Agent-Based Practices for an Intelligent Tutoring System Architecture

Keith Brawner(✉), Greg Goodwin, and Robert Sottilare

Army Research Laboratory, Adelphi, USA
{keith.w.brawner.civ, gregory.a.goodwin6.civ,
robert.a.sottilare.civ}@mail.mil

Abstract. The Generalized Intelligent Framework for Tutoring (GIFT) project is partially an effort to standardize the systems and processes of intelligent tutoring systems. In addition to these efforts, there is emerging research in agent-driven systems. Agent-based systems obey software and messaging communication protocols and accomplish objectives to the original system, but have different architectural structure. This paper describes the upcoming research changes for GIFT, from a module-driven system to an agent-driven system, the reasons for wanting to do so, the advantages of the change, some initial technical approaches which encapsulate current functionality, and the types of research that this change will enable in the future.

Keywords: Intelligent tutorins systems · Agent based systems · eLearning · mLearning · Software-as-a-service

1 Introduction

The Generalized Intelligent Framework for Tutoring (GIFT) is a science and technology project whose goal is to reduce the technical cost time and skills required to author intelligent tutoring systems (ITS) and to increase the effectiveness of automated instruction in new domains [1]. This is accomplished through the implementation of four primary principles: domain independence, componentization, generalized ITS authoring tools, and automation. A core design philosophy of GIFT is to separate domain-dependent from domain-independent components. This allows the same tutoring infrastructure to be used to train car repair, or medical triage, or team situational awareness, and reduces the number of unique ITS components. Under the principle of componentization, the modules in GIFT, their functions and the messages exchanged between them are standardized to simplify tutor creation and modification. Using this design, an ITS author does not need to have computer programming or instructional system design skills to create a functional ITS.

Componentization simplifies design and processes through constrained input/output sets. The constraint of these input/output sets, in turn, renders them easier to automate or self-construct. Recent projects involving GIFT attempt to build a "policy" which maps inputs to outputs in a few fashions. As a few examples, an instructional policy may recommend immediate or delayed feedback based on a profile of a learner, a learner profile may choose the frequency with which to communicate information to an

D.D. Schmorrow and C.M. Fidopiastis (Eds.): AC 2016, Part II, LNAI 9744, pp. 3–12, 2016.
DOI: 10.1007/978-3-319-39952-2_1

instructional module, or a model of the domain may choose specific implementations of feedback (in game, avatar-driven, flashing, etc.). These policies can be constructed, based on historical data, via software process or modified based on the observations of student state and instructional effects.

Just as componentization simplifies the engineering design space of its components, the creation of policy-driven input/output functions is easier to automate. Techniques for automatically creating agent-driven input/output policies are well studied in the reinforcement learning literature, including techniques such as neural networks, entropy-reducing decision trees, Markov processes, and others. However, in a system like GIFT with disparate processes, input/output options, and data sources, these techniques result in policies that are customized towards each module. A general-purpose solution for optimizing the finite-action set is preferred. Over time, ITSs have developed from custom-crafted systems into systems of interchangeable parts and into systems of software-customized policies. In this paper we will outline the next step of ITS evolution into true agent-based systems, which construct their own policies.

This paper briefly reviews the history of the creation of agent-based ITS, agent-based frameworks for educational purposes, how an agent-based and policy-driven system can be constructed over top of an existing modular system (i.e. GIFT), the advantages and disadvantages of doing so, and initial planning steps of implementation. The paper presents draft designs for interoperability and communication as well as sample technologies for general-purpose adaptation in the presence of data for the purpose of gaining knowledge or optimizing instruction.

2 Existing Work with Agent Frameworks and Intelligent Tutoring Systems (ITS)

In order to frame the discussion of the emergence and development of an agent-based system, there should be a discussion of what defines an agent, and how the term is used. Franklin and Graesser present the essence of agency as having components of sensing the environment, acting upon it, having a sense of time, and pursuing goals [2]. They also state that such agents can be composed of multiple sub-agents, each meeting the above criteria. Based on this definition, the modules of GIFT do not currently have all the traits of agents, but do meet some of the criteria. GIFT modules have information from the environment (the system) and produce outputs, however they individually do not always have a means of assessing the impact of that output on the environment (knowing if they have achieved a goal). A plug-in, or within-module process, which is able to track within-module data, determine the module output, serve the goal of the module (usually modeling), and adjust itself over time would enable GIFT modules to meet the criteria of agents set for by Franklin and Graesser.

