontal velocity after Susan drops the apple?

YOU: The two velocities are the same.
TUTOR: Perfect! After Susan drops the apple, the apple still has the same horizontal component of velocity as Susan, because no horizontal net force acts on the apple to change its velocity.

What can you say about the distance traveled by the apple and Susan in the same amount of time, i.e. the time it takes the apple to land?

Susan is riding her bike with constant velocity to school while holding an apple. On the way, she accidentally drops the apple. Assume there is no air resistance.

Where does that apple land relative to Susan along the road?

Please begin by briefly answering the above question. After briefly answering the above question, please go on to explain your answer in as much detail as you can.

The horizontal distances are the same because both Susan and the apple travel with same velocity in the same amount of time.

What does Newton's first law say?

aman An object at rest stays at rest and an object in motion stays in motion with the same speed and in the same direction unless acted upon by an unbalanced force

DeepTutor Right!Newton's first law says that if an object moves with a constant velocity, the net force on the object is ZERO.The amount of tension in the rope is the same as (equal to) the magnitude of the downward force of gravity.
How does Newton's first law apply to the piano being lowered down with constant speed?

aman Tension in the rope will remain the same as the piano is lowered down.

DeepTutor Remember than according to Newton?s 2nd Law, an imbalance of forces (or a non-zero net force) causes an object to accelerate. While its true that in order for the piano to start moving, there needs to be an imbalance of forces, in this case we are asking about a moment when the piano is already moving and is moving at a constant velocity.How does Newton's first law apply to the piano being lowered down with constant speed?

aman It balances with the gravity.

DeepTutor What do you mean by "it"? Please type only the replacement word or phrase.

aman tension in the rope

DeepTutor Let me try again.Because the piano moves straight downwards with a constant velocity, what can you say about the net force acting on it?

aman tension in the rop

An piano is lowered straight down from a top floor ap rope and is moving at constant velocity. How does the rope compare to the downward force of gravity?

Please begin by briefly answering the above question. answering the above question, please go on to explain in as much detail as you can.

Figure 10. Three DeepTutor clients showing three different renderings of the learner-view of the DeepTutor Service: the currently official learner view in DeepTutor (top), an under-development Android app (bottom left) and a client developed for a Masters project (bottom right).

Client applications that access the full DeepTutor tutoring system (not individual components) can be designed quite easily. The main reason is the relatively simple but efficient current interface that allows the learner to focus on the interactive tutorial dialogue. Figure 1 bottom shows on the left-hand side an Android-based app client for DeepTutor designed by a small team of 5 Computer Science undergraduate students as a semester-long class project. The app has an interface design for a vertical versus horizontal positioning of the mobile device. The right-hand side of Figure 1 includes another DeepTutor client designed by a Masters student in Computer Science as his Masters project on Human-Computer Interaction.

It should be noted that more complex learner views are in the plans for DeepTutor. For instance, we plan to add several supplemental instructional aids and monitoring and informing elements such as how many tasks are left to cover in the current session or game-like features such as showing what percentage of a learner's peers successfully finished the current task. The current interface of DeepTutor is as simple as it can be and it was intentionally kept this way. The goal was to reduce the number of on-screen distractors in order for the learner to focus on the tutorial dialogue. Adding more elements would make the interface richer which could distract the learners from the main tutorial interaction. It would be an interesting topic to investigate though.

We imagine that other users, e.g. developer of tutoring systems, may need to access specific functionality/components of DeepTutor according to the GIFT goals. As an example, we can imagine someone willing to access the output of the assessment module. As of this writing, the client-server protocol does not allow export of specific functionality. To allow export of functionality at a finer-grain level the current DeepTutor XML protocol must be extended such that the server provides developers/researcher clients output from specific modules, e.g. the assessment module. The exact format of the query and response must be clearly defined.

We believe that efforts to standardize access to GIFT-defined CBTS modules using XML protocols are best. The specification of these protocols needs to be done at different levels of abstractness such that the protocol is general enough to be applicable to all types of tutoring systems (at higher, more general levels of specification) and detailed enough for specific types of tutoring systems to be readily implementable by various groups. For instance, a general specification for querying the assessment module would include a general query element that indicates that an user input is needed together with a context variable which may contain other useful information for best assessing the student input (the context variable could be as simple as an user identifier and a session identifier or much more complex including a comprehensive list of factors that might impact assessment) and the format of the response from the assessment component of the tutoring service. This general specification can be further specified for *benchmark-based tutoring systems* (AutoTutor – Graesser et al., 2005, Guru – Olney et al. 2012; DeepTutor – Rus et al., to appear) as well as for *reasoning-based tutoring systems* (Why-Atlas; VanLehn et al., 2007). We use this broad categorization of tutoring systems to help us illustrate the need for further specifying general query formats. A *benchmark-tutoring system* is one that requires an expert-generated or benchmark response against which the student response is assessed (DeepTutor is such a system; Rus et al., to appear). For benchmark-tutoring systems the assessment query will need to pass (a pointer to) the benchmark response as one of the input parameters. *Reasoning-based systems* are able to infer the correct response automatically (Why-Atlas; VanLehn et al., 2007). For reasoning-based systems the benchmark response may not be needed but instead (a pointer to) a knowledge base.

In summary, a web service together with XML-based protocols may offer the best option for moving forward in GIFT. The advantage of using a web service solution with an XML-based protocol has the advantage of being easily extendable (new functionality can be added by simple adding new tags in the XML protocol). Another advantage is the decoupling the logical view from the actual implementation. The decoupling of functionality from actual implementation can be very useful. For example, the XML protocol can offer a GIFT-like view of the system with components so defined to meet GIFT standards while the actual, back-end implementation can be so designed to best fit particular types of ITSs. Sometimes refactoring and exporting functionality is conceptually challenging as for some tutoring systems there is a tight connection between components that GIFT suggest be separate. For instance, in LP-based ITSs such as DeepTutor, there is a tight relationship between learner models and the domain model because the domain is organized from a learner perspective (Rus et al., in press). Separating the learner model from the domain model is conceptually challenging and probably not recommended. The decoupling of functionality allows keeping the best implementation while offering differing views recommended by standards.

The combination of web service/XML protocol is also more advantageous when it comes to updates and extensions. There is no need to download and recompile a client application with the latest version of a component or the whole tutoring system.

We conclude this section by noting that the service model can further be refined into two types of service-based models: single service versus bundle of services. The current DeepTutor system is a **bundle of services**. In this model, the functionality of the various modules would be available as separate web services, e.g. the assessment module could be a separate web service. There are some interesting aspects of the **bundle of services model**. For instance, in DeepTutor some functionality is offered through a combination of the two DeepTutor services: debugging capabilities are offered through a combination of the tutoring and support services. That is, a developer polishing various components has to use both services.

All services can eventually be bundled together in a single, deep service (containing many subservices) in which case we have a **single-entry service model**. This model implements the concept of a one-stop-shop meaning users will use on access point for the components or the whole tutoring system.

3 The SEMILAR Library For Assessing Natural Language Student Inputs

Our SEMILAR (SEMantic similarity) toolkit, includes implementations and extensions of a number of algorithms proposed over the last decade to address the general problem of semantic similarity. SEMILAR includes algorithms based on Latent Semantic Analysis (LSA; Landauer et al., 2007), Latent Dirichlet Allocation (Blei, Ng, & Jordan, 2003), and lexico-syntactic optimization methods with negation handling (Rus & Lintean, 2012a; Rus et al., 2012b); Rus et al, in press). Due to space reasons, we do not present the set of methods available but rather discuss the design of SEMILAR as a Java library and its implications for using an akin design for GIFT.

The Java library design for SEMILAR has the advantage of being easily integrated as compiled code into Java applications which, at least in theory, should be platform independent. However, users have to download the whole package, install it, and then compile it with their tutoring systems. If these systems or components are written in a programming language different from Java, extra effort will be needed for integration. We call this **the library-API model** for a GIFT framework. Indeed, a GIFT framework based on the library-API model will require downloading and installing large software packages on various platforms by users of various technical backgrounds which may make the whole effort more challenging. For instance, the SEMILAR library and application is 300MB large (it includes large models for syntactic parsing among other things). SEMILAR can be regarded as a tutoring component for assessing students' natural language inputs. If ITS developers were to use SEMILAR as a library they have to download it and integrate it in their products. They have to install and update the API when updates become available. In fact, this is how SEMILAR is currently integrated in DeepTutor. Changes in implementation, e.g. bug fixes, would require a new download and reintegration of the systems that rely on the library. When SEMILAR will be available as a web service, all is needed is understanding the API, in the form of an XML-based communication protocol, and connect to the tutoring service. The need for a network connection are a potential risk for the service model in the form of network congestion which may make the service inaccessible or slow at times.

