The Knowledge Engineering Review, Vol. 20:2, 99–116. © 2005, Cambridge University Press
doi:10.1017/S0269888905000421 Printed in the United Kingdom

Agent-oriented software engineering

CAROLE BERNON[1], MASSIMO COSSENTINO[2] and JUAN PAVÓN[3]

[1]*IRIT—University Paul Sabatier, 118 Route de Narbonne, 31062 Toulouse Cedex 09, France;*
e-mail: bernon@irit.fr
[2]*Istituto di Calcolo e Reti ad Alte Prestazioni—Italian National Research Council, Viale delle Scienze, ed. 11. 90128 Palermo, Italy;*
e-mail: cossentino@pa.icar.cnr.it
[3]*Facultad de Informática, Universidad Complutense Madrid, Ciudad Universitaria s/n, 28040 Madrid, Spain;*
e-mail: jpavon@sip.ucm.es

Abstract

Considering the great number of agent-oriented methodologies that can be found in the literature, and the fact that each one defines its own concepts and system structure, one of the main challenges in agent-oriented software engineering (AOSE) research is how to make these methodologies interoperable. By defining concepts used in a specific domain in a non-ambiguous way, meta-modelling may represent a step towards such interoperability. Consequently the main objective of the AOSE TFG (Technical Forum Group) is to establish a strategy for identifying a common meta-model that could be widely adopted by the AOSE community. This paper sums up the approach used by this TFG which consists of (i) studying and comparing the meta-models related to some existing methodologies (ADELFE, Gaia, INGENIAS, PASSI, RICA and Tropos) in order to find commonalities and (ii) giving a clear and basic definition for the core concepts used in multi-agent systems for relating and positioning them in a unified MAS meta-model. The first proposal, set up by the working group, for this unified meta-model then concludes this paper.

1 Introduction

The purpose of the Agent-Oriented Software Engineering Technical Forum Group (AOSE-TFG) is the creation of a path towards integration and interoperability of methodological approaches for multi-agent system (MAS) development. This involves the definition of a common framework for MAS specification, which includes the identification of a minimum set of concepts and methods that can be agreed in the different approaches. The tool for defining this framework is meta-modelling. The principle of meta-modelling has been already used in other fields of software engineering, for instance, in the specification of UML by OMG, to describe the elements of the language, their constraints and relationships. This approach is also valid in specifying the concepts that are used by agent-oriented methodologies.

With this assumption, one of the main issues in AOSE TFG has been to elaborate and discuss MAS meta-models for different methodologies. Work on the compilation of one or more MAS meta-models can be used as a reference point by the whole community. Although this work presents several challenges (see Section 4.4.1), all of AOSE TFG members agree that achieving concrete results in this area would be very useful for several reasons: (i) this partly solves the lack of standardization in this area, as it has been remarked in the AgentLink Roadmap (Luck *et al*., 2005), (ii) this could encourage the development of more flexible and versatile design tools, and (iii) this is one of the essential steps for reaching a concrete maturity in the study of the whole agent design process.

https://doi.org/10.1017/S0269888905000421 Published online by Cambridge University Press

The definition of MAS meta-models has led to the identification (and formalization) of a meta-model that AgentLink members hope will meet a sufficient consensus to be adopted as a common reference point by the European research community in this area. Other aspects of agent-oriented software engineering have been faced as well with specific contributions about agent modelling languages and agent mobility design.

This paper is organized as follows: Section 2 provides an over view of the talks and discussion held during AOSE TFG meetings about modelling languages and MAS meta-models, Section 3 introduces the main motivations that are behind the choice of identifying a unified MAS meta-model, Section 4 describes the main sources that have been considered in this unification work, with the results presented in Section 5; finally, some conclusions are drawn in Section 6.

