Chapter XXXV
Agent-Oriented Modeling and Simulation of Distributed Manufacturing

Kuldar Taveter
University of Melbourne, Australia

Gerd Wagner
Bradenberg University of Technology at Cottbus, Germany

ABSTRACT

This chapter proposes an agent-oriented method for modeling and simulation of distributed production environments. The proposed method views a manufacturing enterprise as consisting of active entities—agents. The method makes use of the Radical Agent-Oriented Process (RAP) methodology introduced by Taveter and Wagner (2005) which is based on Agent-Object-Relationship (AOR) modeling. The chapter first presents the AOR Modeling Language and the RAP/AOR viewpoint modeling framework of the methodology. Thereafter its lays out principles of turning the modeling constructs of the RAP/AOR methodology into the implementation constructs of a simulation environment and briefly describes the simulation environment. The method is aimed at the creation of environments for modeling and simulation of distributed manufacturing.

INTRODUCTION

With the advent of virtual enterprises, new business models emerge. In them, a manufacturing enterprise should be capable of composing its manufacturing processes in a modular fashion so that if the factory receives an order at short notice, the satisfaction of which requires only a part of a full-length manufacturing process of the enterprise, the order will be scheduled and satisfied in a dynamic, flexible, and fast manner. One way to achieve this is to view a manufacturing enterprise as a collection of active entities—*agents*—so that each resource would be represented by an agent responsible for scheduling and performing its manufacturing operations. An agent is autonomous and does not know the decision logic of the other agents as a rule. The decision logic is thus specified for each agent individually and not for the system of agents as a whole. Differently from conventional modeling approaches, including UML (OMG, 2003a, 2003b), this is closer to how real socio-technical systems—consisting of both human and technical components—operate. This is why our approach can be characterized as inspired by nature.

THE RAP/AOR METHODOLOGY

The Radical Agent-Oriented Process/Agent-Object-Relationship (RAP/AOR) methodology of simulation and software engineering, which was introduced by Taveter and Wagner (2005), is based on the Agent-Object-Relationship Modeling Language (AORML) proposed by Wagner (2003a) and the Business Agents' approach presented in Taveter (2004). The ontological foundation of the RAP/AOR concepts is provided by the Unified Foundational Ontology (UFO) proposed by Guizzardi and Wagner (2005). The UFO defines an ontological distinction between active and passive entities—that is, between agents and (non-agentive) objects of the real world. The agent metaphor subsumes *artificial* (software and robotic), *natural* (human and animal), as well as *social/institutional* agents (groups and organizations). We will subsequently describe AORML, which is used as the main graphical description for work products of RAP/AOR. Thereafter, we will introduce the RAP/AOR viewpoint modeling framework forming the core of the methodology.

The AOR Modeling Language

In AORML, an entity is an agent, an event, an action, a claim, a commitment, or an ordinary object. Only agents can communicate, perceive, act, make commitments, and satisfy claims. Objects are passive entities with no such capabilities. Besides human and artificial agents, AORML also includes the concept of *institutional agents*, which are composed of a number of other agents that act on their behalf. Organizations and organizational units are important examples of institutional agents.

There are two basic types of AOR models: *external* and *internal* models. An external AOR model adopts the perspective of an external observer who is looking at the (prototypical) agents and their interactions in the problem domain under consideration. In an internal AOR model, the internal (first-person) view of a particular agent to be modeled is adopted. While a (manufacturing) domain model corresponds to an external model, a design model (for a specific agent-oriented information system) corresponds to an internal model which can be derived from the external one. Since the use of external AOR models suffices for the purposes of simulation, in this chapter internal AOR models are treated only marginally.

An *external AOR diagram* specified by Figure 1 shows how the types and instances (if applicable) of institutional, human, and artificial (for example, software) agents of a problem domain can be represented, together with their internal agent types and instances and their beliefs about instances of "private" and external ("shared") object types. There may be attributes and/or predicates defined for an object type and relationships (associations) among agent and/or object types. A predicate, which is visualized as depicted in Figure 1, may take parameters. As in UML (OMG, 2003a, 2003b), an instance of a type is graphically rendered by a respective rectangle with the underlined name of the particular instance as its title.

As formulated in Wagner (2003a) and reflected by Figure 1, if an object type belongs exclusively to one agent or agent type, the corresponding rectangle is drawn inside of this agent or agent type rectangle. Otherwise, if the

Related Content