4 Discussion and Conclusions

We presented three models based on our experience with implementing a set of coherent functionalities related to intelligent tutoring systems and semantic processing. Each of the models has its own advantages and disadvantages. Ideally, all three models should be adopted by GIFT. However, if it were to choose we believe that the service-based models are the best solution for an emerging world of mobile devices in which accessing software services in the cloud is becoming the norm. The library-API and web service solutions are functionally equivalent with the former presenting more technical challenges for users with diverse backgrounds and computing environments and also being less suitable for a mobile computing world.

One apparent downside of the web service model is that potential developers cannot alter the code themselves in order to conduct research. This is just an apparent downside as a quick fix would be for each component to offer enough parameters, in the form of a switchboard, to allow potential users to alter behavior without the need to change the code. In fact, this solution should be preferred as users would not need to spend time to understand and alter the code, a tedious and error-prone activity.

Standardization efforts for XML-based protocols may start with previous efforts where available. For instance, the dialogue processing community has made attempts to standardize dialogue acts/speech acts, a major component in dialogue-based ITSs, for more than a decade. The resulting Dialogue Act Mark-Up in Several Layers (DAMSL) XML schema can be used as a start to standardize speech acts in dialogue ITSs.

In summary, we favor a **one-stop-shop service** model with **switchboard**-like facilities for implementing GIFT. Table 1 below illustrates the pros and cons of the three models discussed in this paper.

Table 6. Comparison of the three proposed model: single-entry service, bundle of services, and library.

	One-Stop-Shop/Single-Entry Service	**Bundle of Services**	**Library**
Programming Language Independent	YES	YES	NO
Install and update on local machine/ environment	NO	NO	YES
Fit for emerging mobile and cloud-computing fitness	EXCELLENT	EXCELLENT	POOR
Customization	VERY GOOD	VERY GOOD	EXCELLENT
Cost of Customization	LOW	MEDIUM	HIGH (error prone and time to work with someone else' code)
Extendible	EXCELLENT	EXCELLENT	GOOD

5 Acknowledgements

The authors would like to thank members of DeepTutor project (www.deeptutor.org). This research was supported by the Institute for Education Sciences (IES) under award R305A100875 by a grant from the Army Research Lab (ARL). Any opinions, findings and conclusions, or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the IES or ARL.

6 References

1. Blei, D.M., Ng, A.Y., & Jordan, M.I. 2003. Latent dirichlet allocation, The Journal of Machine Learning Research 3, 993-1022.
2. Corcoran, T., Mosher, F.A., & Rogat, A. (2009). Learning progressions in science: An evidencebased approach to reform. Consortium for Policy Research in Education Report #RR-63. Philadelphia, PA: Consortium for Policy Research in Education.
3. Duschl, R.A., Schweingruber, H.A., & Shouse, A. (Eds.). (2007). Taking science to school: Learning and teaching science in grades K-8. Washington, DC: National Academy Press.
4. Graesser, A. C.; Olney, A.; Haynes, B. C.; and Chipman, P. 2005. Autotutor: A cognitive system that simulates a tutor that facilitates learning through mixed-initiative dialogue. In Cognitive Systems: Human Cognitive Models in Systems Design. Mahwah: Erlbaum.
5. Landauer, T.; McNamara, D. S.; Dennis, S.; and Kintsch, W. (2007). Handbook of Latent Semantic Analysis. Mahwah, NJ: Erlbaum.
6. Freedman, Reva, Yujian Zhou, Jung Hee Kim, Michael Glass, and Martha W. Evens.

7. SGML-Based Markup as a Step toward Improving Knowledge Acquisition for Text Generation AAAI 1998 Spring Symposium: Applying Machine Learning to Discourse Processing
8. VanLehn, K. (2011). The Relative Effectiveness of Human Tutoring, Intelligent Tutoring Systems, and Other Tutoring Systems, Educational Psychologist, 46:4, 197-221.
9. Olney, A., D'Mello, A., Person, N., Cade, W., Hays, P., Williams, C., Lehman, B., & Graesser, A. (2012). Guru: A computer tutor that models expert human tutors. In S. Cerri, W. Clancey, G. Papadourakis & K. Panourgia (Eds.), Proceedings of the 11th International Conference on Intelligent Tutoring Systems (pp. 256-261). Springer-Verlag.
10. Patil, A. S., & Abraham, A. (2010). Intelligent and Interactive Web-Based Tutoring System in Engineering Education: Reviews, Perspectives and Development. In F. Xhafa, S. Caballe, A. Abraham, T. Daradoumis, & A. Juan Perez (Eds.), Computational Intelligence for Technology Enhanced Learning. Studies in Computational Intelligence (Vol 273, pp. 79-97). Berlin: Springer-Verlag.
11. Rus, V. & Lintean, M. (2012a). A Comparison of Greedy and Optimal Assessment of Natural Language Student Input Using Word-to-Word Similarity Metrics, Proceedings of the Seventh Workshop on Innovative Use of Natural Language Processing for Building Educational Applications, NAACL-HLT 2012, Montreal, Canada, June 7-8, 2012.
12. Rus, V., Lintean, M., Moldovan, C., Baggett, W., Niraula, N., Morgan, B. (2012b). The SIMILAR Corpus: A Resource to Foster the Qualitative Understanding of Semantic Similarity of Texts, In Semantic Relations II: Enhancing Resources and Applications, The 8th Language Resources and Evaluation Conference (LREC 2012), May 23-25, Instanbul, Turkey.
13. Rus, V.; Lintean, M.; Banjade, R.; Niraula, N.; Stefanescu, D. (2013). SEMILAR: The Semantic Similarity Toolkit, The 51st Annual Meeting of the Association for Computational Linguistics, System Demo Paper, August 4-9, 2013, Sofia, Bulgaria.
14. Rus, V., D'Mello, S., Hu, X., and Graesser, A.C. (to appear) .Recent Advances In Conversational Intelligent Tutoring Systems, AI Magazine.
15. VanLehn, K., Graesser, A. C., Jackson, G. T., Jordan, P., Olney, A., & Rose, C. P. (2007). When are tutorial dialogues more effective than reading? Cognitive Science, 31, 3-62.
16. VanLehn, K. (2011). The Relative Effectiveness of Human Tutoring, Intelligent Tutoring Systems, and Other Tutoring Systems, Educational Psychologist, 46:4, 197-221.

Authors

Vasile Rus: Dr. Vasile Rus is an Associate Professor of Computer Science with a joint appointment in the Institute for Intelligent Systems (IIS). Dr. Rus' areas of expertise are computational linguistics, artificial intelligence, software engineering, and computer science in general. His research areas of interest include question answering and asking, dialogue-based intelligent tutoring systems (ITSs), knowledge representation and reasoning, information retrieval, and machine learning. For the past 10 years, Dr. Rus has been heavily involved in various dialogue-based ITS projects including systems that tutor students on science topics (DeepTutor), reading strategies (iSTART), writing strategies (W-Pal), and metacognitive skills (MetaTutor). Currently, Dr. Rus leads the development of the first intelligent tutoring system based on learning progressions, DeepTutor (www.deeptutor.org). He has coedited three books, received several Best Paper Awards, and authored more than 90 publications in top, peer-reviewed international conferences and journals. He is currently Associate Editor of the International Journal on Artificial Intelligence Tools.