2 Modelling languages and MAS meta-models

Discussions during AOSE TFG meetings were obviously inspired by the main challenges of the field, some of them addressed by Zambonelli & Omicini (2004). More precisely, the lack or the possible improvement of modelling tools concerning agents or multi-agent systems, and the lack of agreement on concepts used in the AOSE area are the main factors explaining the specific attention for modelling languages and MAS meta-models. The most significant contributions on these two topics are reported in the following sub-sections.

2.1 Modelling languages

One of the challenges in the AOSE area is to provide new tools to express and control the behaviour of agents and/or the behaviour of the MAS they are evolving within. Working in such a way would facilitate the spreading of agent-based concepts and applications in the industrial world.

Trencansky (2005) presented an industrial initiative to model MASs: AML (Agent Modelling Language). AML is a semi-visual modelling language, versatile and easy to expand, based on the UML 2.0 specifications, thus allowing one to reuse well-defined concepts and to enable the support of existing CASE tools (Cervenka *et al.*, 2005). AML is designed to support business modelling, requirements specification, analysis, and design of software systems that use MAS concepts and principles. AML has a layered structure built on top of UML to provide an abstract syntax, a semantics and a notation. Expressing the typical features of MASs is done by extending UML with several new modelling concepts (such as agents, resources, environment, role, social aspects or ontologies) while ensuring that the resulting language preserves UML specificities. A non-preserving extension is also provided to add some metaproperties to UML and complete the definition of the AML meta-model. Above this meta-model and notation, two UML profiles for AML are given. They enable the implementation of AML in CASE tools based on UML 1.* or UML 2.0; AML is supported by IBM Rational Rose and LS/TS from Living Systems. Using these AML profiles, a designer is free to customize AML through the definition of extensions to this language. The V.0.9 release of AML has been available since December 2004, and has been submitted to OMG for the RFP on Modelling Agent-Based Systems.

In specific cases, mobile agents are useful, therefore the provision of tools to model their deployment, migration and interactions, for example, becomes an important issue. Kusek & Jezic (2005) made a proposal to model agent mobility with UML sequence diagrams. This work is justified because this mobility is not represented in the current UML sequence diagrams and because modelling agent creation, mobility paths and current location of an agent has not yet been fully addressed by FIPA, UML, AUML or AML. In AUML a deployment diagram can capture the reason why an agent moves and the location to which it moves, and an activity diagram may express when the agent has to move. In UML these expressions are made possible by extending the activity or sequence diagrams and/or defining a stereotype ‘host’. Finally, AML enables a designer to model the structural and behavioural aspects of entity mobility through, as seen above, extension

https://doi.org/10.1017/S0269888905000421 Published online by Cambridge University Press

of UML relationships and definition of a stereotype '<<move>>'. However, all these modelling languages do not give an overall view about agent roaming and execution path. Four solutions, extending UML sequence diagrams, were described to try to tackle this issue. First, with stereotyped mobility diagrams, an agent is located on a node by sending a stereotype message to it and then moves to another node by sending it a message stereotyped with another value. Second, in swimlaned mobility diagrams, a swimlane represents a node and an agent moves in the same way as in the first approach except that it terminates at the source node and is created again on the destination node. Third, in state representation mobility diagrams, the mobility is represented by changing the state of the moving agent. Finally, in frame fragment mobility diagrams, each frame fragment of a sequence diagram represents execution on a node. The advantages and drawbacks of these approaches were then compared, with respect to the number of nodes involved, the space needed for the graphics, the expression of the mobility, using a case study in which agents roam the Internet to find better prices for products.

While in this section we have dealt with agent modelling related issues, in the following section we present MAS meta-models related to the different methodological approaches born in the AgentLink AOSE community.

2.2 MAS meta-models

Several studies have been carried out recently concerning the idea of assembling a new design process for agents by taking methods from existing agent-oriented methodologies. These can be seen as an adaptation of the *method engineering* approach (Brinkkemper *et al*., 1996). Other researchers and software development companies are working on the production of agent-specific design and coding tools or the extension of existing ones. We can say that from requirements identification down to the final deployment of the executable code there are important research activities that while looking at the agent systems indeed lack a well consolidated and sometimes even defined meta-model of the multi-agent society.

