

Toward Question Answering for Simulations

Mark G. Core, H. Chad Lane, Michael van Lent, Steve Solomon, Dave Gomboc, Paul Carpenter
The Institute for Creative Technologies, The University of Southern California
13274 Fiji Way, Marina del Rey, CA 90292 USA
core, lane, vanlent, solomon, gomboc, carpenter@ict.usc.edu

Abstract

The new research area of explainable artificial intelligence (XAI) allows users to question simulated entities whose motivations would otherwise be hidden. Here, we focus on the knowledge representation issues involved in building such systems.

1 Introduction

As artificial intelligence (AI) systems in military simulations become increasingly complex, it has been difficult for users to understand the activities of computer-controlled entities. Because military simulations are often used for their predictive power, the AI systems that drive them must be validated as behaving realistically and according to doctrine. Detailed specifications are drafted for these AI systems and the resulting behaviors are put under heavy scrutiny. In most cases, because the observer has no way to question AI-controlled entities, the observer's only recourse is watching numerous simulation runs looking for cases where faulty reasoning results in an incorrect action.

Military simulations are also used for training, replacing some or all of the soldiers in a live training exercise. Live training exercises often use after action reviews (AARs), typically led by a senior officer, to identify soldier and unit strengths and weaknesses. US Army Field Manual 25-101, "Battle Focused Training", states that "The OPFOR [opposing force] can provide valuable feedback on the training based on observations from their perspectives...the OPFOR can provide healthy insights on OPFOR doctrine and plans, the unit's action, OPFOR reactions to what the unit did." [Army, 1990][Appendix G] If the OPFOR are simulated entities, these entities must be able to answer questions to improve the student's understanding of their actions (human OPFORs in live exercises are available during AARs). The simulated friendly forces also need to participate in the AAR because otherwise human commanders may not see how their orders translate into the behavior of units and entities.

Figure 1 (a screenshot of our system's user interface) introduces the concept of an explanation system for simulated entities; following [van Lent *et al.*, 2004] we use the term, explainable artificial intelligence (XAI) system. Users select a time point to discuss, an entity to be questioned, and the

question itself. Some of the questions are specific to the particular entity (e.g., what is your health?) while others concern a larger group (e.g., what is your unit's task?).

In this paper, we first discuss the challenges involved in building such a system and then present our new modular architecture for this task and our implementation of this architecture for the military simulation, the One Semi-Automated Forces Objective System (OOS) [Courtemanche and Wittman, 2002]. We highlight the central database of the architecture as it allows us to link abstract plans to actions recorded in the log files, meaning simulated entities can explain how they are attempting to achieve their goals.

2 XAI Challenge

Explanation systems for simulated entities have been built previously [Johnson, 1994; van Lent *et al.*, 2004] but were specific to the AI systems controlling the simulated entities in those applications, and not directly applicable to other AI systems. If those systems do not represent the information necessary for explanation then the user will be limited in the questions he can ask. Consider the example of sending a fire team to clear a room. Once the fire team is in position outside the room, the grenadier throws a grenade before the team enters the room. This could be encoded as a procedure (the step before entering the room is always throwing the grenade) in which case, the system cannot explain why the grenade was thrown.

This problem is not new; the literature review in [Swartout and Moore, 1993] points out that researchers studying explanation for expert systems fairly quickly agreed that the data structures of the expert system had to be designed with explanation in mind. Swartout and Moore advocated building a high-level knowledge base containing facts about the domain and problem-solving strategies, and using an automatic program writer to build an expert system from this specification. The problem is more complicated for XAI because the executable code must interface with an external simulation (a technical challenge) and be adopted by users of the resulting package (a social challenge). Military users must be convinced that the resulting AI controlled entities are as realistic as the AI systems built by more traditional approaches. Another issue is the extra effort needed to create such a knowledge base, but it may be the case that such effort is offset by

Simulator: OneSAF Objective System Log File: xai_entity_beh_entity_data_1098653174938.xml

Morphy [2:16]: I am the fire team leader of Blue Fireteam 1. My unit's task is to clear the E room of the post office.

User: Can you give more detail?

Morphy [2:16]: My unit's task is to clear the E room of the post office, entering through the NE door. The unit is to stack up on the left side of the door. The weapon control status for the clear room task is free.

User: Let's talk to Private Bisquier.