Nobal Niraula: Nobal B. Niraula received the B.E. in computer engineering from Pulchowk Campus, Tribhuvan University, Nepal, the M.E. in information and communication technology and the M.Sc. in communication networks and services from Asian Institute of Technology, Thailand and Telecom SudParis, France respectively. He was a research engineer at INRIA, Saclay, France where he worked in semantic web, database systems and P2P networks. Currently, he has been doing his PhD at The University of Memphis, USA. His research interests are primarily in Intelligent Tutoring Systems, Dialogue Systems, Information Extraction, Machine Learning, Data Mining, Semantic Web, P2P and Ad hoc networks. He has received the best paper and the best presentation awards. He also has intern experiences in leading research labs such as AT&T Labs Research.Mihai Lintean: Dr. Mihai Lintean is currently a research scientist at Carney Labs LLC and previous to that he was a Postdoctoral Research Fellow in the Computer Science Department at the University of Memphis, where he worked with Dr. Vasile Rus ondialogue based tutoring systems for teaching conceptual physics to high school students. Mihai's primary research interests are in Natural Language Processing (NLP), with focused applicability on educational technologies such as intelligent tutoring systems. Particularly he is interested in measuring semantic similarity between texts, representing knowledge through relational diagrams of concepts, automatic generation of questions, and using various machine learning techniques to solve other complex NLP problems. Mihai has published numerous papers and articles in reputable, peer-reviewed conferences and journals. He currently serves as co-chair of the Applied Natural Language Processing Special Track at the 25th International Conference of the Florida Artificial Intelligence Research Society (FLAIRS 2012).

Rajendra Banjade: Rajendra Banjade is a PhD student in Computer Science at The University of Memphis. He is a research assistant in the DeepTutor project (www.deeptutor.org) - a dialogue based tutoring system. Rajendra's research interests are in the area of Natural Language Processing, Information Retrieval, and Data Mining. Currently, he is focusing on measuring semantic similarity of short texts (word and sentence level) using knowledge based and corpus based methods and heading towards more human like inferencing techniques. His current research focus is on robust methods to evaluate student answers in conversational intelligent tutoring systems. He is keenly dedicated to enhancing the SEMILAR toolkit (www.semanticsimilarity.org) which is an off-the-shelf semantic similarity toolkit. Before joining The University of Memphis, he worked for five years as a Software Engineer (R&D) at Verisk Information Technologies CMMI III, Kathmandu (a subsidiary of Verisk Analytics inc.) where he got opportunities working on various healthcare data mining projects including DxCG Risk Solutions engine. He received an outstanding employee award at Verisk. Rajendra is a certified Scrum Master and Software Developer, and Certified HIPAA professional. He holds bachelor's degree in Computer Engineering.

William B. Baggett: William B. Baggett earned a PhD in Cognitive Psychology from The University of Memphis in 1998. He also holds an MS in Computer Science and an MBA in Management Information Systems. William is currently a Project Coordinator in the Computer Science Department at The University of Memphis, where he works on DeepTutor. DeepTutor is an intelligent tutoring system, implemented as a web application, which uses natural language dialogue to teach conceptual physics to high school and college students. Previously, William was a Professor and part-time Department Chair of Computer Information Systems at Strayer University and an adjunct Professor of Computer Science at The University of Memphis. In both positions, William taught graduate and undergraduate Computer Science courses, mentored, tutored, and advised students, and developed new curricula. He was also a Business Analyst at FedEx Express where he wrote software specifications for PowerPad, a mission-critical handheld computer carried by FedEx Express couriers. PowerPad software is designed to promote optimal courier behavior including the efficient pickup and delivery of FedEx shipments, package tracking, and conformance to policies and procedures for a wide variety of domestic and international services.

Dan Ștefănescu: Dr. Dan Ștefănescu is a Postdoctoral Research Fellow in the Department of Computer Science of the University of Memphis and the Institute for Intelligent Systems (IIS). As a member of DeepTutor team, his main research activity is dialogue-based Intelligent Tutoring Systems. Previously, Dr. Ștefănescu was a Senior Researcher at the Research Institute for Artificial Intelligence (RACAI) in Bucharest, Romania. He graduated from the Computer Science Faculty of ”A.I. Cuza” University of Iași in 2002 and obtained his MSc in Computational Linguistics from the same university in 2004. In 2010 he was awarded the PhD title (Magna Cum Laude) at the Romanian Academy for a thesis on Knowledge Extraction from Multilingual Corpora. He authored more than 50 papers in peer-reviewed journals and conference proceedings and successfully participated in various software competitions like the Question-Answering competitions organized by Conference and Labs of the Evaluation Forum (CLEF), Microsoft Imagine Cup and Microsoft Speller Challenge. His research work covers various Natural Language Processing topics like: Question Answering, Information Extraction, Word Sense Disambiguation, Connotation/Sentiment Analysis, Collocations/Terminology Identification, Machine Translation, or Query Alteration for Search Engines.

The SCHOLAR Legacy: A New Look at the Affordances of Semantic Networks for Conversational Agents in Intelligent Tutoring Systems

Donald M. Morrison and Vasile Rus

Institute for Intelligent Systems, The University of Memphis, Memphis, Tennessee
{dmmrrson and vrus}@memphis.edu

Abstract. The time is ripe for a new look at the affordances of semantic networks as backbone structures for knowledge representation in intelligent tutoring systems (ITSs). While the semantic space approach has undeniable value, and will likely continue to be an essential part of solutions to the problem of computer-based dialogue with humans, technical advances such the automatic extraction of ontologies from text corpora, now encourage a vision in which intelligent tutoring agents have access to forms of knowledge representation that allow them to more fully "understand" something of what they are talking about with learners. These developments have important implications for key ITS components including the structure of expert domain models, learner models, instructional modules, and dialogue strategies, particularly in respect to issues of transportability across systems. As such, they in turn have important implications for the design of a general-purpose framework such as the U.S. Army's Generalized Intelligent Framework for Tutoring (GIFT).

Keywords: Intelligent tutoring, semantic networks, semantic spaces, ontology extraction.

1 Introduction

The idea that a computer might be programmed to carry on an intelligent conversation with a human emerged in the early days of artificial intelligence, possibly as early as the 1940s, but was articulated most fully in computer pioneer Alan Turing's famous "Turing test" [40] in which a human is invited to carry on a typed conversation with both a hidden human and a machine, and has to decide which is which. A computer program that passes the Turing test is considered to be intelligent. Early programs that were claimed to have passed the test included ELIZA [43], which employed the ping-pong conversational strategies of a Rogerian psychotherapist, thus, allowing ELIZA to be "free to assume the pose of knowing almost nothing of the real world" [p. 42], and PARRY, which was designed to mimic the behavior of a paranoid schizophrenic, and reportedly fooled about half the psychologists who interacted with it [11].

SHRDLU [44] was similar to ELIZA and PARRY in that it was capable of engaging humans in something like natural language dialog, but distinctly different in one important respect; it actually "understood" something about what it was talking about, albeit within a highly restricted domain, a virtual world consisting of a set of wooden blocks of different shapes and colors. SHRDLU could be instructed to pick up a block, put it on top of another block, and would remember which block was now on top.

SCHOLAR [9, 10, 17] is thought to have been the first program to demonstrate the use of mixed-initiative, natural-language conversation for the specific purpose of instruction, and as such is widely considered the first intelligent tutoring system [e.g., 29,14]. The program was intended to demonstrate what Carbonell, a computer scientist at Bolt, Beranek and Newman (BBN), termed an "information-structure-oriented" (ISO) tutorial system. In contrast to "ad hoc-frame oriented" (AFO) systems of the time, which were organized around "frames" of specific text strings, questions, and answers written by a human author, SCHOLAR drew its questions and answers *directly from its own knowledge base*, which, in the examples referenced here, consisted of a set of simple facts about the geography of South America.[8] The following transcript, reported by Freedman [17], gives some sense of how these interactions went:

SCHOLAR: The capital of Chile is Santiago. Correct or incorrect?
STUDENT: Correct.
SCHOLAR: That's right. Approximately what is the area of Brazil?
STUDENT: 2,500,000 square miles.
SCHOLAR: Wrong. Please indicate if the following statement is correct or incorrect:
The area of Paraguay is approx. 47432 square miles.

While it seems unlikely that SCHOLAR fooled anyone, the program was arguably ahead of its time in one important respect. SCHOLAR's knowledge of the world was stored in a *semantic network*, a data structure that featured in the work of Carbonell's colleagues at BBN, Ross Quillian and Allan Collins [32, 12, 13]. Semantic networks do not, in themselves, provide easy solutions to the problem of machine understanding of human language; however, for reasons explained below, there is good reason to take a second look at the various affordances they may offer to designers of general-purpose intelligent tutoring systems (ITSs), including general-purpose frameworks such as GIFT.