With the term ''meta-model'' we mean a model of the concepts that can be used to design and describe actual systems. The models describing a system are instances of the meta-model (i.e. entities of the system model are instances of the meta-model entities). In this way, it is possible to build several models (views) of a system; for instance, one model could represent the organization view of the system at an abstract level (as an instance of the concepts in the meta-model that describe organizational issues) while another could be more implementation oriented and show how the system is deployed in a target platform (as an instance of a platform-specific meta-model).

In the object-oriented' context, the construction of a development process, the definition of its components (analysis, design, testing activities) and the execution of the design rely on a common denominator, the universally accepted concept of the object and related meta-model of the object-oriented system. It is not so in the agent-oriented context, where the lack of such a shared meta-model brings about significant differences in the way different researchers deal with similar (at least in the name) concepts. We are not saying that we desire to achieve a unification of all the agent-oriented development approaches since their number is *per se* a bonus, but instead we mean that a unique, well defined meta-model, including a set of definitions of its concepts, will enable a deeper understanding of the differences between all of these approaches and their integration.

There are several direct applications of meta-models. First, as already argued, meta-models can guide the development process, as they can be seen as templates of what a system model has to be. In the case of MASs, a meta-model specifies what types of entities the developer has to look for: agents, goals, interactions, tasks, resources, etc. It also establishes constraints on the relationships and use of those elements. Meta-models can also be seen as the specification of a modelling language, and therefore they are fundamental in developing tools that can support the development process, as it has been proposed in MESSAGE and implemented by INGENIAS by (Gómez-Sanz & Pavón, 2002).

https://doi.org/10.1017/S0269888905000421 Published online by Cambridge University Press

In the scope of the AOSE TFG, similarly to the work that is going on within the FIPA Methodology TC[1], meta-models are also used to get a better understanding of the agent concepts as applied in different methodologies and to look for some agreement in the main elements that can be used to specify a MAS. In AOSE TFG, several MAS meta-models have been presented and a first proposal of unification, based on three AO methodologies—ADELFE, Gaia and PASSI- has already been published by Bernon *et al.* (2005). Currently, AOSE TFG is considering the MAS meta-models of ADELFE (Bernon & Gleizes, 2005), INGENIAS (Gómez-Sanz & Pavón, 2005), PASSI (Chella *et al.* 2005), RICA (Serrano *et al.*, 2005) and Tropos (Bertolini *et al.*, 2005). Because working on a unifying MAS meta-model was one of the main aims of the AOSE TFG, all of these are described in the next section.

In relation to this activity, Guessoum (2005) proposes an MDA-based approach for MASs to fill the gap between existing MAS tools (such as the multi-agent platforms DIMA, Jade, MadKit or Zeus) and agent-oriented methodologies or meta-models. This approach consists of separating the application logic (described in a PIM—Platform Independent Model) from the underlying technology (described in a PSM—Platform Specific Model). Here, using Meta-DIMA, a MDA-based MAS development process could be as follows: define the PIMs and PSMs by analysing the multi-agent applications, define a library of meta-models by identifying the concepts used and design the transformation rules to implement a meta-model from its description. An initial step has been to try and define a PSM for the multi-agent tool DIMA and PIMs from PASSI and Aalaadin/PASSI meta-models. Some rules were given to enable transformations from these PIMs towards the DIMA-based PSM.

3 Review of MAS meta-models

MAS meta-models from the different AOSE TFG participants became the basis for the construction of the proposed unified meta-model that is presented in this section.

3.1 The ADELFE MAS meta-model

Bernon *et al.* (2004) have developed ADELFE as a methodology devoted to the design of adaptive multiagent systems (AMASs). According to this approach, building a system which realizes the right desired global function (which is functionally adequate) is achieved by designing agents with a cooperation-driven social attitude.

