An Intelligent Tutoring Framework for Simulation-based Training

Dave Gomboc[a], H. Chad Lane[a], Mark Core[a], Ashish Karnavat[a], Daniel Auerbach[a], Milton Rosenberg[a]

[a]*Institute for Creative Technologies, University of Southern California, U.S.A.*

gomboc@ict.usc.edu

Abstract: Truly generic, reusable intelligent tutoring software frameworks remain elusive. As part of our effort to develop ITSs for simulations, a software framework with minimal dependencies on domain specifics has emerged. Herein, we describe this framework, its functionality, components, configurability, and use of natural language generation.

Keywords: Intelligent tutoring systems, architecture, framework, simulation, training

Introduction

The Intelligent Guided Experiential Learning (IGEL) project focuses on supporting students' learning while practicing skills and problem solving using simulations. A primary goal of the IGEL framework is to minimize dependency upon domain specifics. IGEL is integrated into ELECT BiLAT [5, 10], an immersive simulation teaching cross-cultural trust-building and negotiation strategies. Our work is informed by prior research [4] that used other simulators such as OneSAF Objective System [2] and Full Spectrum Command.

Simulation runs are represented as a sequence of actions and the resulting changes in the state of the world. In BiLAT, states correspond to mental representations of meeting partners and deals being made; actions correspond to meeting actions (e.g., saying something, making an offer in a negotiation).

The learning objectives (LOs[1]) that IGEL attempts to impart are encoded hierarchically. For example, BiLAT has a top-level LO that represents all aspects of socialization. One child of this LO represents "making small talk", which in turn is subdivided further. Within the LO tree, direct siblings are more related than more distant nodes.

Expert opinions (EOs), formed by the expert model both in anticipation of and in reaction to student actions, indicate what guidance can be provided. In BiLAT, the user action making small talk concerning culture yields positive evidence that the student understands the LO regarding socializing by discussing culture. Evidence against comprehension of a different LO regarding conversational pacing is also yielded if such small talk is mistimed.

IGEL components run in their own processes and communicate over TCP/IP, enabling Wizard of Oz experiments and other multi-machine setups. Generic code employs Java reflection to convert in-memory objects to and from canonical XML and RDBMS representations as required. In Figure 1, we classify components as either domain-dependent or -independent. Arrow labels indicate most message types sent between components. Domain knowledge is often stored as external data to preserve code domain-independence.