2 Affordances of Semantic Networks for Intelligent Tutoring Systems

Researchers in artificial intelligence have explored a range of solutions to the problem of representation of conceptual knowledge, from symbolic representations to purely statistical ones [25,19]. Semantic networks of the type employed by SCHOLAR, where concepts and their relationships are represented as nodes and edg-

[8] Carbonell was born in Uruguay. A second database was developed to provide tutoring for an online text editing system.

es, are arguably closest to symbolic natural language in that noun-predicate-object clusters (semantic triples) are incorporated and preserved. In “semantic space” models, on the other hand, relationships among concepts are represented mathematically. Methods include Latent Semantic Analysis (LSA) [24], Hyperspace Analogue to Language (HAL) [26], Latent Dirichlet Allocation (LDA) [5], Non-Latent Similarity (NLS) [8]; Word Association Space (WAS) [39], and Pointwise Mutual Information (PMI) [33].

In general terms, these semantic space models identify the meaning of a word through “the company it keeps” [15:11], that is, by examining the co-occurrence of words across large numbers of documents and using this data to calculate statistical measures of semantic similarity. This approach has been used successfully in a variety of applications where measures of document similarity are useful, such as in text retrieval and automatic scoring of student essays [25]. In intelligent tutoring applications, probabilistic semantic space engines allow for the automatic creation of domain models as “bags of words” [20]. For example, AutoTutor employs LSA measures of text similarity to evaluate the extent to which a learner’s answers to its questions correspond to scripted correct answers consisting of unordered sets of expected words and phrases [42].

When applied to the problem of knowledge representation in intelligent learning systems, the selection of one approach over another results in important trade-offs. Although the choice of probabilistic semantic models in intelligent tutoring systems avoids the time-consuming tasks involved in creating more granular, linguistically encoded models of domain knowledge, it also imposes significant constraints on the functionality of the system, including limits on its ability to engage in true dialog with a human learner, which in turn constrains both its ability to represent what is in the learner’s head *and* the nature and quality of the apparent (virtual) social relationship between the agent and the learner.

Most importantly, an agent that relies exclusively on a probabilistic semantic model cannot generate substantive questions of its own, nor can it respond to a learner’s questions. Rather, because its knowledge is enclosed in a “black box” [1] it is limited to asking scripted questions with scripted answers, then evaluating the extent to which the learner’s answers conform. As a result, it naturally assumes the role of a traditional pedagogue, a teacher who looks only for correct answers to questions.

2.1 Some Recent Developments

In spite of these limitations, in recent years the use of probabilistic, black box semantic models has been favored over semantic network representations, owing, as noted above, largely to the difficulties inherent in laborious manual authoring of useful domain models based on semantic networks [35]. However, over the past decade or so this situation has begun to change in important ways. While the extraction of propositions (semantic triples) from connected text—the building blocks of semantic network solutions—remains as one of the hardest problems in artificial intelligence and machine learning [35,19], considerable progress has been made [e.g., 2, 31, 30, 6, 4].

For example, Berland & Charniak [2] developed an algorithm which, given a seed word such as *car*, and a large corpus of text to mine, identified the following as possible fillers for the slot ___ *is-part-of* ____[*car*]: *headlight*, *windshield*, *ignition*, *shifter*, *dashboard*, *radiator*, *brake*, *tailpipe*, etc. Similarly, Pantel & Ravichandran [31] describe an algorithm for automatically discovering semantic classes in large databases, labeling them, then relating instances to classes in the form X *is-a* Y. For example, for the instances *Olympia Snowe*, *Susan Collins*, and *James Jeffords*, the algorithm settled on *republican*, *senator*, *chairman*, *supporter*, and *conservative* as possible labels, meaning that it could form the basis for assertions such "*Olympia Snowe is a republican*."

Other relevant work includes the corpus of annotated propositional representations in PropBank [30], and AutoProp [6] a tool that has been designed to "propositionalize" texts that have already been reduced to clauses. More recently, members of the DBpedia project [4] have been working to extract semantic triples from Wikipedia itself. As of September 2011, the DBpedia dataset described more than 3.64 million "things," with consistent ontologies for some 416,000 persons, 526,000 places, 106,000 music albums, 60,000 films, 17,500 video games, 169,000 organizations, 183,000 species and 5,400 diseases. A similar project, Freebase, allows users to edit ontologies extracted from Wikipedia [27], while YAGO2 [21] is a knowledge base of similar size (nearly 10 million entities and events, as well as 80 million facts representing general world knowledge) that includes the dimensions of space and time in its ontologies. All of these projects employ a form of semantic network to represent conceptual knowledge.

Given the labor required in building formal representations of procedural knowledge by hand, it is natural to consider the possibility of automatic extraction of production rules from text corpora, using machine learning (data mining) methods similar to those for extracting declarative knowledge. As it turns out, work on this problem is already producing promising results. For example, Schumacher, Minor, Walter, & Bergmann [36] have compared two methods of extracting formal "workflow representations" of cooking recipes from the Web, finding that the frame-based SUNDANCE system [34] gives superior results, as rated by human experts. Song et al. [37] have tested a method for extracting procedural knowledge from PubMed abstracts. Jung, Ryu, Kim, & Myaeng [23] describe an approach to automatically constructing what they call "situation ontologies" by mining sets of how-to instructions from the large-scale web resources eHow (www.eHow.com) and wikiHow (www.wikihow.com).

While the implications of this work for the development of intelligent learning systems remain unclear, the possibilities inherent in semantic data mining of both declarative and procedural knowledge clearly deserve attention. It seems the most likely scenario is that future systems will employ different knowledge representations for different purposes. For example, Rus [35] describes the use of a hybrid solution, Latent Semantic Logic Form (LS-LF), for use in the extraction of expert knowledge bases from corpora such as textbooks. Also, while the use of semantic networks in particular domains may allow an agent to engage in something approaching intelligent conversation regarding these domains, the agent may still need a way of coping with user utterances that it cannot handle in any other way, much as humans make educated, intuitive guesses about the meaning of ambiguous or confusing utterances. For example, Hu & Martindale [22] discuss the use of a semantic vector model as a means of evaluating the relevance and novelty of a given utterance in a series of discourse moves, which is clearly useful in the event that an agent has no other way of evaluating a user's utterance.

2.2 Implications for General-purpose Tutoring Systems

The field of intelligent tutoring has come a long way in the four decades that separate us from the time of SCHOLAR. A recent estimate [28], identified some 370 ITS "architecture families," or which 12 were considered "major architectures," defined as those with at least ten scholarly papers published between the years 2009-2012. However, in spite of these efforts (representing investments of untold millions of taxpayer dollars), the field has not yet had much of an impact on educational practice. The study cited above, for example, estimated less than 1 million users worldwide. To put this in perspective, a recent estimate puts the number of school-age children in the U.S. at 70 million, and in the world at over 1 billion [7].

Important barriers to more widespread adoption and impact of ITSs include two important and related problems. One is the high cost of authoring domain-specific systems, recently estimated to require between 24 and 220 hours of development time for one hour of instruction, with a mean of around 100 hours [16]. A second problem is that ITSs tend to be constructed as "unique, one-of-a-kind, largely domain-dependent solutions focused on a single pedagogical strategy" [38]. Among other things, because components are not shareable, this means that returns on investment in particular systems is limited to whatever impact those particular systems might on their own, like stones tossed into a pond that make no ripples.

The use of semantic networks to represent expert domain knowledge might go far to reduce authoring costs and could also lead to portable expert models, and, by extension, learner models. As we have seen, a considerable amount of work is already going on in the semi-automatic (i.e., supervised) extraction of domain ontologies from text corpora. What this means, conceptually, is that the ontology of a particular domain becomes not just a single person (or team's) unique description of the domain of interest, but a structure that emerges from the way the domain is represented linguistically in some very large number of texts, written by different authors. While it is true that supervised extraction introduces and reflected the biases of the human supervisors, ontologies constructed in this way arguably have much more in common than those constructed entirely from scratch for specific purposes. The ability to extract domain models directly from text corpora also, of course, speeds the development process, and, to the extent that expert models constructed in this way are architecture-independent, they are more likely to acquire general currency than dedicated models developed for the particular purposes of specific systems. Finally, to the extent that learner models, or at least some portion of them, are seen as overlays of expert models (i.e., flawed or incomplete versions of expert maps), these may also become transportable across systems, and because these models can be expressed mathematically, as graphs, it becomes possible to estimate differences between learner models and expert models computationally.