An agent belonging to an AMAS ignores the global function of the system, only pursues a local goal and tries to always keep cooperative relations with other agents. Such an agent is called a 'cooperative agent' because its social attitude is based on cooperation, but its lifecycle is still a classical one. It consists of having perceptions, taking decisions and then doing actions (perceive–decide–act). Local cooperation rules enable the agent to detect and solve non-cooperative situations (NCSs) that play a fundamental part in the ADELFE design. These NCSs are cooperation failures (e.g. cooperation protocol not obeyed, or unpredictable situations) and they are inconsistent with the cooperative social attitude of a cooperative agent.

The MAS meta-model adopted for ADELFE (Bernon *et al.*, 2005) is therefore explained by the features such cooperative agents possess. It tries to constrain the agent behaviour with a cooperative attitude by using local cooperation rules that an agent observes and which enable it to detect and solve NCSs of different kinds (such as incomprehension and uselessness). The concept of environment is essential for MASs, agents are evolving in an environment which can be a physical one, made up of elements of different kinds, or a social one, made up of other agents. An element composing an environment is a specialization of the UML Classifier, more especially an agent can also be viewed as such a specialization. A cooperative agent has interactions with its

[1] http://www.fipa.org/activities/methodology.html

https://doi.org/10.1017/S0269888905000421 Published online by Cambridge University Press

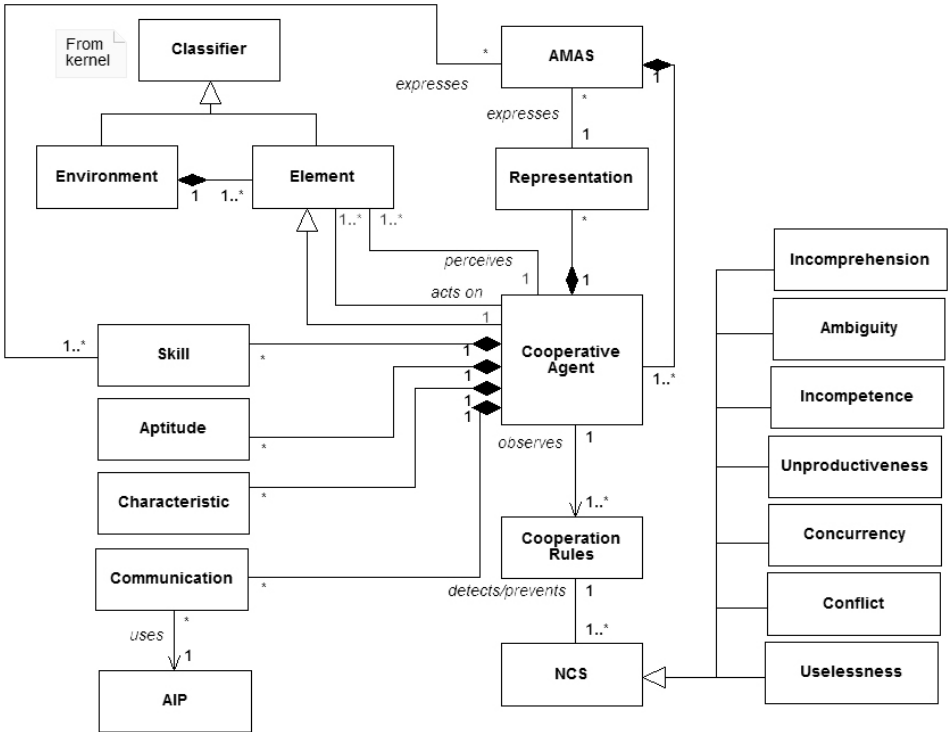


Figure 1 The MAS meta-model adopted by ADELFE

environment, it can receive information through perceptions and act on the environment during its action phase. Interactions may use communications which can be done in a direct manner (by exchanging messages, using AIPs to express the communication pattern, for instance) or an indirect one (environment mediated). An agent has a representation of the world in terms of beliefs about other agents, the configuration of the physical environment around it and the agent itself. The agent uses these representations to determine its behaviour. A representation can be shared by different agents. If an agent has representations that may evolve, they can be expressed using an adaptive MAS.

