Available online at www.sciencedirect.com

ScienceDirect

The Journal of Systems and Software 79 (2006) 1565–1577

The Journal of Systems and Software

www.elsevier.com/locate/jss

A combined specification language and development framework for agent-based application engineering

José Alberto R.P. Sardinha [a,*], Ricardo Choren [b], Viviane Torres da Silva [a], Ruy Milidiú [a], Carlos J.P. de Lucena [a]

[a] *Pontifical Catholic University of Rio de Janeiro, Computer Science Department, Rua M. de S. Vicente 225, Gávea, 22451-900 Rio de Janeiro, RJ, Brazil*

[b] *Military Institute of Engineering, Department of Systems Engineering, Pça Gen. Tibúrcio 80, Praia Vermelha, 22290-270 Rio de Janeiro, RJ, Brazil*

Received 8 March 2005; received in revised form 1 February 2006; accepted 11 February 2006
Available online 23 March 2006

Abstract

Software agents are used to solve several complex problems. Software engineering processes and tools are vital to support all the development phases of agent-based systems, and guarantee that these systems are built properly. Indeed, many modeling languages and implementation frameworks are available for system engineers. However, there are few methods that properly combine agent-oriented design and implementation initiatives. In this paper, we propose a development method that goes from the requirements elicitation to the actual implementation of agent systems using a modeling language and an implementation framework. We also present a case study to illustrate the suitability of our approach.

Keywords: Software agents; Agent oriented software engineering; Framework

1. Introduction

The agent technology is used to develop systems that perform several complex tasks. A software agent embodies some goals, some actions that are executed to achieve these goals, some high-level message interface, and a set of agency properties, such as autonomy, adaptation, interaction and learning. The system's overall goal is achieved through the synergetic cooperation of the agents.

An agent-oriented software engineering is currently under development and its main goal is to determine how agent quality affects software engineering and what additional tools and concepts are needed to apply software engineering processes and structures to agent systems (Wooldridge and Ciancarini, 2001). In this sense, the engineering of agent-oriented applications require particular tools and techniques for developers to improve the design and the implementation of agent systems. For instance, methodologies are needed to guide the analysis and design phases, agent architectures are required for the design of individual components, and integrated supporting infrastructures are essential to obtain productivity (Luck et al., 2004).

At present, there are a number of agent-oriented modeling languages and methodologies, such as Castro et al. (2002), DeLoach (1999), Padgham and Winikoff (2002b) and Zambonelli et al. (2003), and there are also some platforms or

[*] Corresponding author. Tel.: +55 21 2540 6915x133; fax: +55 21 2511 5645.
E-mail addresses: sardinha@inf.puc-rio.br (J.A.R.P. Sardinha), choren@de9.ime.eb.br (R. Choren), viviane@inf.puc-rio.br (V.T. da Silva), milidiu@inf.puc-rio.br (R. Milidiú), lucena@inf.puc-rio.br (C.J.P. de Lucena).

0164-1212/$ - see front matter © 2006 Elsevier Inc. All rights reserved.
doi:10.1016/j.jss.2006.02.018

architectures for agent-based system implementation, such as Howden et al. (2001) and Bellifemine et al. (2005). However, there very few methods that attempt to properly combine agent design and implementation initiatives. Such methods are important not only to allow the fast and easy development of agent systems but also to foster the interest of commercial organizations in agent systems. Consequently, this can provide further dissemination of agent technologies (Luck et al., 2004).

In this paper, we propose a development method for building multi-agent systems that goes from the requirements elicitation to the actual implementation of the agent system. This method combines some software engineering techniques in a straightforward manner, and consequently presents some progress to the agent-oriented field. Three phases are proposed in this method: (i) an agent design phase, (ii) a migration or mapping path from the design results to an agent framework, and (iii) the actual implementation of the agent system.

The design phase uses an agent-oriented modeling language called ANote (Choren and Lucena, 2004, 2005). ANote is a modeling language that adopts the metaphor of agents and design views to provide natural and refined means to specify agent systems. It has a conceptual meta-model and its diagrams are able to model goals, actions, interactions and some agency characteristics.

The generated artifacts in the design phase are then mapped to a framework called Agent's SYNergistic Cooperation (ASYNC) framework (Sardinha et al., 2003b). The ASYNC object-oriented framework is used to implement the agents and the system environment. The framework also provides a communication infrastructure and uses some hot spots in order to implement the agent's goals, actions and interaction protocols. Finally, the agent system is implemented by instantiating the ASYNC framework and properly implementing the defined hot spots.

The proposed method is a result of the development of early case studies from which we could evaluate some important factors for the deployment of agent systems and also recommend a mapping from design to implementation. These case studies were a market simulator for creating offerings (Milidiu et al., 2001), learning agents in a two-person game scenario (Sardinha et al., 2003a), a distributed system for the procurement of travel packages (Sardinha et al., 2005) and a multi-agent system for a supply chain management scenario.

In order to demonstrate the applicability of the proposed method, the LearnAgents (Sardinha et al., 2005) case study is presented. The LearnAgents is a multi-agent system for the procurement of travel packages with multiple and simultaneous auctions that participated in the 2004 edition of the Trading Agent Competition (TAC) travel scenario (Wellman et al., 2003). We present the system specification using ANote, the mapping relations from the ANote artifacts to the ASYNC framework structure and some resulting implementation code.

The paper is organized as follows. In Section 2, we review ANote with a brief description of its concepts and artifacts. Section 3 presents a summarized explanation of the ASYNC framework. Section 4 presents our proposed method for building agent-based systems, which is the main contribution of this work. This section also uses the LearnAgents system to illustrate all the phases of our approach. Section 5 describes some related work and, finally, Section 6 reviews the method contributions and suggests some future work.

2. The ANote

From an analysis point of view, systems including the agent technology require dedicated basic concepts and languages (Luck et al., 2004). ANote (Choren and Lucena, 2004, 2005) is a modeling language that was developed to offer a standard way to describe concepts related to the agent-oriented modeling process. It has an underlying conceptual meta-model (Fig. 1) that identifies the set of concepts and relationships supported by ANote. The conceptual meta-model is used to define scope and to identify the elements that can be present in ANote diagrams. This meta-model defines, at the high level, the interactive, environmental and societal concepts such as goals, interaction protocols, environment, resources and organizations. Furthermore, at the level of individual agents, it has elements to represent basic agent concepts such as actions, communication and plans.

These concepts define a variety of different aspects of concern, or views, which may complement or overlap each other during the system specification. ANote defines seven views based on its conceptual meta-model: goal, agent, scenario, planning, interaction, environmental and organizational views. Each view generates an artifact (diagram) and it is a partial specification that enables the designer to concentrate on a single set of properties each time. Therefore, the designer considers only those features that are important in a particular context.

ANote goal view provides an initial identification of a tree of goals that outline the system functions. Analyzing requirements in terms of goal decomposition and refinement can be seen as teasing out many levels of requirements statements, each level addressing the demands of the next level (Choren and Lucena, 2005). In this view, complex goals can be functionally decomposed into their constituent goals and flows, providing a description as a hierarchical tree of goals.

The agent view specifies the agent classes in the application solution and their relationships. In this view, no details about agent behavior are provided since its purpose is to specify the systems structure. Agent classes specify the roles that will perform the goals elicited in the goal view and that compose the system organizations. When two agents need to inter-