3 Conclusion

While the specific affordances of semantic networks in respect to problems of knowledge representation, learner modeling, and conversational fluency of intelligent agents have yet to be fully explored, and while such structures do not by any means solve fundamental problems, the future is indeed promising. As argued here, the movement to structure the vast store of human knowledge on the Web in the form of explicit ontologies, as evidenced in the Semantic Web project and its many associated technologies, is well underway, and has undeniable momentum. The future of human knowledge representation almost certainly lies in this direction, with some obvious potential benefits to ITS developers. For example, to the extent that expert domain models are conceived as populated ontologies, then it becomes easier to conceive of portable domain models, and, to the extent that a learner models are *also* conceived of as populated ontologies, then learner models can also be portable across systems.

Interestingly, the underpinnings of the Semantic Web originated in the work of Ross Quillian, the same work that SCHOLAR, the ancestor of modern ITSs, was based on. Now that the technology is beginning to catch up with that initial vision, the time has arguably come to take another look at the affordances of semantic networks. In particular, the designers of systems such as GIFT, which seek to provide a general-purpose framework for development of ITS systems of the future, are advised to look carefully at the specific implications of the reemergence and increasing importance of semantic networks as general-purpose structures for representing the knowledge of both experts and learners, and as the basis for bringing these structures into alignment through natural processes of teaching and learning.

References

1. Anderson, J.R. The expert model. In Polson, M. C., & Richardson, J. J. (eds.) Foundations of intelligent tutoring systems. Lawrence Erlbaum. 21-53 (1988)
2. Berland, M., Charniak, E. Finding parts in very large corpora. In Annual Meeting-Association For Computational Linguistics 37, 57-64 (1999, June)
3. Bickmore, T., Schulman, D., Yin, L. Engagement vs. deceit: Virtual humans with human autobiographies. In Intelligent Virtual Agents, pp. 6-19. Springer Berlin/Heidelberg (2009)
4. Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., Hellmann, S. DBpedia-A crystallization point for the Web of Data. Web Semantics: Science, Services and Agents on the World Wide Web, 7(3), 154-165 (2009)
5. Blei, D. M., Ng, A. Y., Jordan, M. I. Latent dirichlet allocation. The Journal of Machine Learning Research, 3, 993-1022 (2003)
6. Briner, S.W., McCarthy, P.M., McNamara, D.S. Automating text propositionalization: An assessment of AutoProp. In R. Sun & N. Miyake (eds.), Proceedings of the 28th Annual Conference of the Cognitive Science Society, pp. 2449. Austin, TX: Cognitive Science Society (2006)
7. Bruneforth, M. & Wallet, P. Out-of-school adolescents. UNESCO Institute for Statistics. (2010).

8. Cai, Z., McNamara, D. S., Louwerse, M., Hu, X., Rowe, M., Graesser, A. C. NLS: A non-latent similarity algorithm. In Proc. 26th Ann. Meeting of the Cognitive Science Soc., CogSci'04, pp. 180-185 (2004)
9. Carbonell, J. R. AI in CAI: Artificial intelligence approach to computer assisted instruction. IEEE Transactions on Man-Machine Systems 11(4): 190-202 (1970)
10. Carbonell, J. R., Collins, A. M. Natural semantics in artificial intelligence. In Proceedings of the Third International Joint Conference on Artificial Intelligence. IJCAI 73, 344 -351 (1973, August)
11. Colby, K. M., Hilf, F. D., Weber, S., Kraemer, H. C. Turing-like indistinguishability tests for the validation of a computer simulation of paranoid processes. Artificial Intelligence, 3, 199-221 (1972)
12. Collins, A. M., Loftus, E. F. A spreading-activation theory of semantic processing. Psychological review, 82(6), 407 (1975)
13. Collins, A. M., Quillian, M. R. Retrieval time from semantic memory. Journal of verbal learning and verbal behavior, 8(2), 240-247 (1969)
14. Corbett, A. T., Koedinger, K. R., Anderson, J. R. Intelligent tutoring systems. Handbook of human-computer interaction, 849-874 (1997)
15. Firth, John Rupert. A synopsis of linguistic theory, 1930-1955. (1957).
16. Folsom-Kovarik, J. T., S. Schatz, and D. Nicholson. Return on investment: A practical review of ITS student modeling techniques. M&S Journal, Winter Edition (2011): 22-37.
17. Freedman, R. Degrees of mixed-initiative interaction in an intelligent tutoring system. In Proceedings of the AAAI Spring Symposium on Computational Models for Mixed Initiative Interaction, Palo Alto, CA. Menlo Park, CA: AAAI Press (1997)
18. Graesser, A. C., Lu, S., Jackson, G. T., Mitchell, H. H., Ventura, M., Olney, A., Louwerse, M. M. AutoTutor: A tutor with dialogue in natural language. Behavior Research Methods, Instruments, & Computers, 36(2), 180-192 (2004)
19. Graesser, A. C., McNamara, D. S., Louwerse, M. M. Two methods of automated text analysis. Handbook of Reading Research, 34 (2009)
20. Harris, Z. Distributional structure. Word 10 (2/3): 146–62 (1954)
21. Hoffart, J., Suchanek, F. M., Berberich, K., Lewis-Kelham, E., De Melo, G., Weikum, G. Yago2: exploring and querying world knowledge in time, space, context, and many languages. In Proceedings of the 20th international conference companion on World Wide Web, pp. 229-232. ACM (2011, March)
22. Hu, X., Martindale, T. Enhancing learning with ITS-style interactions between learner and content. Interservice/Industry Training, Simulation & Education 2008, 8218, 1-11 (2008)
23. Jung, Y., Ryu, J., Kim, K. M., Myaeng, S. H. Automatic construction of a large-scale situation ontology by mining how-to instructions from the web. Web Semantics: Science, Services and Agents on the World Wide Web, 8(2), 110-124 (2010)
24. Landauer, T. K., Dumais, S. T. A solution to Plato's problem: The Latent Semantic Analysis theory of the acquisition, induction, and representation of knowledge. Psychological Review , 104 , 211-140 (1997)
25. Landauer, T. K., Laham, D., Foltz, P. W. Automated scoring and annotation of essays with the Intelligent Essay Assessor. Automated essay scoring: A cross-disciplinary perspective, 87-112 (2003)
26. Lund, K., & Burgess, C. Producing high-dimensional semantic spaces from lexical co-occurrence. Behavior Research Methods, Instruments, & Computers, 28(2), 203-208 (1996)
27. Markoff, J. Start-up aims for database to automate web searching. The New York Times (2007-03-09) Retrieved 4/21/2013