Bisquier [2:16]: I am the grenadier of Blue Fireteam 1. I am currently transiting to the left stack position outside of the NE door of the E room of the post office.

User: Let's talk to Private Morphy.

Morphy [2:16]: I am currently transiting to the left stack position outside of the NE door of the E room of the post office.

1. Who are the other members of your unit?	• 2:16	1. <u>Morphy</u>
2. What are the unit roles of the members of your unit?	• 2:26	2. <u>Bisquier</u>
3. What weapons do you have?	• 2:27	3. <u>Browne</u>
4. What is your health/damage status?	• 2:28	4. <u>DeFirmian</u>
5. What is your location?	• 2:29	5. <u>Evans</u>
6. What is your unit's task?	• 2:30	6. <u>Byrne</u>
7. What is your current task?	• 2:31	7. <u>Marshall</u>
8. When did you start your current task?	• 2:32	8. <u>Lopez</u>
9. When will you complete your current task?	• 4:11	9. <u>Nakamura</u>
10. How do you execute your task?	• 4:20	10. <u>Denker</u>
11. What are your rules of engagement?	• 4:21	11. <u>Ashley</u>
12. What are the task assignments of the members of your unit?	• 4:22	12. <u>Torre</u>
	• 4:23	13. <u>Leko</u>
	• 4:24	14. <u>Ivanisevic</u>
	• 6:07	15. <u>Ljubojevic</u>
		16. <u>Romanishin</u>
		17. <u>Gligoric</u>

Figure 1: Interface to XAI for OOS

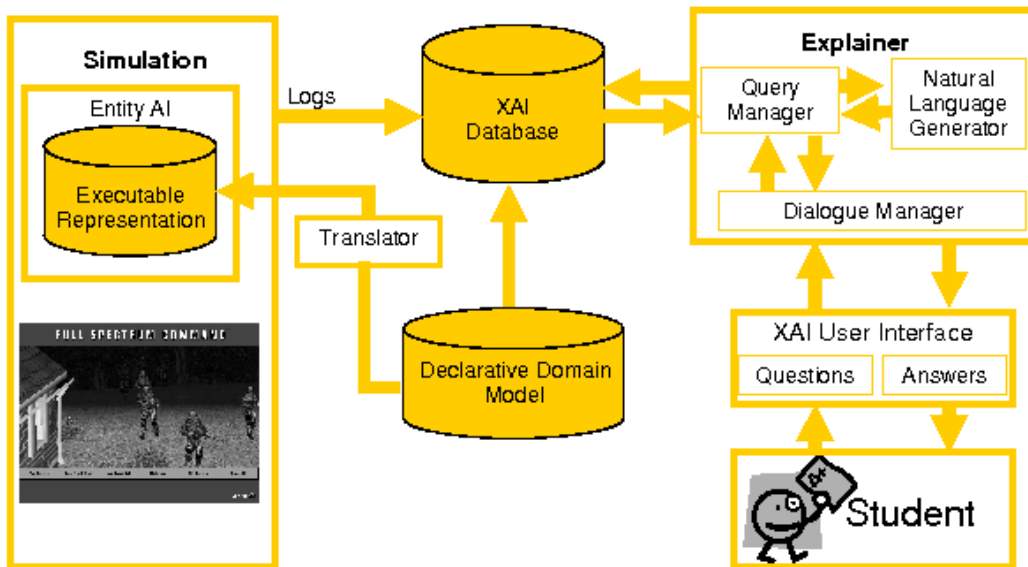


Figure 2: Domain Independent XAI Architecture

the increased usability of the system (e.g., the testing process may be quicker).

Swartout and Moore went on to discuss a plan-based approach to generating explanations, an architecture designed explicitly to facilitate:

- handling requests for clarification and elaboration
- customization based on a user model
- explanations at varying levels of detail
- natural explanations (e.g., human explanations often use pronouns and discourse markers)

We follow this advice in proposing a decoupled approach where the AI systems of a simulation are not changed but instead a declarative representation of the domain is translated into the knowledge representation used by the AI system; during explanation, the entity's actions can be reconnected to the original data structures. This domain representation would contain facts about the domain, definitions of terminology, and detailed representations of actions and problem solving strategies.

