Science of Computer Programming 167 (2018) 70–90

Contents lists available at ScienceDirect

Science of Computer Programming

www.elsevier.com/locate/scico

AdSiF: Agent driven simulation framework paradigm and ontological view

Mehmet F. Hocaoğlu

Istanbul Medeniyet University, Faculty of Engineering & Natural Sciences, Göztepe/Istanbul, Turkey

ARTICLE INFO

Article history:
Received 22 March 2017
Received in revised form 27 June 2018
Accepted 9 July 2018
Available online xxxx

Keywords:
Agent-driven simulation
Agent Programming
Logic programming
Ontology
State-oriented programming

ABSTRACT

AdSiF (Agent driven Simulation Framework) provides a programming environment for modeling, simulation, and programming agents, which fuses agent-based, object-oriented, aspect-oriented, and logic programming into a single paradigm. The power of this paradigm stems from its ontological background and the paradigms it embraces and integrates into a single paradigm called state-oriented programming. AdSiF commits to describe what exists and to model the agent reasoning abilities, which thereby drives model behaviors. Basically, AdSiF provides a knowledgebase and a depth first search mechanism for reasoning. It is possible to model different search mechanism for reasoning but depth first search is a default search mechanism for first order reasoning. The knowledge base consists of facts and predicates. The reasoning mechanism is combined with a dual-world representation, it is defined as an inner representation of a simulated environment, and it is constructed from time-stamped sensory data (or beliefs) obtained from that environment even when these data consist of errors. This mechanism allows the models to make decisions using the historical data of the models and its own states.

The study provides a novel view to simulation and agent-modeling using a script-based graph programming structuring state-oriented programming with a multi-paradigm approach. The study also enhances simulation modeling and agent programming using logic programming and aspect orientation. It provides a solution framework for continuous and discrete event simulation and allows modelers to use their own simulation time management, event handling, distributed, and real time simulation algorithms.

1. Introduction

Programming languages have their own world view and they define an existence (the thing) based on ontological commitments. For example, the object-oriented paradigm is to envision the world as several entities that interact with one another and to implement definite functions. Its ontological commitment is based on Kant's noumena and it is simply claimed that objects exist independently of human perception and are not ontologically exhausted by their relations with humans or other objects [1]. This allows modelers to describe a tangible world as if it were real. In contrast, the world view in the logic paradigm is a syntactic and semantic structure built from propositions and predicates, where the goal is to answer the queries of a user appropriately [2]. Logic programming is accommodated in AdSiF to drive behaviors depending on a reasoning result that is achieved by a Horn clause subset of the first-order predicate calculus. AdSiF extends object definition at three levels. At the first level, it wraps an object's functions by states and collects states in semantically

E-mail address: mfatih.hocaoglu@medeniyet.edu.tr.

https://doi.org/10.1016/j.scico.2018.07.004
0167-6423/

meaningful behavior descriptions. At the second level, it enhances the relation concept. A relation may exist between two entities persistently or contemporarily and it may force a set of specific actions both the existence and breaking phases of the relation. The third level is about consciousness. An object is enhanced with a reasoning capability, which allows it to manage its behaviors and to understand events and states around it, with a memory keeping earlier states of entities in the world and states it has.

The agent-oriented-programming (AgOP) paradigm makes application and simulation models more social, flexible, and interoperable because it focuses on responses, is capable of behavior planning, and has anthropomorphic and autonomous characteristics. The common feature of agents is their autonomous acting on behalf of other agents and/or modelers. The agent metaphor was used to describe certain types of software programs at least 30 years ago in (distributed) artificial intelligence [3,4]. Agents are typical inhabitants of artificial open systems like the Internet. The open systems were characterized by Hewitt [5] in the 1980s as systems with continuous availability, extensibility, modularity, arm-length relationships, concurrency, asynchronous work, decentralized control, and inconsistent information. This characterization shows the fact that an agent is a continuously running program, where the work can be meaningfully described as an autonomous completion of orders or goals while the program interacts with the environment [6].