The idea of using a framework of cooperative agents as part of an intelligent tutoring system is not new. The problems faced by the field in 1995 were similar to the problems faced with modern-day systems (e.g., lack of reuse, lack of standards, and lack of flexibility). Overcoming these problems by creating a modular framework of agents to provide tutoring capabilities, was the objective of the Generic Instructional Architecture (GIA) project [3]. The GIA and GIFT projects share similar goals,

but where GIFT attempts to modularize and then automate, GIA attempts automation directly. In a system such as GIA, the Agent Communication Language (ACL) forms the backbone of communication, with agents advertising their functionalities, availability, and ontology for communication. However, the lack of call for specific agents, or specific groups and types of agents, adds, rather than cuts, from developmental time of a system. Not specifying the required agents, policies, and functions, results in a lack of development for specific system instantiation. This weakness is present in the lack of adoption of the system for in-the-wild tutoring.

Gascueña presents another agent-based system composed of a Student, Domain, Pedagogical, and Educational Module, similar to structure adopted by the GIFT project [4]. Each module in Gascueña's system could have multiple agents and each agent its own software program which can provide recommendation on the output or action of the total module. Examples of these agents include Pedagogical agents for Preferences, Accounting, Exercises, and Tests. Other researchers have proposed similar designs that include Assistant, Evaluation, and Pedagogical agents [5], or may divide these agent capabilities into services such as a Domain, User, Adaptation, and Application Service [6]. GIFT handles functions such as Adaptation and Application through optional and additional plug-in services, which are updated based on outcomes from processing system outputs. However, currently neither GIFT nor Gascueña provide a suggestion of conflict management between recommend agent actions. Overly specifying functions, agents, and policies results in a tightly bound system where few functions can be added.

Inside of the framework of an educational system, additional agents beyond the above are needed that function to enable content/training delivery to a student. The previously mentioned agents only function as part of online instructional management while greater functionality is needed for full individualization. Examples of the types of needed agents are provided by Lin et al. [7] who describe a series of external agents running outside the core instructional loop. These include an adviser agent, which advises the next content to view, a collaboration agent, for collaborating with peers, a course planning agent, which plans a student path through a course, a course delivery agent, which accommodates different delivery styles, and several others. These agents are managed by an agent management and deployment service. Other work, such as Regan's Training and Learning Architecture (TLA) [8], or the Personalized Assistance for Learning (PAL) [9] effort describe the system-of-systems approach to agent construction. Further work examples can be seen in the Dynamic Tailoring System (DTS), which provides an agent derived from the Soar cognitive architecture programmed for the purpose of pedagogy and scaffolding [10]. These systems show the flexibility required for addition, but, like GIA, cannot function as a pure delivery system.

Each of these systems discussed so far has either (a) not specified the functions which are required for agents (e.g. recommendation agent), or (b) overly specified the information for agents (e.g. domain hinting agent). The problem with not specifying the requirements for agency adequately is that it makes it impossible for the system to be adopted as an agent-based system. On the other hand, when the functions are too constrained, they become less flexible and this limits the manner of system expansion. As we look to ways of transforming GIFT to an agent-based system, it is important to provide specifications for the addition of new agents without being overly restrictive on how those agents will function within the GIFT environment. One of the primary

advantages of GIFT is that very little new ITS functionality is dictated, existing functions are standardized, and the ability to expand is clearly defined.

3 Online Agents and Online Learning

Terms such as "microadaptive instruction" or "inner instructional loop" may refer to one or more tutoring actions taken with the student. These actions may include hinting, prompting, metacognitive reflection, and others. All these actions are provided to help the learner overcome an impasse in problem solving. For the systems able to assign multiples of these actions (i.e. a service which recommends hinting conflicting with a service recommending prompting), there is a mitigation function to help make sure that all available instructional actions are not taken simultaneously. As an example, AutoTutor Lite uses a cycle of pump → hint → prompt → assert as the student progressively needs more content or assistance [11]. Generally speaking, these online actions are taken in order to nudge, rather than didactically instruct, the student towards the preferred manner of thinking on problem solving. The online and real-time components of these decisions deal with data and decisions which are of small grain size, or of small individual impact.

