2000

A Knowledge-Based Agent Modeling and Design Environment

Vijayan Sugumaran
Oakland University, sugumara@oakland.edu

Sooyong Park
Sogang University, sypark@ccs.sogang.ac.kr

A Knowledge-Based Agent Modeling and Design Environment

Vijayan Sugumaran, Dept. of DIS, Oakland University, Rochester, MI 48309, sugumara@oakland.edu
Sooyong Park, Dept. of Computer Science, Sogang University, Seoul, Korea, sypark@ccs.sogang.ac.kr

Abstract

Agent-oriented software systems are becoming large and complex. This paper presents a methodology for agent-oriented software development, grounded in software engineering principles. It also presents a knowledge-based agent modeling and design environment that supports different phases of the agent-software lifecycle.

Introduction

As software applications get large and complex, very sophisticated environments are needed to support and execute heterogeneous and distributed real-time applications. To manage this complexity, intelligent agent technology is beginning to be employed as part of the solution in various software environments (Maes, et al., 1998). Since its introduction in the AI community, agent technology has permeated to various application domains as simple as e-mail filtering, to as complex as Air-traffic Control (Jennings et al., 1998). Recently, in distributed and heterogeneous environments such as Electronic commerce (EC) applications, intelligent agents are increasingly being utilized to perform various tasks.

Since agents are used in many application areas, a systematic approach that is grounded within the software engineering paradigm is highly important for the development of agent-oriented software. However, there has not been enough research on this subject in the Software Engineering Community (Woolridge et al., 1999).

This paper presents a systematic approach for developing agent oriented software and an agent development environment. This development environment, uses a knowledge-based approach for generating new software agents from a family of agent oriented system.

Agent-Based Software Development Model

To develop agent-oriented software, a full life cycle model is needed. Our suggested model, which is depicted in Figure 1, is an adaptation of the traditional Waterfall model and contains the following major activities:

a) domain analysis - problem domain understanding and modeling, and agent identification,
b) agent modeling - intra-agent and inter-agent modeling,
c) agent design - agent architecture & componentization,
d) agent implementation – implementing the agents using agent building tools and communication languages,
e) agent integration – integration of multi-agents and other components, and
f) verification and validation – testing and simulation of the agent functionalities.

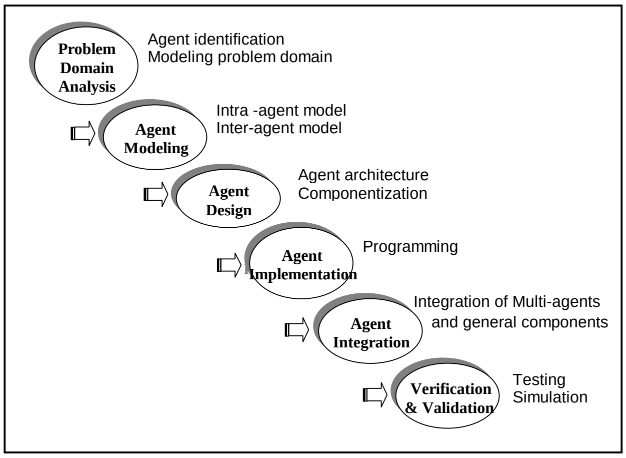

Figure 1. Agent-Oriented Software Development Process Model

In order to effectively carryout the above mentioned activities, there is a great need for a software engineering environment that would facilitate agent modeling, design, development, and testing. In particular, this environment should support the user in problem domain analysis and domain model creation, from which potential agents could be identified. It should provide mechanisms for generating customized agents from generic agent templates based on the user requirements. During the customization process, the system should perform consistency checking to ensure that the resulting agents are consistent with each other in terms of functionalities and interface.

Agent Modeling

Before agents could be implemented, one has to model these agents to clearly articulate their functionalities and how they interact with each other in cooperative problem solving. To accomplish this, we propose two types of models: a) intra agent model, and b) inter agent model. The intra agent model represents the characteristics and behaviors internal to the agent, while the intra agent model represents how agents communicate and interact with each other.