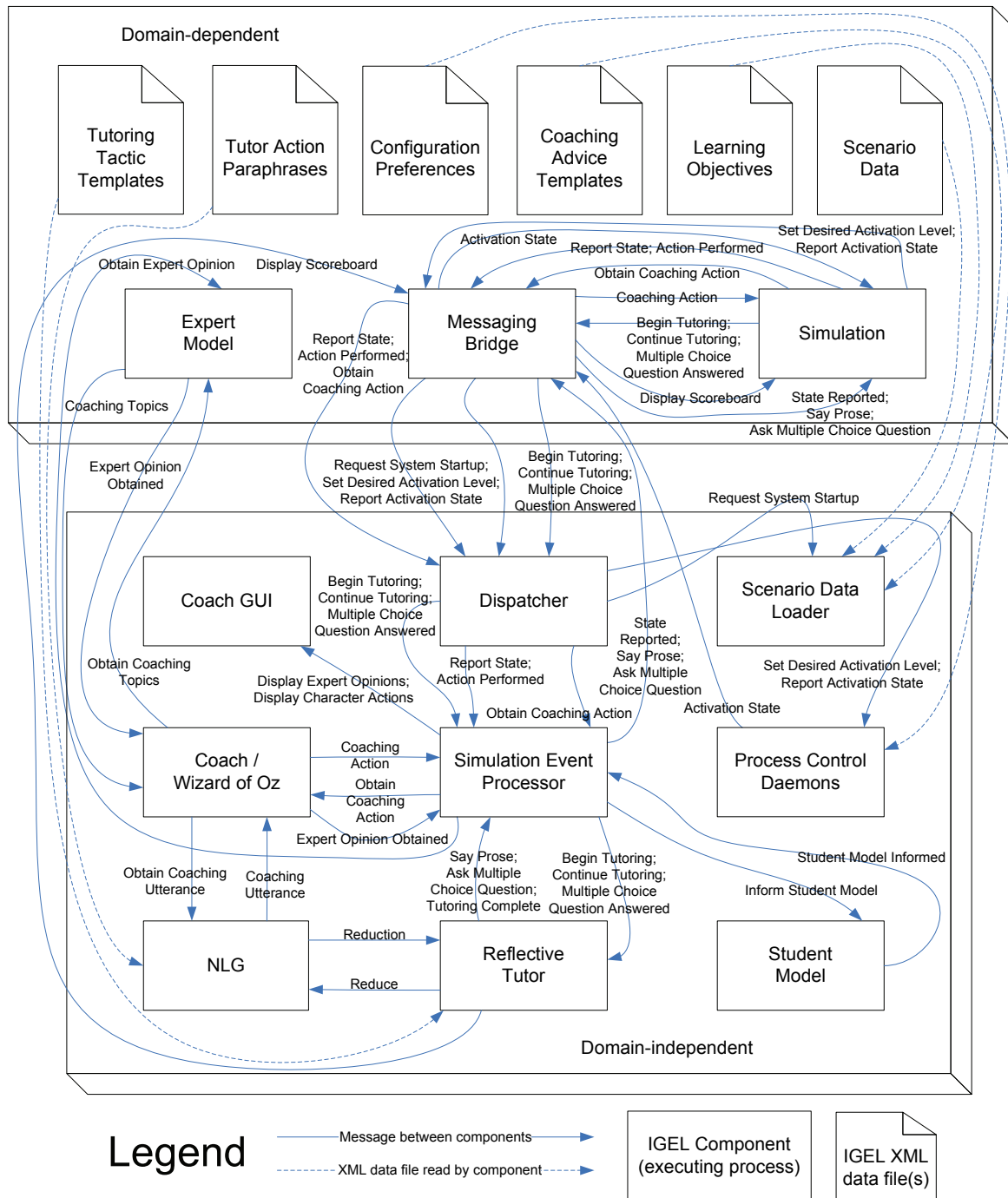[1] Learning Objectives are not to be confused with the IEEE LTSC's Learning Objects.

Figure 1: IGEL System Architecture

1. Domain-dependent framework components

1.1 Expert Model

The expert model performs domain-dependent reasoning on behalf of the tutoring system. Its two major roles are as *critic* and *problem solver*: assessing student actions at a state, and recommending actions to perform given a state. Both of these procedures create expert opinions that are recorded in our database for our coach and tutor to make use of.

Integrating the expert model with, or even within, the simulator is desirable to ensure the fidelity of the expert model to the simulation, but requires tight collaboration between the simulation and ITS builders. Organizational and geographical disparity, differences in

development funding and timing, and incentive disparity between developers may make such collaboration infeasible.

Nothing precludes the expert model from itself hiding multiple, specialized expert models. BiLAT's expert contains two modules that specialize on culturally appropriate meeting conversation and negotiation (haggling), respectively. Experts could also be used collectively (e.g., each expert contributing a vote to the final decision): see Littman et al. [9] for an effective use of this technique for solving crossword puzzles.

1.2 Messaging Bridge

If the simulation's and IGEL's representation of scenario data, states, actions, and software control messages differ, then format conversion is necessary. The bridge adjusts messages appropriately on the fly during communication between IGEL and the simulation.

2. Domain-independent framework components

In BiLAT, we distinguish between coaching (giving advice during problem solving) and reflective tutoring (review subsequent to problem solving), because we believe the types of student interactions appropriate in each case are different. We reason that the student, while actively engaged with the simulation, will typically lack the additional cognitive resources required to answer questions or have time to comprehend lengthy guidance. In contrast, the learner's complete attention can be focused upon our after-action review (AAR), where the reflective tutor engages the student in extended dialogue regarding their experience. Katz et al. [7, 8] goes some way towards supporting this reasoning.

In other domains, the distinction between the roles of coach and tutor may be less clear-cut, because the training opportunities are not as obviously separable. In such cases, both roles could be assigned to the same component.