3.1 Learning Agents

The typical approach for the creation of an intelligent tutoring system follows a relatively simple process. First, a system is developed and deployed into a production setting while using a baseline (usually manually created) process. Second, the learner data for this system is analyzed using an approach such as a Bayesian network, reinforcement learning algorithm, or equivalent. Third, the findings are used to improve the initial models, which serve as the baseline for the next version of the system. This collect-model-update cycle varies for different ITS. Many systems have followed this approach for affective learner modeling [12], domain modeling [13], or instructional modeling [14].

An alternative approach for creating an ITS is to develop agents which can learn in the presence of new data. While this approach reduces human control of the final model, and perhaps reduces scientific validity, it generally produces improved model quality [14]. This is because an agent-based ITS can continuously improve its underlying models based on ground truth observations, customized to the actual content delivered. An example of such a system is one which begins to build an initial model from available data/decisions, modifies or customizes the model for a new student, and puts the new model into practice immediately. It makes most use of policy information compatible with multiple instructional domains, allows configuration from observed evidence, and can potentially share this knowledge with other similar agents and processes.

3.2 Potential Implementations

GIFT is based on the learning effect chain, whereby learner data informs learner states which inform instructional strategy selection which influence learning gains [1]. This chain is instantiated in the GIFT software as a Domain Module which informs a Learner Module which informs a Pedagogical Module which selects instructional strategies which are implemented as instructional tactics in the Domain Module. Each portion of data is passed in real-time in order to deliver content to the student. The most basic implementation of agent policies is to add a policy-handling component to the existing structure. The left side of Fig. 1 shows how these items exist in current GIFT architecture, while right side of Fig. 1 shows the addition of policy information. Figure 2 shows an in-depth implementation of the concept of policy overlay to existing functionality.

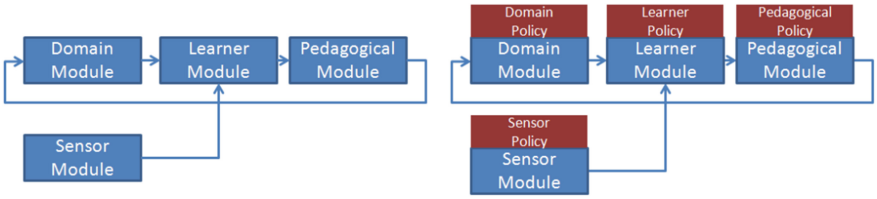

Fig. 1. The addition of a policy component to the existing GIFT structure in order to accommodate real-time agent functionality.

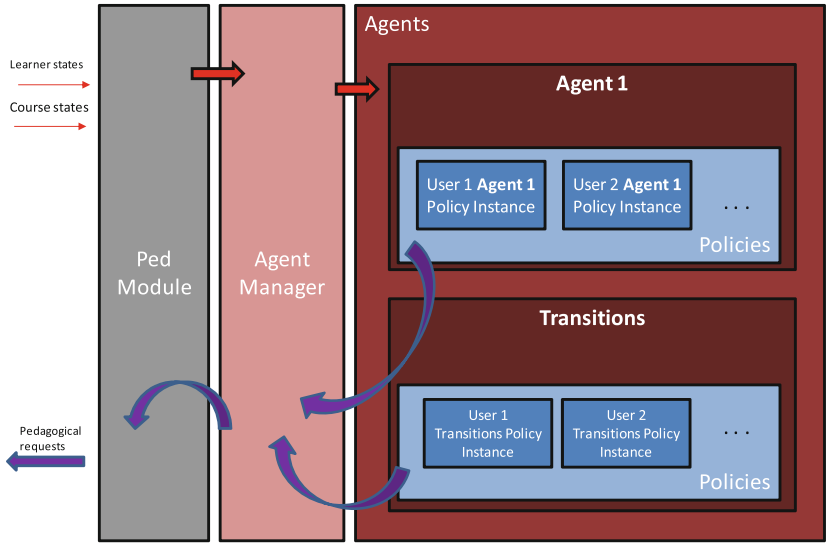

Fig. 2. Addition of a policy component shown in greater detail within an instructional module (pedagogical policy)