The Intra agent modeling approach proposed in our methodology is based on the BDI model (Rao, 1991), and

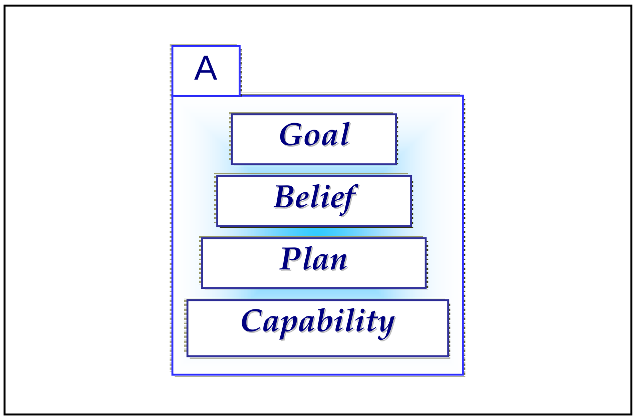

Figure 2. Abstract view of Intra agent model

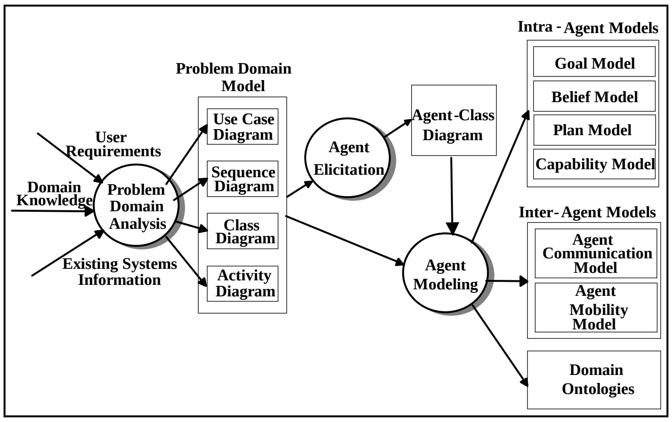

Figure 3. Problem Domain Analysis & Agent Modeling

Reticular Agent Mental Models proposed by Thomas (1993). We represent the internals of an agent using the following four models: a) goal, b) belief, c) plan, and d) capability. Figure 2 shows an abstract view of the Intra agent model. Goal is an ultimate objective that an agent has to achieve, and the goal model consists of a hierarchy of sub-goals. An agent may delegate tasks to other agents in order to achieve a certain sub-goal. Goals are categorized into five groups: a) achieve, b) cease, c) maintain, d) avoid, and e) optimize. The belief model captures information about the environment in which the agent resides in, and the agent itself. This forms the agent's knowledge-base or ontology. The plan model shows the behavior of an agent to achieve its goals. It is similar to a sequence diagram, which shows the series of steps that have to be taken in order to achieve a certain goal. These steps may involve updating values, eliciting services from other agents and objects, sending messages, and migrate to another domain (agent mobility). The capability model shows the operations that an agent can perform. It shows the internal processing of an agent, i.e., taking a set of inputs and generating a set of outputs.

Agent's mobility and the types of messages exchanged between agents in multi agent systems are modeled in the Inter agent modeling process. This results in two types of models: a) agent mobility model and b) agent communication model. The agent mobility model shows how an agent migrates from one domain to another to perform a specific task. It also shows the mechanisms that determine the destination (host) that the mobile agent migrates to. The agent communication model shows the exchanging of messages among agents. It shows the types of messages that agents use to communicate and their corresponding arguments.

Knowledge-Based Agent Generation

Based on our agent oriented software development lifecycle model, we are developing a Knowledge-Based Agent Generation Environment, which supports the different phases within the lifecycle. In particular, this environment supports the following three major activities: a) problem domain analysis and agent modeling, b) agent design and generation, and c) agent testing and application generation. One of the objectives of this environment is to promote agent reuse. Using this agent development environment, users can design and implement a family of agent-based systems in a particular application domain. The generic architectures of agents and systems thus created can be stored in a reuse repository, which can then be used as a starting point for designing a new system in the same application domain. The following paragraphs describe the process of designing and implementing an agent-oriented software application, as we envision it.

Domain Analysis and Agent Modeling

The initial phase supported by the environment in developing agent-oriented software is the domain analysis and agent modeling. This phase consists of the following steps: a) Problem Domain Analysis, b) Agent Elicitation, c) Intra Agent Modeling, and d) Inter Agent Modeling (as shown in Figure 3). A UML based Object- Oriented Analysis Method is used for the problem domain analysis. Since the utility of UML has been extensively validated in different industries, the UML based approach will provide an objective view of the domain in the early stages of Agent Elicitation (Seleic, et al., 1998; Herman, 1998). It also facilitates deep understanding of the problem domain with static and dynamic aspects of the system.