On the left part of Figure 1, we find Skill, Aptitude and Characteristic; skills represent the specific knowledge that enable each agent to realize its own partial function in the world. If these skills have to evolve, they may also be implemented as an AMAS. Characteristics are intrinsic or physical properties of the agent while aptitudes relate the agent's capability of reasoning both about knowledge and beliefs it owns.

3.2 Gaia

The Gaia methodology evolved from an initial version by Wooldridge *et al.* (2000), mainly focused on the design of small-scale, closed agent-based systems to the new one by Zambonelli *et al.* (2003), which is based on the key consideration that an organization is more than simply a collection of roles and agents. Therefore, the main difference is that it has been designed in order to explicitly model and represent the social aspects of open agent systems, with particular attention to the social goals, social tasks and organizational rules.

Having a deeper look at the MAS meta-model for the second, extended version of Gaia (Figure 2), we notice that the basic building blocks of the former version of Gaia—namely agents, roles, activities, services, and protocols—are still present but now they are located in the context of a specific environment and of a specific organization.

The Gaia agent is an entity that can play one or more roles; a role is a specific behaviour to be played by an agent, defined in terms of permission, responsibilities, and activities, and of its

https://doi.org/10.1017/S0269888905000421 Published online by Cambridge University Press

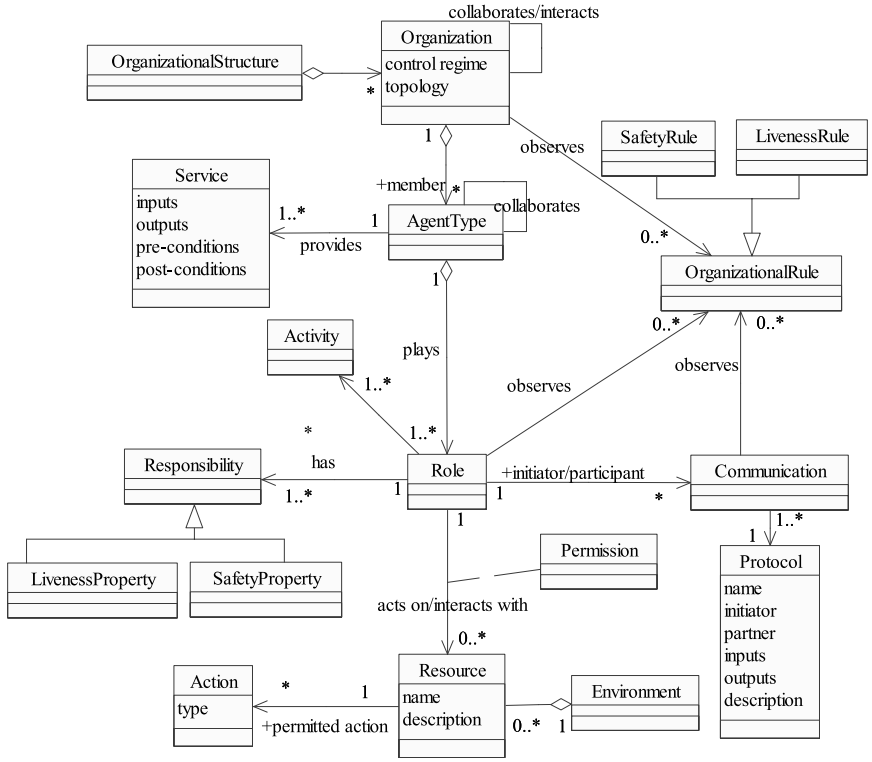


Figure 2 The MAS meta-model adopted by Gaia

interactions with other roles. In playing a role, an agent actualizes its behaviour in terms of services that can be activated accordingly to some specific pre- and post-conditions.