However, as agents and agent-based practices are added to GIFT, a manner of representation is needed, this is especially prevalent for a changing structure over time, such as when a policy is learned for a specific class of users over a large series of interactions. A manner of autonomously changing system behavior can be added through the addition of a policy component as an overlay to the module. In this manner, the initial module configuration (via various configuration files) can be referenced in mutable policy, changed in the presence of new data, and de-conflicted in the event of conflict. The addition of this type of policy information enables additional features and functions to be supported. Examples of these behaviors for specific model instantiations are presented in Table 1.

Table 1. Listing of modules, example module policies, and data sources for model updates

Module	Policy example	Cause for alteration
Domain module	Selection of instructional tactic, selecting a shorter segment instead of a longer one	Time available for student
Domain module	Generation of an after action review (AAR) as part of a sequence of content	Student actions
Learner module	Prediction of student state, based on difficulty of concepts (mined from previous example)	Updated assessment of individual student, classroom, or introduction of new data source
Pedagogical module	Selection among conflicting instructional policies (“hint” or “prompt”)	Varying observed effects dependent on authored quality

4 Offline Agents and Services

In contrast to online agents which perform as part of the real-time decision loop for managing instructional decisions and delivery, there is a need for offline agents and services. The factors driving the need for an offline agent or service include output that takes too much time to produce in real-time, the size of the instructional decision is larger than a single time-step, a need to reach back to an alternative data source, or output that is appropriate for multiple online modules.

The expansion of the GIFT architecture shown in Fig. 1 allows for a single policy which links to other policy components in other modules. As an example, a policy of “progressive mastery” (teaching one concept to mastery, then moving to the second concept) can be linked across the pedagogical and domain modules in order to coordinate instruction. Such a policy would require components to be present in both of the key areas, initialized at runtime, and have a communication component for synchronization. This policy add-on enables two core functionalities: switching policies based on observed results, and having policies based on multiple module functionality. This addition of a policy component, although simple, provides significant capability.

4.1 Functions and Features

The Advanced Distributed Learning (ADL) group has put together a number of technologies for providing underlying standards for the next generation of intelligent tutoring systems. Some examples of the offline services offered by agents include items such as the Learning Registry, which is an indexing service which extends to a number of publishers for the point of being able to integrate content, as a reusable learning object, into various learning systems. On top of this layer, it is anticipated that offline agents can be constructed to provide additional information on the content (e.g. metadata, paradata, learning object descriptors) or to provide content-matching services between the users and the content. The accumulated products of these offline agents would have be available to the online services and agents such that online services could make use of them. As new offline agents are developed to provide new products, online ITS agents would simply require new policies to enable them to make appropriate use of those new products.

Finally, it is possible that some policies may be developed offline but utilized online. For example, some instructional decisions may require information gathered across several modules, or may be based on products compiled from other offline agents but that inform decisions made in an online environment. Such information may require the use of a policy component which is constructed offline, but shared in an online environment. This type of function would learn from many different instructional decisions from many learners in many domains, with varying amount of learner history.

4.2 Potential Implementations

Unlike online agent, policies, and services, GIFT has no prescribed structure for the implementation and communication of offline components. Provided that formats and configurations are suitable for online instantiations, there is no requirement for offline standardization at this time. This leaves the offline implementations free to use all data

Table 2. Module-specific services and policies for GIFT

Module using agent/service	Service/policy example	Data source
Domain module	Creation of customized scenarios around instructional needs	Bank of specific, instructionally valid, examples
Domain module	Pre-generation of hints, loaded in for selection at runtime	Domain-specific text corpus, learning objectives
Domain module	Recommendation service	Based on learner learning goals
Learner module	Modeling of competencies and mapping to taught concepts	Learner record store, trace data of transferability of skills from previous sessions
Pedagogical module	Update to instructional model for a specific domain	History of many learners across many courses

available and all standard formats. The limitations on offline services are more relaxed, and examples of various potential services are provided in Table 2.