2.1 Coaching

IGEL requires notification from the simulation when the student has opportunity to perform one or more actions, when the user interface indicates the student intends to or has in fact performed any actions, and also when the student requests a hint, so that the coach may choose to give hints and feedback to the student at any of these appropriate moments.

IGEL supports multiple versions of our scaffolding algorithm inspired by cognitive apprenticeship [1]. The coach provides hints and feedback very frequently at first, then pulls support away gradually as the student demonstrates increasing skill, until the student succeeds without assistance. We support both deterministic and probablistic scaffolding; both the initial rate of guidance and how quickly guidance is faded are configurable. Also, strategies to always or never attempt to give advice assist debugging. Advice provision is never guaranteed: we cannot insist that all potentially relevant remarks be authored.

At any advice-giving moment, IGEL can choose to either comment about a learning objective, or directly describe why a particular action was, was not, or would be beneficial. Our default content selection strategy is to give LO-based commentary when it has not been given previously for LOs relevant to the situation at hand, and to give action-based commentary otherwise (assuming the action-based commentary has not been given previously). It is also possible for the instructor to specify that commentary for specific LOs should be prohibited or preferred while the coach is running via the coach's internal GUI. The NLG templates used for coaching commentary are specified in spreadsheets to make them readily authorable by non-programmers.

Initial research regarding the coach's effectiveness is reported elsewhere within these proceedings [11]. A Wizard of Oz (WoZ) version of our coach provides its user full access to the simulation history as perceived by IGEL, the library of possible actions the user can perform, and what the automated coach would have said to the student were it being used.

2.2 Simulation Event Processor

IGEL's main loop during the simulation's execution lies in the simulation event processor. The processor coordinates the activities of the expert model and the coach with the simulation, ensuring that each component is signalled to provide its input at the appropriate time.

2.3 Student Model

Our rudimentary, work-in-progress student model is based upon the hierarchical set of LOs defined for the domain. IGEL records all states encountered, actions performed, and each action's accordance or discordance with relevant LOs, as determined dynamically by the expert model. The student model is retained through multiple simulation executions during the same simulation software instantiation, but not yet between program instantiations.

2.4 Reflective Tutoring

At the conclusion of a distinct problem-solving episode in the simulation, our reflective tutor conducts an AAR with the student that reviews the student's performance. The AAR provides an opportunity for our software to discuss topics more deeply and interact with the student regarding decisions made while practicing with the simulation.

The AAR's agenda is derived from the EOs formed during practice. Discussion topics may be prioritized by different metrics, including error magnitude and action chronology. EOs sharing LOs are clustered so that closely related issues relating to different points of time during the simulation are juxtaposed in the AAR. Ideally, IGEL would also create EOs during the AAR. Student selection of a wrong answer to a multiple choice question that the tutor presents provides evidence that the student misunderstands not only the topic of discussion, but also the distractor wrongly selected. Updating our student model to reflect this mistake would ideally cause the tutor to consider addressing this revealed misconception.

While prototypes of our tutor used JShop2 [6] and Jess [3], currently IGEL's tutoring capabilities stem from a simple reactive planner written in Java. Tutoring tactic templates are specified in an external data file that trained non-programmers may modify, allowing tutoring improvements without recompilation. Available tutoring tactics include making remarks and asking multiple-choice questions about why something the student did during the simulation was incorrect. Distractors for multiple-choice questions may be dynamically selected based upon the LO hierarchy (but not yet the student model), or hand-authored, at the content author's discretion. Plausible future extensions include a tactic to ask the student to "act again", and to dynamically select the tactic applied to address a particular EO.

NLG templates embedded within tutoring tactics (and also coaching advice) combine hard-coded text with XML elements; at runtime, repeated application of NLG substitution operators eventually reduces all elements to plain text. About forty distinct low-level NLG operators are supported, each of which accept an NLG template (and frequently, additional information), and return a suitably adjusted template for further processing. Typically, domain-dependent substitutions precede domain-independent substitutions in our pipeline.