The environment abstraction is a key element in the Gaia MAS meta-model; it explicitly specifies all the entities and resources a MAS may interact with, restricting the interactions by means of the permitted actions.

As already said, the explicit representation of the agent organization is the main improvement in the Gaia extension by Zambonelli *et al.* (2003), this is mainly achieved by introducing the organizational rules and the organization structure. Organizational rules specify some constraints that the organization has to observe; these constraints may be global, affecting the behaviour of the society as a whole, or concerning only specific roles or protocols, while organization structure establishes the overall architecture of the system, that is the position of each role in the organization and its relationship with other roles. Organizational rules and organizational structures are strictly related, in that organizational rules may help designers in the identification of the organizational structures that more naturally suit these rules.

3.3 INGENIAS

INGENIAS (Pavón & Gómez-Sanz, 2003) is both a methodology and a set of tools for the development of multi-agent systems (MASs). It tries to integrate results from other proposals and evolves in accordance with standards and well-proved results in the area. The basis of INGENIAS is the definition of metamodels for MASs, by extending and refining the proposal of the MESSAGE project (Caire *et al.*, 2002). Methods and tools in INGENIAS are defined in terms of meta-models, in such a way that changes in the meta-models are easily adapted in the methods and tools. This facilitates evolution and adoption of new notations and techniques. Currently, INGENIAS provides a set of tools for modelling (graphical editor), documentation of MAS specifications, code generation (for different target platforms), and validation and verification of intentional and social properties of agents.

https://doi.org/10.1017/S0269888905000421 Published online by Cambridge University Press