Consider the previous example of a procedure for clearing a room always having the "throw grenade" action before "enter room". The idea behind the procedure is that the grenade will suppress the enemy forces inside the room. System designers may not encode suppression as a precondition, but in the declarative domain model, we can define the concept suppression and include it as the goal of the throw grenade action in the context of clearing the room. The AI system will not use this information but it will be available during explanation to answer the question "Why was a grenade thrown into the room?"

Figure 2 shows how the declarative domain model augments the representations of an AI system. Each action type in the log file has a unique identifier linking it to its declarative representation and allowing the XAI database to reconstruct declarative representations of all the actions logged by the AI system. To answer user questions, the explainer selects information from the database and presents it via natural language and graphics displayed in the user interface.

3 XAI for OOS

Our first instantiation of the XAI architecture connects to the One Semi-Automated Forces Objective System (OOS), an entity based simulation system being developed by the U.S. Army [Courtemanche and Wittman, 2002]. XAI for OOS was a challenging project because of a short development time (2 months) and the fact that the target simulation was still a work-in-progress (the official release is in 2006). Interfacing early with a simulation or training aid is important because decisions by the developers of such programs can deeply impact the ability of an XAI system to explain entity actions and we plan to work with the OOS team to overcome the limitations discussed below.

The OOS team's approach to entity simulation is to encode declarative action representations in XML and use Java code to actually implement the described action. Our version of OOS only had the Java code (the XML descriptions were not

ready) and we had to enter action descriptions by hand into the XAI database (a relational database).

An additional source of information is the simulation scenario which contains the static information associated with the simulated world, and the mission table. The static information defines all the entities including their rank, unit, and weapons carried. The mission table contains a series of unit tasks and their parameters (e.g., unit task = clear room(agent,room), agent = fireteam 1, room = east room of the warehouse). During the simulation, entities perform individual tasks in service of these unit tasks, and log files record the states of these entities (e.g., when they start and end tasks) and information about any weapon fire events. After the simulation is complete, XAI loads the log files into its relational database and the query generator searches for "interesting" time points. The time menu on XAI's interface is sensitive to the current entity (i.e., it displays the interesting time points for that entity). For each entity, the query generator identifies interesting times, defined as when the entity fired its weapon and the start, end, and mid points of entity tasks. This definition of interesting times is a placeholder; when deploying an XAI system, feedback from subject matter experts is vital for defining what questions the system should answer and what events are interesting.

After this initialization process is complete, the user can investigate the results of the simulation using the interface shown in figure 1. The screenshot shows 12 of the 16 questions supported by the system and includes questions about the entities' state (e.g., "What is your health/damage status"), questions about the scenario (e.g., "Who are the other members of your unit"), and questions about tasks. The user can ask for descriptions of the current individual (primitive) task, the current group task, parameters of the task, or how, in general, to perform the task.

Although this list of questions reflects our research into questions asked during military after action reviews, they are limited to information directly encoded in the log files. In our current architecture, the query manager executes queries to retrieve answers to questions. In future work, we will replace this component with a general purpose XAI reasoner capable of answering the more complicated questions necessary for military training purposes. Another influence on what questions can be answered is the completeness of the simulator's log files. In [Gomboc *et al.*, 2005] we discuss requirements for such log files and how the requirements impact what questions can be answered.

The question list is context sensitive, and all 16 questions are not always available. If the query generator determines that there was no weapon fire event at the current time, then weapon fire questions (13-15) will be removed from the list. The question list is also sensitive to the dialogue history as the request, "Can you give me more detail?", can only be made when the previous turn described the current unit task, or described how to perform a unit task; currently our system is only able to elaborate on these two answers.

The interface to XAI for OOS is controlled by a servlet engine that communicates via XML with the explainer. Users select an entity and a time point from the menus on the bottom of the screen, and then select a question from the same

menu area. The middle frame of the page contains a dialogue history (all the questions asked by the user and the system responses).

Currently, the system starts by talking to the first entity of the first unit (in our test scenario, this is “Private Morphy”) as shown in the screenshot. 2:16 is the first significant time for Morphy as he starts his task of clearing the east room of the post office. The first time a user talks to an entity, the dialogue manager creates an introduction (such as the one shown in the first line of the dialogue). Since the user has never spoken to Morphy, he introduces himself by giving his unit role, and since the user has not spoken to any other members of Blue Fireteam 1, Morphy describes the unit’s task.

