Information Sciences 195 (2012) 190–210

Contents lists available at SciVerse ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Designing a meta-model for a generic robotic agent system using Gaia methodology

Daniel Castro Silva [a,*], Rodrigo A.M. Braga [b], Luís Paulo Reis [b], Eugénio Oliveira [b]

[a] *University of Coimbra, Department of Informatics Engineering / CISUC - Center for Informatics and Systems of the University of Coimbra, Pólo II, Pinhal de Marrocos 3030-290 Coimbra, Portugal*

[b] *FEUP – Faculty of Engineering of the University of Porto, Department of Informatics Engineering/LIACC – Artificial Intelligence and Computer Science Laboratory, Rua Dr. Roberto Frias s/n, 4200-465 Porto, Portugal*

A R T I C L E I N F O

Article history:
Received 6 January 2010
Received in revised form 15 April 2011
Accepted 7 January 2012
Available online 24 January 2012

Keywords:
Multi-robot systems
Open systems meta-model
Agent-oriented software engineering
Gaia

A B S T R A C T

The emergence of multi-agent systems in the past years has led to the development of new methodologies to assist in the requirements and architectural analysis, as well as in the design phases of such systems. Consequently, several Agent Oriented Software Engineering (AOSE) methodologies have been proposed. In this paper, we analyze some AOSE methodologies, including Gaia, which supports the architectural design stage, and some proposed extensions. We then use an adapted version of this methodology to design an abstract generic system meta-model for a multi-robot application, which can be used as a basis for the design of these systems, avoiding or shortening repetitive tasks common to most systems. Based on the proposed Generic Robotic Agent Meta-Model (GRAMM), two distinct models for two different applications are derived, demonstrating the versatility and adaptability of the meta-model. By adapting the Gaia methodology to the design of open systems, this work makes the designers' job faster and easier, decreasing the time needed to complete several tasks, while at the same time maintaining a high-level overview of the system.

1. Introduction

In the past years, systems that support the use of mobile robots have emerged at a growing rate. These systems are used in a wide range of areas, including military [19], medical [9], industrial [49], household applications [31,30], and other smaller or more specific fields of research, such as the Robocup Rescue[1] and Soccer competitions,[2] as well as several projects currently under development. Despite the need to develop coordination methodologies among the several entities, the benefits of a system capable of working in a distributed manner (with simpler entities performing smaller tasks that together contribute to a larger task completion) are certainly greater than the complexities introduced by this model [1].

Several Agent-Oriented Software Engineering (AOSE) methodologies have been proposed over the years to help model multi-agent systems, some deriving from existing more traditional software engineering methodologies (usually object-oriented approaches), others with a more innovative origin. These methodologies are diverse in terms of the development phases they support, from requirement elicitation to implementation.

The main goal of the work presented in this paper is to introduce a meta-model for a generic robotic agent system, that can be used as a basis for the modeling of such systems, thus avoiding or shortening repetitive tasks common to most of these systems. This meta-model can be achieved by using a methodology that supports the architectural design stage.

* Corresponding author. Tel.: +351 91 670 69 14.
E-mail addresses: dcs@dei.uc.pt (D.C. Silva), rodrigo.braga@fe.up.pt (R.A.M. Braga), lpreis@fe.up.pt (L.P. Reis), eco@fe.up.pt (E. Oliveira).

[1] More information available online at http://www.robocuprescue.org/.

[2] More information available online at http://www.robocup.org/.

0020-0255/$ - see front matter © 2012 Elsevier Inc. All rights reserved.
doi:10.1016/j.ins.2012.01.029

One such methodology is Gaia, a somewhat generic methodology that excludes requirements elicitation and implementation, focusing on analysis and design of the system [47]. Even though the Gaia methodology was originally intended to be used in the design of organizational systems, with a more defined set of hierarchical organization, it can also be used in the design of open systems, with minor modifications, as introduced in Section 4.

In the following section, we present a brief overview on some existing AOSE methodologies, as well as a brief analysis on some proposed extensions to the Gaia methodology (Section 2.4). In more detail, we present and analyze:

- the differences between the original Gaia methodology and the official proposed improvements
- the proposed AUML extensions
- the proposed ROADMAP extension
- the extensions and replacements as proposed by [10]

We also introduce a model for an extended version of the Gaia methodology, using the Software Process Engineering Metamodel (SPEM) 2.0 notation – see Section 3. This is believed to be an original contribution that provides a higher formalization and facilitates a better understanding of the Gaia methodology.

In Section 4, we introduce the proposed meta-model for a multi-agent system including several robotic vehicles and some peripheral agents using Gaia-based methodology. We then instantiate the Generic Robotic Agent Meta-Model (GRAMM) into two distinct designs for two different systems (Section 5).

Finally, on Section 6, we present some conclusions about the work that has already been done, as well as the guiding lines for the next steps to be taken.

2. AOSE methodologies and extensions

The ever-growing use of agent-based technology and applications has brought forward the need for methodologies that could aid designers not only in development and deployment, but also in the early analysis and design phases of a project, in a manner similar to what traditional software engineering techniques have done for more conventional software projects, namely when using an object-oriented paradigm [27].

In this section, we briefly introduce some of the most widely known AOSE methodologies that have emerged in the past years (for a more complete and detailed review of existing methodologies, see [4] or chapter 7 of [41]).

2.1. MaSE

The Multiagent Systems Engineering (MaSE) methodology was introduced in 2000 as a comprehensive methodology, including analysis and design stages [46]. It uses a number of graphical models to describe goals, behaviors, agent types, and agent communication interfaces, also providing detailed definition of internal agent design [16]. In the first analysis phase, goals are determined and structured by analyzing an already existing initial system specification. The second analysis phase is centered around use cases, detecting roles, use cases and use case scenarios from the system specification. In the third analysis phase, the identified roles are refined, producing a more detailed description of each role and their respective goals, as well as interactions with other roles. In the design stage, roles are mapped into specific agent classes; communication protocols between agent classes are detailed; the internal details of each agent class are defined, using components and connectors; and finally a system-wide deployment diagram is created. A tool named agentTool was developed to support the MaSE methodology (and more recently the Organization-based MaSE, or O-MaSE), from the initial system specification to implementation, using a set of inter-related graphical models [15].

2.2. Tropos

Tropos was introduced in 2002 as a comprehensive AOSE methodology, encompassing all stages of project design, from early requirements elicitation to detailed design [24]. Tropos can be considered as loosely based on a use-case model used in traditional Software Engineering methodologies. Its key concepts include actor, goal, plan, resource, dependency, capability and belief [8]. During the early requirements elicitation, actors and goals are identified from stakeholders and their objectives, using a goal-oriented analysis. Dependencies between actors and goals are also identified. In the late requirements stage, all functional and non-functional requirements for the system are specified in more detail. In this stage, the system is considered as a single actor, while external entities present in the environment are considered as interacting actors. The architectural design stage produces a model of the system architecture, describing how components work together. During the detailed design stage, detailed models of each component are produced, showing how goals are fulfilled by agents. In this stage, details such as agent communication language and protocols are specified using a more detailed modeling language such as UML (Unified Modeling Language) [23].

2.3. Prometheus

Prometheus was introduced in 2002 as a result of industry and academy experience [38]. It provides support and a detailed process for specification to implementation stages of a project, and includes concepts such as goals, beliefs, plans and