5 Efforts Towards Unification and an Interoperable Learning Ecosystem

One of the core concepts behind the GIFT architecture has been to unify the various commonly used portions of intelligent tutoring systems. This effort now allows for the incorporation of agents and policy components which can be expanded across each of its core modules, and across offline and online processes. However, GIFT is not unique in its role to attempt standardization among various services. Techniques such as those implemented in the Open Agent Architecture (OAA) serve as brokering functions between application agents, meta-agents (facilitate coordination with other agents), and user agents. The typical functions from this and similar systems from various frameworks include the user side (front end), processing side (back end), and functionality side (module purposes).

Not only do we seek to unify commonly used portions of intelligent tutoring systems, but also to enable GIFT to be interoperable with the larger learning ecosystem. As described above, the ADL group has been developing standards and agents (e.g., PAL, TLA) for such a purpose. As more of these services are developed by various groups, the risk is that interoperability challenges will multiply. In such an increasingly diverse and complex ecosystem, the need for intelligent agents will only grow. Agents will be needed to broker the needs of systems like GIFT and the potential catalogue of services available in the larger learning ecosystem. As an example, the Virtual Human Toolkit (VHTk) provides the functionality to model the various aspects of virtual humans. This functionality includes aspects and services such as speech processing, emotional modeling of the learner, emotional modeling of the virtual human, the gestures of the virtual human, rendering, and other services. It has been used in many tutoring or learning programs [15]. This particular product/package provides for the integration of many functional back-end features required for human modeling. Virtual humans serve as a good example of agents or agent based systems that would routinely interact with an ITS like GIFT. Currently GIFT has limited ability to operate with the VHTk through basic service calls to supporting functionality. However, in a more agent based GIFT, this would be accomplished much more easily by creating the necessary policies via the VHTk, having them represented within modules, and delivering them to the students.

A final example of services being developed in the larger learning ecosystem is the Generalized Learning Utilities (SuperGLU). This is a collection of back-end services and features developed through Office of Naval Research initiative funding to provide functionality for offline processes and policies [16]. These services include the communication of learner performance data, through the xAPI standard [8], the ability to add agents without knowledge of the other agent components, and the overall integration of various tutoring services and data structures. SuperGLU is additionally intend to work with LearnSphere, as an effort to unite the storage and processing of data generated by tutoring systems.

6 Conclusion and Future Research

Transforming GIFT into an agent-based system has both advantages and some disadvantages. The anticipated benefits include the simplification of development through automating the creation and refinement of underlying models and to simplify interoperability with a larger learning ecosystem. A potential disadvantage is that as agents take over the task of refining models and managing interactions among different systems, humans become monitors and managers of system development rather than architects. There is some risk that this would result in humans being less effective or efficient in troubleshooting or in making system-wide changes that might improve performance, considering the additional complications of having a system which changes over time.

The benefits of making GIFT into a learning agent-based system are enticing. Developing learner and pedagogy models through research is time and resource intensive. Developing intelligent learning agents that can in effect figure out what works and automatically improve those models could potentially save huge amounts of time and resources. Furthermore the larger learning ecosystem is an ever changing entity. Using intelligent, learning agents to constantly search that ecosystem and establish interfaces between GIFT and new and changing services and agents in that ecosystem would also save substantial resources.

On the other hand, the risks of converting GIFT (or any system) into an agent-based system should not be overlooked. As humans manage system evolution by changing system rewards through new or revised policies, there is always the risk that those changes may not have the intended effect on learning performance. For example, suppose a policy was created to reduce time to train one module in a course without decreasing performance in that module. Suppose the policy change was successful, but a second order consequence was that there was a decrement in student performance on another module in the same course. Single-minded focus on limited metrics can result in unintended effects on other, unmeasured, items. To troubleshoot such issues, it will be necessary for system managers to be able to understand why those policy changes had those effects. This would not be a simple problem to solve and it would be more difficult if managers could not see or understand what the agent had done to reduce training time for the targeted module. Add to this the possibility that multiple agents could be making changes to different models at the same time, and one can see that untangling the causes and effects on overall system performance could be quite difficult. Thus it will be important to develop ways for agents to provide system managers with human-readable reports on changes made to the system as well as other agent activities. These and perhaps other challenges lie ahead as we implement an agent-based design for GIFT.