28. Nye, B.. Two sigma or two percent: A mapping study on barriers to ITS adoption. (in preparation)
29. Nwana, H. S. Intelligent tutoring systems: an overview. Artificial Intelligence Review, 4(4), 251-277 (1990)
30. Palmer, M., Kingsbury, P., Gildea, D. The Proposition Bank: An annotated corpus of semantic roles. Computational Linguistics, 31, 71-106 (2005)
31. Pantel, P., & Ravichandran, D. Automatically labeling semantic classes. In Proceedings of HLT/NAACL (4), 321-328 (2004, May)
32. Quillian, M.R. Semantic memory. In M. Minsky, (E.), Semantic Information Processing. MIT Press, Cambridge, MA (1968).
33. Recchia, G., Jones, M. N. More data trumps smarter algorithms: Comparing pointwise mutual information with latent semantic analysis. Behavior Research Methods, 41(3), 647-656 (2009)
34. Riloff, E., Phillips, W. An introduction to the sundance and autoslog systems. Technical Report UUCS-04-015, School of Computing, University of Utah (2004)
35. Rus, V. What next in knowledge representation? Proceedings of the International Conference on Knowledge Engineering, Principles and Techniques. Cluj-Napoca (Romania), July 2-4, pp. 1-9 (2009)
36. Schumacher, P., Minor, M., Walter, K., Bergmann, R. Extraction of procedural knowledge from the web: A comparison of two workflow extraction approaches. In Proceedings of the 21st international conference companion on World Wide Web, pp. 739-747. ACM (2012, April)
37. Song, S. K., Choi, Y. S., Oh, H. S., Myaeng, S. H., Choi, S. P., Chun, H. W., and Sung, W. K. Feasibility study for procedural knowledge extraction in biomedical documents. Information Retrieval Technology, 519-528 (2011)
38. Sottilare, R. A., Goldberg, B. S., Brawner, K. W., & Holden, H. K. 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. (2012, January).
39. Steyvers, M., Griffiths, T. L., Dennis, S. Probabilistic inference in human semantic memory. Trends in Cognitive Sciences, 10(7), 327-334 (2006)
40. Turing, A. M. Computing machinery and intelligence. Mind, 59(236), 433-460 (1950)
41. VanLehn, K. The behavior of tutoring systems. International journal of artificial intelligence in education, 16(3), 227-265 (2006)
42. Wiemer-Hastings, Peter, Arthur C. Graesser, and Derek Harter. The foundations and architecture of AutoTutor. In Intelligent Tutoring Systems, pp. 334-343. Springer Berlin Heidelberg, 1998.
43. Weizenbaum, J. ELIZA—A computer program for the study of natural language communication between man and machine. Communications of the ACM, 9(1), 36-45 (1966)
44. Winograd, T. Procedures as a representation for data in a computer program for understanding natural language. MIT AI Technical Report 235 (February 1971)

Authors

Dr. Chip Morrison is a Faculty Affiliate at IIS. A graduate of Dartmouth, Dr. Morrison holds an M.A. from the University of Hong Kong and an Ed.D. from Harvard. His current research interests include models of human cognition and learning, and the application of these models to conversation-based intelligent learning systems.

Vasile Rus: Dr. Vasile Rus is an Associate Professor of Computer Science with a joint appointment in the Institute for Intelligent Systems (IIS). Dr. Rus' areas of expertise are computational linguistics, artificial intelligence, software engineering, and computer science in general. His research areas of interest include question answering and asking, dialogue-based intelligent tutoring systems (ITSs), knowledge representation and reasoning, information retrieval, and machine learning. For the past 10 years, Dr. Rus has been heavily involved in various dialogue-based ITS projects including systems that tutor students on science topics (DeepTutor), reading strategies (iSTART), writing strategies (W-Pal), and metacognitive skills (MetaTutor). Currently, Dr. Rus leads the development of the first intelligent tutoring system based on learning progressions, DeepTutor (www.deeptutor.org). He has coedited three books, received several Best Paper Awards, and authored more than 90 publications in top, peer-reviewed international conferences and journals. He is currently Associate Editor of the International Journal on Artificial Intelligence Tools.

XNAgent: Authoring Embodied Conversational Agents for Tutor-User Interfaces

Andrew M. Olney, Patrick Hays, & Whitney L. Cade

Institute for Intelligent Systems & Department of Psychology
365 Innovation Drive
Memphis, Tennessee 38152
{aolney,dphays,wlcade}@memphis.edu
http://iis.memphis.edu

Abstract. Embodied conversational agents are virtual characters that engage users in conversation with appropriate speech, gesture, and facial expression. The high cost of developing embodied conversational agents has led to a recent increase in open source agent platforms. In this paper, we present XNAgent, an open source platform for embodied conversational agents based on the XNA Framework. By leveraging the high-level class structure of the XNA Framework, XNAgent provides a compact implementation that is suitable both as a starting point for the development of a more advanced system and as a teaching tool for AI curricula. In this paper we describe how we created an embodied conversational agent in XNA using skeletal and morph animation, motion capture, and event-driven animation and how this process can facilitate the use of embodied conversational agents in the Generalized Intelligent Framework for Tutoring.

Keywords: XNA, ECA, GIFT, agent, HCI, conversation, interface, tutoring

1 Introduction

It is well known that we unconsciously and automatically interact with computers using social norms [1]. Embodied conversational agents (ECAs) capitalize on this phenomena as characters with human-like communicative capabilities. By doing so, ECAs leverage pointing, gestures, facial expressions, and voice to create a richer human-computer interface. As a result ECAs have been used in diverse AI applications, including education [2], where they form an important part of the tutor-user interface.

ECAs combine research in discourse, computer animation, speech synthesis, and emotion. Consequently ECA systems tend to be costly to build [3] As a result, in the past decade, a great deal of tutoring research has used closed-source platforms such as Microsoft Agent [4], adapted commercial/open source game engines [5], or low-level libraries like OpenGL [6]. These approaches present different types of challenges. Game engines usually have support for basic character animation but lack native lip-sync and fine animation control, and game engines come with a complex API with many features that may not be relevant for education research, e.g. bullet/explosion physics or first-person shooter perspective. Conversely low-level libraries have no similar irrelevant complexity but require designing the AI from the ground up. Given the challenges of both the game-engine and low-level routes, recent researchers have released open source platforms for ECA development [7, 8, 9, 10] based on either game engines or low-level libraries.

The design and development challenges described above for ECAs are manifest in the development of computer-based training environments and have recently been addressed by the Generalized Intelligent Framework for Tutoring Framework [11]. One of the design goals of the Generalized Intelligent Framework for Tutoring (GIFT) is to provide authoring capability for the creation of computer-based training components. One such component is the tutor-user interface, which in modern intelligent tutoring systems often uses an ECA. Accordingly, in this paper we present an open source solution to ECA development that meets the design goals of the GIFT Framework. Rather than use a game engine with its inherent complexities or a low-level library that requires a large investment of initial development, we present an ECA platform that combines the best of these using Microsoft's XNA framework [12]. By providing high-level libraries, a runtime environment for managed code (C#), free development tools, and extensive support in the form of code samples, official forums, and commercially available books at all levels, the XNA framework provides a solid foundation for ECA development. In this the following sections we describe how we implement the face and body of XNAgent using skeletal and morph animation via vertex shaders, motion capture, and event-driven animation. At each step the content creation pipeline is outlined to illustrate how XNAgent may be adapted to new AI contexts. We conclude by considering the design goals of the GIFT Framework and how they are addressed by XNAgent.

2 Face

The face of an ECA can be considered independently of the body in terms of speech, emotions, and facial expressions. The classic reference for facial expression is the Facial Action Coding System, which uses the anatomy of the face, primarily in terms of muscle groups, to define facial action units [13]. While it is certainly possible to create "virtual muscles" and animate with them, a number of other real-time approaches exist which give satisfactory results [14]. Perhaps the most well-known and widely used facial animation approach is morph target animation.

In morph target animation, a version of the head is created for each desired expression. For example, one version for smiling, frowning, or a "w" lip shape. Each of these shapes becomes a target for interpolation, a morph target. If two morph channels exist, e.g. a neutral face and a smiling face, the interpolation between them can be described by the distance between matching vertices across the two faces. In practice, this distance is often normalized as a weight such that a weight of 1 would push the neutral face all the way to happy. The advantage to using morph target animations is that each morph target can be carefully crafted to the correct expression, and then mixtures of morph targets can be used to create huge number of intermediate expressions, e.g. smiling while talking and blinking.

FaceGen Modeler, by Singular Inversions, is a popular software package for creating 3D faces that has been used in psychological research on gaze, facial expression, and attractiveness [15]. FaceGen Modeler contains a statistical model of the human face with approximately one hundred and fifty parameters to vary face shape and texture. Using FaceGen Modeler, a virtual infinite variety of human faces can be created by manipulating these parameters, and for a given custom face FaceGen Modeler can output thirty-nine morph targets including seven emotions and sixteen visemes (the visual correlates of phonemes used for lip-sync). XNAgent uses FaceGen Modeler output, so a correspondingly large variety of faces can be implemented in XNAgent.

Since XNA does not provide native support for morph targets, we have implemented them using vertex shaders. A shader is a program that runs directly on the graphics card. In XNA, shaders are written in High Level Shader Language that resembles the C programming language, and the shaders compile side by side with C#. To implement morph target animation, XNAgent's vertex shaders operate on each vertex on face and perform bilinear interpolation (interpolation on two axes). Thus there are three versions of the XNAgent head loaded at any particular time: a neutral head that was skinned with the body (see Section 3), a viseme head for the current viseme, and an emotion/expression head for the current emotion. It is possible to have more channels for additional morphing, and these are easily added if necessary.

