Verifying and Explaining Agent Behavior in an Implemented Agent System

Dung N. Lam and K. Suzanne Barber
The Laboratory for Intelligent Processes and Systems
Electrical and Computer Engineering, The University of Texas at Austin
Austin, TX 78712
{dnlam,barber}@lips.utexas.edu

Abstract

As agent systems become more sophisticated, there is a growing need for agent-oriented debugging, maintenance, and testing methods and tools. This paper presents the Tracing Method and accompanying Tracer tool to help verify actual agent behavior in the implemented system against expected (or designed) agent behavior. The Tracing Method captures dynamic run-time data as actual agent behavior, creates modeled interpretations in terms of agent concepts (e.g. beliefs, goals, and intentions), and compares those models to the agent behavior expected by the designer or developer; thereby, gaining insight into both the design and the implemented agent behavior. The Tracer tool can help: (1) determine if agent design specifications are correctly implemented and guide debugging efforts and (2) discover and examine motivations for agent behaviors such as beliefs, communications, and intentions.

1. Introduction

There are several agent-oriented software design methodologies and development environments, but there are few agent-oriented methods and tools that have been created for debugging, maintaining, or testing the resulting implemented system. The objective of the Tracing Method and Tracer tool is to ensure that an agent is performing actions for the right reasons and, if an unexpected action occurred, to help explain why an agent decided to perform the action. Due to the increasing sophistication of agent software and the number of factors to consider when understanding or explaining agent behavior, comparing the implementation’s actual behavior with expected behavior can be an intensive task, requiring time and knowledge about the design, implementation, and application domain. Performing the task of verifying agent behavior in the implementation can identify undesired agent behaviors (bugs) and help to comprehend how agents operate and how those agents can be improved (for maintenance and testing).

A motivation for applying agent-oriented techniques is to make the problem and solution easier to understand (i.e., by localizing beliefs and goals into autonomous agents), but after the agent system has been implemented as source code, distinguishing agent concepts in the implementation can become difficult as the complexity of the implementation increases. This research aims to extract agent concepts from the implementation and to regain the advantages of conceptualizing the implementation in terms of agent concepts. Figure 1 illustrates the relationships among the agent concepts, which initially include agent, goal, belief, intention, action, event, environment and message.

Figure 1: Agent concept relationships
2. The Tracing Method

Due to the inherent unpredictability of autonomous agents in an uncertain environment and the possibility of emergent behavior, Jennings stresses a need for a better understanding of how agent interaction affects an individual agent’s behavior [1]. The idea of the Tracing Method is to capture uncertain, dynamic run-time data (e.g., environmental events and communicated beliefs), to observe each agent’s behavioral response, and to help explain this behavior. Using the Tracing Method shown in Figure 2, interpretations (models or diagrams that represent the actual agent behavior in terms of agent concepts) are created using observations resulting from the implementation’s execution. These interpretations can be the same models and diagrams that result from reverse engineering (e.g., flow control, component dependence, and class inheritance models or state-chart and process-flow diagrams), or the interpretations can be lists of observations about what agents are doing in terms of agent concepts.

The Tracing Method involves logging agent behavior during execution, abstracting and mapping the log entries and run-time data into interpretations, and comparing those interpretations with the models of expected agent behavior. As a result, the agent concepts (e.g., beliefs about the current state of the environment) are instantiated with actual run-time data. By comparing the interpretations with the models of expected behavior, actual behavior can be verified against expected behavior.

If the interpretations are inconsistent with models of expected behavior, the implementation or the expected behavior may need to be changed. Details about the inconsistencies are presented to the user so that the implementation or expected behavior can be corrected. Since each observation can be traced back to a location in the source code, correcting the inconsistency is facilitated.

The Tracing Method can be used repeatedly throughout the software life-cycle from the first skeleton code to the final system. The Tracer tool aims to automate the developer’s task of analyzing run-time data, creating interpretations of actual agent behavior, and relating those interpretations to models of expected agent behavior. Essentially, the Tracer tool translates the procedural execution of the implementation into declarative statements about what and when the agent believes, intends, and performs. The results of the Tracer tool are behavioral and structural models, representing interpretations of the logged data based on order, duration, and other run-time attributes. Since the interpretations are similar to design models, they can be compared to the original design models to ensure important areas of the agent design have been traced.

3. Summary

The Tracing Method and Tracer tool offer the ability to (1) determine if agent design specifications are correctly implemented and guide debugging efforts and (2) examine and discover motivations, such as beliefs, intentions, and communicated messages, for agent behaviors. The Tracer tool has been applied to a target-monitoring UAV domain and was helpful in gaining insight into agent behavior by automating the process of generating interpretations of the implementation execution and presenting observed agent behavior in terms of agent concepts (e.g., beliefs, goals, intentions, actions, messages, etc.). Using the Tracer tool, detecting, locating, and correcting agent behavior is facilitated because each element of the interpretations can be traced back to a precise location in the source code.

This research proposes a method and tool to create models of agent behavior that not only describe what is occurring in the implementation but why a respective agent behavior occurred (e.g., agent X took action a because of belief b). To enable such explanations, the Tracing Method requires only a high-level understanding of where agent concepts are modified in the implementation. In this regard, the behavior of agents in unfamiliar agent systems can be quickly understood.

4. References