References

1. Sottilare, R.A., Brawner, K.W., Goldberg, B.S., Holden, H.A.: The Generalized Intelligent Framework for Tutoring (GIFT). US Army Research Laboratory (2012)
2. Franklin, S., Graesser, A.: Is it an agent, or just a program?: a taxonomy for autonomous agents. In: Jennings, N.R., Wooldridge, M.J., Müller, J.P. (eds.) ECAI-WS 1996 and ATAL 1996. LNCS, vol. 1193, pp. 21–35. Springer, Heidelberg (1997)
3. Cheikes, B.A.: Gia: an agent-based architecture for intelligent tutoring systems. In: Proceedings of the CIKM 1995 Workshop on Intelligent Information Agents. Citeseer (1995)
4. Gascueña, J.M., Fernández-Caballero, A.: An agent-based intelligent tutoring system for enhancing e-learning/e-teaching. Int. J. Instr. Technol. Distance Learn. **2**, 11–24 (2005)
5. Lopes, C.R.: A multiagent architecture for distance education systems. In: Null, p. 368. IEEE (2003)
6. Chepegin, V., Aroyo, L., De Bra, P., Houben, G.: CHIME: service-oriented framework for adaptive web-based systems. In: Conferentie Informatiewetenschap, pp. 29–36. Citeseer (2003)
7. Lin, F., Holt, P., Leung, S., Hogeboom, M., Cao, Y.: A multi-agent and service-oriented architecture for developing integrated and intelligent web-based education systems. In: International Workshop on Applications of Semantic Web Technologies for E-Learning at the International Conference on Intelligent Tutoring Systems (2004)
8. Regan, D.A.: The training and learning architecture: infrastructure for the future of learning. In: Invited Keynote International Symposium on Information Technology and Communication in Education (SINTICE), Madrid, Spain (2013)
9. Regan, D., Raybourn, E.M., Durlach, P.: Learning modeling consideration for a personalized assistant for learning (PAL). In: Design Recommendations for Adaptive Intelligent Tutoring Systems: Learner Modeling, vol. 1 (2013)
10. Wray, R.E., Woods, A.: A cognitive systems approach to tailoring learner practice. In: Proceedings of the Second Annual Conference on Advances in Cognitive Systems ACS, p. 38 (2013)
11. D'mello, S., Graesser, A.: Autotutor and affective autotutor: learning by talking with cognitively and emotionally intelligent computers that talk back. ACM Trans. Interact. Intell. Syst. (TiiS) **2**, 23 (2012)
12. Ramirez, G.A., Baltrušaitis, T., Morency, L.-P.: Modeling latent discriminative dynamic of multi-dimensional affective signals. In: D'Mello, S., Graesser, A., Schuller, B., Martin, J.-C. (eds.) ACII 2011, Part II. LNCS, vol. 6975, pp. 396–406. Springer, Heidelberg (2011)
13. Duong, H., Zhu, L., Wang, Y., Heffernan, N.: A prediction model that uses the sequence of attempts and hints to better predict knowledge: "better to attempt the problem first, rather than ask for a hint". In: Educational Data Mining 2013 (2013)
14. Liu, Y.-E., Mandel, T., Brunskill, E., Popovic, Z.: Trading off scientific knowledge and user learning with multi-armed bandits. In: Educational Data Mining, London, UK (2014)
15. Lane, H.C., Hays, M., Core, M., Gomboc, D., Forbell, E., Auerbach, D., Rosenberg, M.: Coaching intercultural communication in a serious game. In: Proceedings of the 16th International Conference on Computers in Education, pp. 35–42 (2008)
16. Benjamin, N., Xiangen, H., Graesser, A., Zhiqiang, C.: Autotutor in the cloud: a service-oriented paradigm for an interoperable natural-language its. J. Adv. Distrib. Learn. Technol. **2**, 49–63 (2014)