The process to create this introduction is the same process as answering the questions “what is your unit role?” and “what is your unit’s task”. The dialogue manager uses the query generator to retrieve the relevant information from the database and sends it to the natural language generation (NLG) module. Because of development time constraints, NLG was implemented with natural language templates (written in XSLT); slots in the templates are filled with information from the database. NLG formats names and times so that they appear as links in the user interface (clicking on them changes the current entity or time). We plan to use this facility in future work to encode formatting such as bulleted lists.

4 Discussion

Rather than simply writing an XAI module that only worked for its target AI system and simulator, we used our generic and modular architecture for XAI systems in building XAI for OOS. In a short time, with this new architecture, we have been able to exceed the capabilities of our predecessor system, XAI for Full Spectrum Command [van Lent *et al.*, 2004]. Because task information is linked to individual actions recorded in the log files, XAI for OOS can explain the generic method for achieving its current task as well as discussing the current parameters of this task.

The modularity of the system is important to allow interoperability with different AI systems and different simulators. As a portability test, we modified our system to accept log files from Full Spectrum Command (FSC). The OOS scenario that we have been using (light infantry) is very similar to the domain of FSC and we focused on supporting the questions that made sense in both domains. The major changes needed were changing the database schema to match the new format, updating the query manager so it could find the needed information in this new format, and changes to NLG to support new actions and objects.

The modularity of the system will make it easier to improve. For example, we plan to extend the dialogue manager to maintain a more extensive dialogue history. This code can be rewritten or replaced by a dialogue management toolkit without disturbing the other parts of the system. Even the user interface is simply a module of the system, and another interface supporting the explainer’s XML communication format could replace our current interface with no changes to the rest of the system.

The modularity of our XAI architecture will also enable it to integrate with external software. Currently, with respect to training, XAI is best classified as a discovery system where users investigate the events of the simulation with no external direction (i.e., learning is left entirely up to the student). We are currently designing an intelligent tutoring module to provide the pedagogical presence necessary for effective training. The tutor could use the XAI system to illustrate an important lesson, or direct the student to use XAI to investigate an interesting time point, among other strategies.

A topic for future work is defining the general relationship between question answering systems and tutoring. One possibility is to teach the students about the QA system itself (how to use it). We are developing an investigation model that encodes the detective skills that an expert XAI user would possess. This model will be similar to troubleshooting guides for electronics (e.g., try to isolate the component causing the problem, then investigate the component in detail searching for faults). We can imagine developing investigation models for other QA tasks such as finding relevant research papers, or bargain shopping for airfares.

Acknowledgments

Thanks to Milton Rosenberg, William Swartout, and David Traum for their guidance and support. Note, the project described here has been sponsored by the U.S. Army Research, Development, and Engineering Command (RDECOM). Statements and opinions expressed do not necessarily reflect the position or the policy of the U.S. Government, and no official endorsement should be inferred.

References

- [Army, 1990] FM 25-101: Battle Focused Training. Headquarters Department of the Army. Washington D.C., 1990.
- [Courtemanche and Wittman, 2002] A. Courtemanche and R. Wittman. OneSAF: A product-line approach for a next-generation CGF. In *Proc. of the Eleventh SIW Conference on Computer-Generated Forces and Behavioral Representations*, pages 349–361, 2002.
- [Gomboc *et al.*, 2005] D. Gomboc, S. Solomon, M. G. Core, H. C. Lane, and M. van Lent. Augmenting behavior models to support automated explanation and tutoring. In *Proc. of the Fourteenth Conference on Behavior Representation in Modeling and Simulation*, 2005.
- [Johnson, 1994] W. L. Johnson. Agents that explain their own actions. In *Proc. of the Fourth Conference on Computer Generated Forces and Behavioral Representation, Orlando, FL*, 1994.
- [Swartout and Moore, 1993] W. R. Swartout and J. D. Moore. Explanation in second generation expert systems. In J.M. David, J. P. Krivine, and R. Simmons, editors, *Second Generation Expert Systems*. Springer-Verlag, 1993.
- [van Lent *et al.*, 2004] M. van Lent, W. Fisher, and M. Mancuso. An explainable artificial intelligence system for small-unit tactical behavior. In *Proc. of the Sixteenth Conference on Innovative Applications of Artificial Intelligence Conference*, Menlo Park, CA, 2004. AAAI Press.