Aspect-oriented programming (AOP) is a way to create common or similar functionality needed by different parts of a program. Programmers describe the necessary behaviors in modules called aspects and rely on a specialized AOP mechanism to weave or to compose them into a coherent program [7]. The scattered and tangled requirements and codes that arise in object-oriented programming can be handled with the vertical and hierarchical design provided by AOP [6]. AdSiF provides a means of handling aspects by categorizing behaviors regarding their semantics and activating and deactivating them conditionally in run time.

AdSiF is a general-purpose, multi-paradigm declarative scripting language. The main strengths of the language, which are reusability, interoperability, expressiveness, orthogonality, and some other software metrics examined in detail in Section 6, are related to agents and simulation models developed in AdSiF. The ontologies supporting the multi-paradigm consist of object-oriented paradigm, logic programming paradigm, agent-oriented paradigm [8,9] and aspect oriented programming. In the literature, we see agent languages such as PROforma [10,11] that combine logic programming, object-oriented programming and agent-oriented programming. AdSiF is distinguishably different from similar languages because, in AdSiF, all of these paradigms are structurally fused together in the State-Oriented Paradigm (SOP), and the richness of the paradigms enhances the world view that is provided to the modelers. The reasons for incorporating each programming paradigm into the state-oriented programming paradigm are explained in related sections. Broadly speaking, the logic programming brings in a reasoning capability with a knowledge-base that keeps the older facts (facts with earlier time labels) about the world and it provides inference-based behavior management such as giving up doing a behavior being done, starting up a behavior, or re-doing a behavior. Aspect orientation is accommodated basically for two reasons. The first reason is to manage scattered requirements that are satisfied by a specific set of behaviors and tangled requirements that are satisfied by a group of behaviors. The second reason is more meaningful for an agent and simulation development environment and it is to be capable of dynamically changing a course of action. In other words, it makes it possible to change active behavior sets that each represent an aspect of the agent, using activation and deactivation conditions attached to the behavior containers. This allows modelers to develop agents, that are capable of behaving in different way depending on the conditions they are in. The agent based programming approach enables an architectural view to be gained in simulation modeling.

The paper is organized as follows. Section 2 is related to agent-based simulation concepts. Section 3 examines the ontological views and state-oriented programming concepts. Section 4 presents how time and cosmology concepts are taken into consideration in AdSiF. In Section 5, AdSiF is evaluated according to several software engineering criteria with pros and cons. In Section 6 simulation examples are presented to clarify the concepts. How simulation algorithms are developed as behaviors and their functions as plugins are explained and a time management approach is given, briefly, in Section 7. Finally, the paper ends with a discussion section.

2. Agent based simulation

An agent-based model represents a system as a collection of autonomous decision-making entities called agents, each of which individually evaluates its situation and makes decisions based on a set of rules [12]. Agents in a simulation are accommodated or a simulation model is designed as agents to provide a strong information exchange capability and robust decision-making. Interoperability standards between agents and specifications for agent-based systems are promoted [13].

The abilities that agents have such as personality, emotions, cultural backgrounds, irrationality, and dysrationalia, which affect their decision-making capabilities, and the abilities making them more intelligent such as anticipation (proactiveness), understanding (avoiding misunderstanding), learning, and communication in natural and body language, and the abilities making them trustworthy such as being rational, responsible, and accountable [14], open a new area in simulation with agents. In this respect, the main reason to accommodate an agent in a simulation environment or for modeling simulation entities as agents is the fact that the core knowledge-processing abilities of agents include: goal-processing, goal-directed knowledge-processing, reasoning, motivation, planning, and decision-making. The synergy of simulation and software agents is the essence of agent-directed simulation (ADS) and it is shown to have important practical implications. ADS as clarified in the following, consists of contributions of simulation to agents (i.e., agent simulation) and contributions of agents to simulation (i.e., agent-supported simulation and agent-based simulation) [15]. The goal-directed characteristics