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 gate-

way 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.