XNAgent utilizes a dynamic, event-driven animation system for facial expressions. Three categories of facial animation are currently supported, including blinking, lip-sync via visemes, and facial expressions. Blinking is implemented using a model of blinking behavior in humans [16] in its own thread. Because the most salient feature of blinking is perhaps that the eyelids cover the eyes, XNAgent imitates blinking through texture animation rather than morph target animation. In texture animation the texture of the face is switched quickly with another version of the face. In the case of blinking the two textures are nearly identical except the blink texture's eyes are colored to match the surrounding skin, thus simulating closed eyes.

Lip-syncing through morph target animation is controlled by the agent's voice, i.e. a text-to-speech synthesizer. Some speech synthesizers generate lip-sync information during synthesis by producing visemes, the visual correlates of phonemes. Each viseme unit typically includes the current viseme and the viseme's duration. In a viseme event handler, XNAgent sets the current viseme morph target and its duration using these values. In the Update() loop, the viseme's time left is decremented by the elapsed time. In the Draw() loop, the viseme morph is expressed with a weight based on the remaining time left. Thus the lip sync remains true independently of the framerate speed of the computer running XNAgent and linearly interpolates between visemes.

Morphing expressions like emotions require a more flexible approach than viseme animations. For example, a smile can be a slow smile that peaks at a medium value, or a rapid smile that peaks at an extreme value. To capture these intuitions, our expression morph animation has parameters for rise, sustain, and decay times, with a maximum weight parameters that specifies what the maximal morph will be during the sustain phase. Currently these three phases are interpolated linearly.

3 Body

Non-facial movements, or gestures, appear to greatly differ from the face greatly in terms of communicative complexity, forming sign language in the extreme case. Our approach is therefore to model the entire body as a collection of joints, such that manipulating the values of these joints will cause the body to move. This common approach to animation is often called skeletal, or skinned animation [17].

In skinned animation a character “shell” is first created that represents a static character. An underlying skeletal structure is created for the shell with appropriate placement of joints and placed inside the shell. The shell and skeleton are then bound together such that a transformation on the underlying skeleton is mirrored in the shell; this result is known as a rigged model. Once a model is rigged, it may be animated by manipulating the skeleton and saving the resulting joint position data. Every saved movement creates a keyframe, and when these keyframes are played back at a rapid rate (e.g. 30 fps) the rigged model will carry out the animated action. Alternatively motion capture technologies can extrapolate joint position data from naturalistic human movement. In this case the resulting animation is still a keyframe animation.

In order to create a body for XNAgent, we used several software packages to form what is commonly known as a 3D authoring pipeline. At each stage of the pipeline there are multiple available techniques and software packages, making navigating this space a complex process. In brief, there are three major phases to creating a body with gestures, namely model creation, rigging, and animation. Model creation can be extremely difficult for non-artists without initial materials to work from. To facilitate the process of body creation, we used the face model generated by FaceGen Modeler together with the FaceGen Exporter to export the face model to the Daz Studio software package. This process seamlessly combines the face and body models and auto-rigs the body with a skeleton. Daz Studio allows for comparable customizations of the body (gender, size, shape) as FaceGen does for the face. In addition, Daz Studio comes with a variety clothing and accessory packs that can be applied to the body in a drag and drop manner. In effect, several hundred hours of 3D authoring can be accomplished by a novice in less than an hour.

In order to create realistic animations, we primarily used the low-cost iPi Desktop Motion Capture system from iPi Soft. The simplest camera configuration for this system uses the Microsoft Kinect camera. Once the motion capture has been recorded by iPi, it can be merged and edited using AutoDesk 3DS Max, where ultimately it is exported for XNA using the kw X-port plugin. A complete description of this process is beyond the space limitations of the current discussion, but a full tutorial, including software installer and step by step slides, is available from the corresponding author’s website[9].

In order to achieve similar functionality to interpolating visemes, skinned animation clips require mechanisms for blending and mixing. Simply put, blending is end to end interpolation, like a DJ fading from one song to the next. Mixing breaks the animation into components and plays them simultaneously, like a DJ taking the beat from one song, vocals from another, and playing them together. Blending and mixing can be done simultaneously if clips are playing in different regions of the skeleton

[9] http://andrewmolney.name

while being blended with other clips in those same regions. XNAgent uses the Communist Animation Library [18] to perform blending and mixing. Currently in XNAgent the skeleton is divided into center, left side, right side, and head regions. These regions are used to represent the following tracks: idle, both arms, left arm, right arm, and head. Animations are assigned to tracks at design time and then played with weights according to what other animations are currently playing in their track. For example, the idle animation consists of motion capture of a person standing and slightly swaying. Some degree of the idle animation is always playing in all the tracks, but when other animations are played in those tracks they are played with a higher weight. Thus lower priority animations like idle will be superseded by higher priority animations in a relatively simple manner.

Animations are triggered in XNAgent by inserting animation tags into the text to speak, either dynamically or manually. When the TTS encounters the tag, it schedules the animation immediately. The mixing properties of the animation are specified in the tag to create new versions of animations, similar to morphing. For example, since the idle animation is always playing, it can be given more weight relative to an arm gesture to create a "beat" gesture [19]. Thus a normal full arm extension animation can be dampened arbitrarily using weighting, bringing the arm closer to the body with increasing weight. In addition, the speed of the animation clip can be modulated to control for the appropriate speed of the beat gesture, since beat gestures are often quick and fluid.

Although XNA has some level of built in support for skinned animations, combining skinned animations with morph target animations requires a custom vertex shader. In XNAgent there are two vertex shaders that operate separately on the head and body of the agent. The head shader applies morphing to calculate a new vertex position and then applies the transformation defined by skinning. This allows the head to be applying morph targets (e.g. speaking) while also nodding or shaking. The second vertex shader focuses strictly on the body and so does not require morphing.

4 Working with XNAgent

One of the most important aspects of any ECA is its ability to integrate into an AI application. Game engines typically don't support integration well and rather present a fullscreen interface for the game, as does XNA. Although text input and other user interface functions can be carried out inside XNA, they are difficult because XNA doesn't provide the native support commonly expected by GUI designers. For example, key presses in XNA are interpreted based on the framerate of the game, meaning that a normal keystroke will produce a double or triple production of letters or numbers. To address the integration issue, XNAgent provides an XNA environment inside a Windows form control. That means that adding XNAgent to an interface is as simple as selecting the XNAgent control from the Visual Studio toolbox and dropping it on a form. The primary method to call on the control is Speak(), which processes both text to speech and animation tags as described in previous sections. In summary, the process for using XNAgent is (1) create a 3D model using the authoring pipeline described above (2) import the model to XNAgent (3) call XNAgent from your application using the Speak() method. We have previously integrated XNAgent into the Guru intelligent tutoring system shown in Figure 1 and conducted a number of experiments [20].