Key operators permitting the repeated application of an NLG operator and the composition of multiple NLG operators allow us to rapidly and flexibly assemble compound operators, and refer to and invoke them in the aggregate. Such substitutions may be nested

arbitrarily deeply. Typical compound substitutions include: a) filling out appropriate references to actions, themes, topics, etc. with appropriate English phrases, punctuating lists, plus looking up and inserting the names of people referenced in the conversation; b) inflecting verbs and nouns, introducing pronouns, and regularizing spacing and capitalization.

2.5 *Process Control Daemons and Tracer*

The global machine daemon (GMD) manages system tasks (e.g., starting and stopping the RDBMS, loading system configuration data) and co-ordinates with the (one-per-machine) local machine daemons to start and stop components as required. In any system configuration, each participating component is assigned an even-numbered, though not usually unique, activation level (AL). Receipt of a message indicating that the system should transit to a particular, odd-numbered system AL causes the process control daemons to conduct an AL-ordered starting or stopping of sets of components that continues until the components executing are once again precisely just those components with ALs less than the system AL.

The tracer collects logging messages from each component and records them into a single, time-stamped file, simplifying post-session analysis of learner behaviour.

Acknowledgements

References

[1] Collins, A., Brown, J. S., and Newman, S. E. (1989). Cognitive apprenticeship: Teaching the crafts of reading, writing, and mathematics. In L. B. Resnick (Ed), *Knowing, learning, and instruction: Essays in honor of Robert Glaser* (pp. 453-493). Mahwah, New Jersey, U.S.A.: Lawrence Erlbaum Associates.

[2] Courtemanche, A. and Wittman, R. (2002). OneSAF: A Product-Line Approach for a Next-Generation CGF. *Proceedings of the Eleventh SIW Conference on Computer-Generated Forces and Behavioral Representations*, 349-361. Orlando, Florida, U.S.A.: SISO.

[3] Friedman-Hill, E. (2003). *Java Expert System Shell*. Greenwich, Connecticut, U.S.A.: Manning.

[4] Gomboc, D., Solomon, S., Core, M. G., Lane, H. C., and van Lent, M. (2005). Design Recommendations to Support Automated Explanation and Tutoring. *Proceedings of the Fourteenth Conference on Behavior Representation in Modeling and Simulation*. Universal City, California, U.S.A.: SISO.

[5] Hill, R. W., Belanich, J., Lane, H. C., Core, M. G., Dixon, M., Forbell, E., Kim, J., and Hart, J. (2006). Pedagogically structured game-based training: development of the ELECT BiLAT simulation. *Proceedings of the 25th Army Science Conference*. Orlando, Florida, U.S.A.: United States Office of the Under Secretary of Defense Acquisition, Technology and Logistics.

[6] Ilghami, O. (2006). Documentation for JSHOP2. *Technical Report CS-TR-4694*, Department of Computer Science, University of Maryland, College Park, Maryland, U.S.A.

[7] Katz, S., O'Donnell, G., and Kay, H. (2000). An Approach to Analyzing the Role and Structure of Reflective Dialogue. *International Journal of Artificial Intelligence in Education*, 11, 320-343.

[8] Katz, S., Allbritton, D., and Connelly, J. (2003). Going Beyond the Problem Given: How Human Tutors Use Post-Solution Discussions to Support Transfer. *International Journal of Artificial Intelligence in Education*, 13(1), 79-116. ISSN 1560-4292.

[9] Littman, M. L., Keim, G. A., and Shazeer, N. (2002). A probabilistic approach to solving crossword puzzles. *Artificial Intelligence: Chips challenging champions: games, computers, and Artificial Intelligence*, 134(1-2), 23-55. ISSN 0004-3702.

[10] Lane, H. C., Core, M. G., Gomboc, D., Karnavat, A., and Rosenberg, M. (2007). Intelligent Tutoring for Interpersonal and Intercultural Skills. *Proceedings of the Interservice/Industry Training, Simulation, and Educational Conference (I/ITSEC 2007)*. Orlando, Florida, U.S.A: NTSA.

[11] Lane, H. C., Hays, M., Core, M., Gomboc, D., Forbell, E., and Rosenberg, M. (2008). Coaching Intercultural Communication in a Serious Game. *Proceedings of the Sixteenth International Conference on Computers in Education (ICCE 2008)*. Taipei, Republic of China (Taiwan): APSCE.