After analyzing the problem domain, a domain model is created which consists of a) use case diagram, b) sequence diagram, c) class diagram, and d) activity diagram. In the agent elicitation process, potential objects that can be "agentified," are derived from the domain model using agent selection rules. Also, additional agents that need to be added are identified. Then, these agents and objects are assimilated and represented in an Agent-Class Diagram. This diagram depicts the relationships between the various objects and agents. Once agents are identified, their internal characteristics are captured in the intra-agent modeling

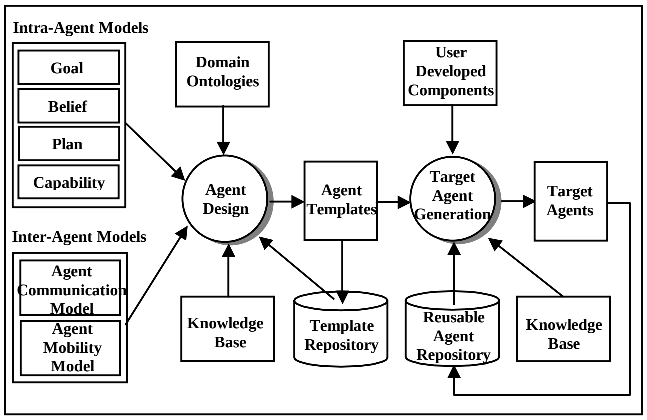

Figure 4. Agent Design and Agent Generation

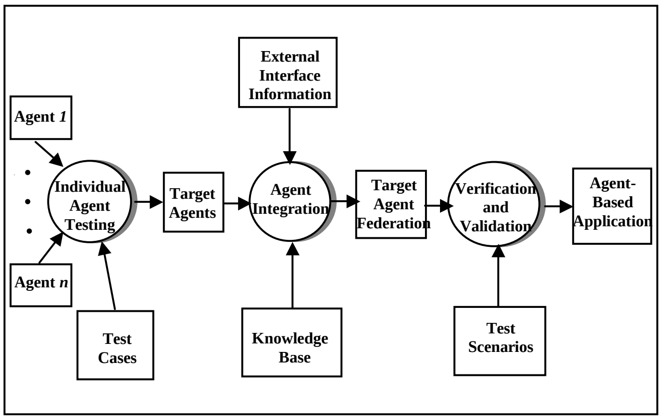

Figure 5. Agent Testing & Application Generation

step and their external behavior is represented in the inter-agent modeling step. The completion of the agent modeling process results in a goal model, belief model, plan model, capability model, agent communication model, agent mobility model, and domain ontologies, as shown in Figure 3.

Agent Design and Target Agent Generation

Once the agent modeling process is completed, the agent design process follows. The agent models created in the previous phase serve as the primary input for this process. Agent design templates are created based on these agent models, as well as the domain ontology and knowledge-base. Figure 4 shows the agent design and target agent generation processes. In the generation of an agent template, various optional design templates from the template repository are considered based on domain ontology and knowledge base that contain design options and feature relations. The new templates that are generated are also stored in the template repository for later reuse. The agent templates then serve as the input to the target agent generation process in which the designer adds the implementation details to the agent. In generating the target agents, the designer can make use of existing agent components with optional features, stored in the reusable agent repository, or integrate custom developed components from other sources. The knowledge-base provides support for consistency checking. The target agents are also stored in the reuse repository for later reuse.

Agent Testing and Application Generation

The final phase is the agent testing and the application generation phase, as shown in Figure 5. The target agents generated from the previous phase are first tested individually using a variety of test cases. Then, these agents are integrated to create a federation of agents and this agent federation is also tested. The agent federation along with domain specific classes contained in the agent-class diagram form the agent-based software application. This application is subjected to verification and validation using several test scenarios.

Summary

In this paper, we have presented an agent modeling method for real world applications, and an overall agent-oriented software development process model. We have also described a knowledge-based agent development environment that supports various phases of this lifecycle. This environment facilitates the analysis and modeling of the problem domain, which helps in crystallizing the requirements for the agent-based system. The domain model acts as the backdrop for the agent modeling, design, and implementation activities.

References

Harmon, P. and Watson, M. *Understanding UML: The Developers Guide*, Morgan Kaufman Publishers, 1998.

Jennings, N.R., Sycara, K. P. and Wooldridge, M. "A Roadmap of Agent Research and Development," In *Journal of Autonomous Agents and Multi-Agent Systems*. (1:1), July 1998, pp. 7-36.

Maes, P., Guttman, R. and Moukas, A. "Agents that Buy and Sell: Transforming Commerce as we Know It," Communications of the ACM, March 1999

Rao, A. S and Georgeff, M. P. "Modeling rational agents within a BDI-architecture," *Proc. of Knowledge representation and reasoning*, Morgan Kaufmann Publishers, 1991, pp. 473-484.

Selic, B. and Rumbaugh, J. *Using UML for Modeling Complex Real-Time Systems*, 1998.

Thomas,S.R., PLACA, An Agent Oriented Programming Language, PhD Thesis, Stanford University, 1993

Woolridge, M., Jennings, N., Kinny, D. "A Methodology for Agent-Oriented Analysis and Design," *Proc. Of Autonomous Agents '99*, Seattle, WA, 1999, pp. 69-76.