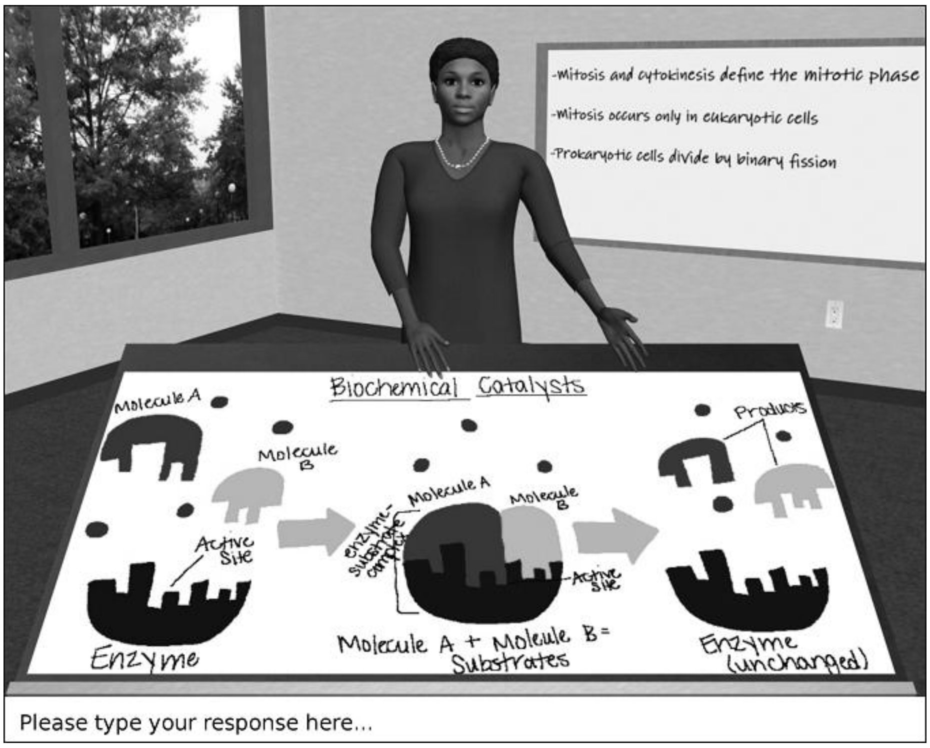

-Mitosis and cytokinesis define the mitotic phase
-Mitosis occurs only in eukaryotic cells
-Prokaryotic cells divide by binary fission
Biochemical Catalysts
Molecule A
Molecule B
Active Site
Enzyme
enzyme-substrate complex
Molecule A
Molecule B
Active Site
Molecule A + Molecule B = Substrates
Products
Enzyme (unchanged)
Please type your response here...

Figure 1: XNAgent running in the Guru intelligent tutoring system.

We argue that XNAgent fulfills many if not all of the design goals for GIFT authoring components [11]. XNAgent decreases the effort of authoring ECAs through its 3D authoring pipeline. Similarly it decreases the skills required for authoring ECAs by making authoring a drag-and-drop process, rather than a pixel-by-pixel process. XNAgent's animation framework allows authors to organize their knowledge about pedagogical animations and helps structure pedagogical animations. Perhaps most importantly in a research environment, XNAgent supports rapid prototyping of ECAs with different properties (gender, size, or clothing) for different pedagogical roles (teacher, mentor, or peer). XNAgent supports standards for easy integration with other software as a Windows form control. By cleanly separating domain-independent code from specific 3D model and animation content, XNAgent promotes reuse. Finally XNAgent leverages open source solutions. Not only is XNAgent open source, but every element in its 3D authoring pipeline either has a freeware version or is free for academic use. Moreover, the recent version of MonoGame, an open source implementation of XNA, promises to make XNAgent cross platform to desktop and mobile devices beyond the Windows desktop.

5 Conclusion

In this paper we have described the XNAgent platform for developing embodied conversational agents. Unlike existing ECA platforms that require either low level graphics programming or the use of complex game engines, XNAgent is written using a high level framework (XNA). Our contribution to this research area is in showing how to implement appropriate speech, gesture, and facial expression using skeletal and morph animation via vertex shaders, motion capture, and event-driven animation. We argue that the XNAgent platform fulfills most of the authoring design goals for GIFT with respect to authoring ECAs. It is our hope that XNAgent will be used by adopters of GIFT to facilitate creation of dialogue based tutoring systems that use ECAs.

6 Acknowledgments

The research reported here was supported by the Institute of Education Sciences, U.S. Department of Education, through Grant R305A080594 and by the National Science Foundation, through Grant BCS-0826825, to the University of Memphis. The opinions expressed are those of the authors and do not represent views of the Institute or the U.S. Department of Education or the National Science Foundation.

7 References

1. Nass, C., Steuer, J., Tauber, E.R.: Computers are social actors. In: Proceedings of the SIGCHI conference on Human factors in computing systems: celebrating interdependence. CHI ’94, New York, NY, ACM (1994) 72–78
2. Dunsworth, Q., Atkinson, R.K.: Fostering multimedia learning of science: Exploring the role of an animated agent’s image. Computers & Education 49(3) (November 2007) 677–690
3. Heloir, A., Kipp, M.: Real-time animation of interactive agents: Specification and realization. Applied Artificial Intelligence 24 (July 2010) 510–529
4. Graesser, A.C., Lu, S., Jackson, G.T., Mitchell, H., Ventura, M., Olney, A., Louwerse, M.M.: AutoTutor: A tutor with dialogue in natural language. Behavioral Research Methods, Instruments, and Computers 36 (2004) 180–193
5. Rowe, J.P., Mott, B.W., W. McQuiggan, S.W., Robison, J.L., Lee, S., Lester, J.C.: CRYSTAL ISLAND: A Narrative-Centered learning environment for eighth grade microbiology. In: Workshops Proceedings Volume 3: Intelligent Educational Games, Brighton, UK (July 2009) 11–20
6. Lester, J.C., Voerman, J.L., Towns, S.G., Callaway, C.B.: Deictic believability: Coordinating gesture, locomotion, and speech in lifelike pedagogical agents. Applied Artificial Intelligence 13 (1999) 383–414
7. Damian, I., Endrass, B., Bee, N., André, E.: A software framework for individualized agent behavior. In: Proceedings of the 10th international conference on Intelligent virtual agents. IVA’11, Berlin, Heidelberg, Springer-Verlag (2011) 437–438

8. Heloir, A., Kipp, M.: Embr — a realtime animation engine for interactive embodied agents. In: Proceedings of the 9th International Conference on Intelligent Virtual Agents. IVA ’09, Berlin, Heidelberg, Springer-Verlag (2009) 393–404
9. de Rosis, F., Pelachaud, C., Poggi, I., Carofiglio, V., De Carolis, B.: From Greta’s mind to her face: modelling the dynamics of affective states in a conversational embodied agent. International Journal of Human-Computer Studies 59(1-2) (2003) 81–118
10. Thiebaux, M., Marsella, S., Marshall, A.N., Kallmann, M.: SmartBody: behavior realization for embodied conversational agents. In: Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems - Volume 1, Estoril, Portugal, International Foundation for Autonomous Agents and Multiagent Systems (2008) 151–158
11. Sottilare, R.A., Brawner, K.W., Goldberg, B.S., Holden, H.K.: The generalized intelligent framework for tutoring (GIFT). Technical report, U.S. Army Research Laboratory â” Human Research & Engineering Directorate (ARL-HRED) (October 2012)
12. Cawood, S., McGee, P.: Microsoft XNA game studio creator’s guide. McGraw-Hill Prof Med/Tech (2009)
13. Ekman, P., Rosenberg, E.: What the face reveals: basic and applied studies of spontaneous expression using the facial action coding system FACS. Series in affective science. Oxford University Press (1997)
14. Noh, J., Neumann, U.: A survey of facial modeling and animation techniques. Technical Report 99-705, University of Southern California (1998)
15. N’Diaye, K., Sander, D., Vuilleumier, P.: Self-relevance processing in the human amygdala: gaze direction, facial expression, and emotion intensity. Emotion 9(6) (December 2009) 798–806
16. Pelachaud, C., Badler, N., Steedman, M.: Generating facial expressions for speech. Cognitive Science 20 (1996) 1–46
17. Gregory, J.: Game engine architecture. A K Peters Series. A K Peters (2009)
18. Alexandru-Cristian, P.: Communist animation library for xna 4.0 (December 2010)
19. McNeill, D.: Hand and mind: what gestures reveal about thought. University of Chicago Press (1992)
20. Olney, A., D’Mello, S., Person, N., Cade, W., Hays, P., Williams, C., Lehman, B., Graesser, A.: Guru: A computer tutor that models expert human tutors. In Cerri, S., Clancey, W., Papadourakis, G., Panourgia, K., eds.: Intelligent Tutoring Systems. Volume 7315 of Lecture Notes in Computer Science., Springer Berlin / Heidelberg (2012) 256–261

Authors

Andrew Olney is presently an assistant professor in the Department of Psychology at the University of Memphis and the associate director of the Institute for Intelligent Systems. His primary research interests are in natural language interfaces. Specific interests include vector space models, dialogue systems, grammar induction, robotics, and intelligent tutoring systems.

Patrick Hays is a research assistant at the University of Memphis. He is a recent graduate with a BA in Psychology. Patrick's work focuses on 3D animation, 3D modeling, graphic arts, and human-computer interaction.

Whitney Cade is a graduate student in the Department of Psychology at the University of Memphis. Her research interests include intelligent tutoring systems, expert tutors, pedagogical strategies, 3D agents, and machine learning.