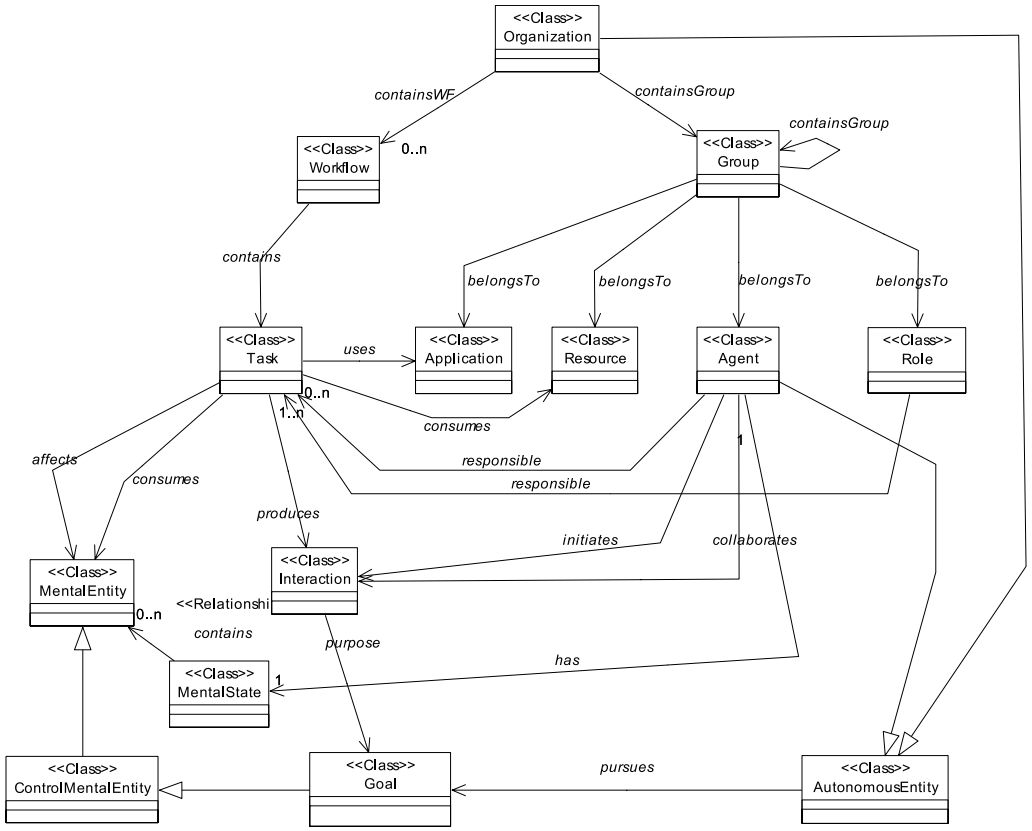


Figure 3 Summary of the INGENIAS MAS meta-model

The INGENIAS meta-model is quite detailed as it is the basis for building the set of tools (editor, code generators, verification and validation) of the INGENIAS Development Kit (IDK, available at `http://ingenias.sourceforge.net`), described in Pavón *et al.* (2005). Figure 3 provides a simplified view of the meta-model with its main elements. But the complete meta-model is structured in terms of five viewpoints.

1. Organization viewpoint. Describes how system components (agents, roles, resources, and applications) are grouped together, which tasks are executed in common, which goals they share, and what constraints exist in the interaction among agents. These constraints are expressed in the form of subordination and client–server relationships.
   An organization is an autonomous entity, which pursues a goal, and can be structured in groups (structural entities), and contains workflows (dynamics of the organization processes). A group may consist of roles, agents, resources, applications. Workflows define precedence relationships among tasks, the resources assigned to tasks and their participants (in terms of roles).
2. Agent viewpoint. Describes single agents, their tasks, goals, initial mental state, and played roles. Moreover, agent models are used to describe intermediate states of agents. These states are presented using goals, facts, tasks, or any other system entity that helps in its state description. This way, an agent model could represent in what state an agent that starts an interaction should be.
   An agent is also an autonomous entity, which plays some roles and pursues goals. It has a mental state, which consists of mental entities, such as goals, facts, beliefs. There is a mental state manager that provides the mechanisms for creating, deleting, modifying mental state entities, and a mental state processor that determines how the mental state evolves and what actions an agent should try.
3. Interaction viewpoint. Describes how interaction among agents takes place. Each interaction declaration includes involved actors, goals pursued by the interaction, and a description of the protocol that follows the interaction.

https://doi.org/10.1017/S0269888905000421 Published online by Cambridge University Press

In INGENIAS, an interaction is initiated by an agent, with some goal (intention). Several agents can participate in an interaction. Several formalisms can be used to describe an interaction, such as UML collaboration diagrams, AUML and GRASIA diagrams.

4. Tasks and goals viewpoint. Describes relationships among goals and tasks, goal structures, and task structures. It is also used to express the inputs and outputs of the tasks and what their effects are on the environment or an agent's mental state.
5. Environment viewpoint. Defines an agent's perception in terms of existing elements of the system. It also identifies system resources and who is responsible for their management.

INGENIAS viewpoints can be complemented with extensions of known notations, such as UML use case diagrams or UML collaboration diagrams. These extensions consist of relating INGENIAS elements with UML entities, for instance use cases with interactions.

Developers should be aware that there are elements that may appear in different viewpoints. This repetition of entities across different viewpoints induces dependencies among them. For instance, the same task entity can appear in an agent view, a task/goal view, an organization view, and an interaction view. Therefore, to completely define a task, creating different diagrams for different views is required. If the developer fails to create all of these diagrams, the system specification may be incomplete. On the other hand, if the developer creates all the required diagrams, but from an organization view a task is assigned to a role and in an agent view it is assigned to another, different role, it could be interpreted that the specification is not consistent. These constraints are checked by the INGENIAS Development Kit.

3.4 PASSI

PASSI (Process for Agent Societies Specification and Implementation) (Cossentino, 2005; Burrafato & Cossentino, 2002) is an iterative-incremental process for designing MASs starting from functional requirements that adopt largely diffused standards like UML (as the modelling language, although extended to fit the needs of agents design) and FIPA (as the agent platform). PASSI covers all the phases from requirements analysis to coding and testing, with specific attention paid to the automation of as many activities as possible with the support of PTK (PASSI ToolKit), a specifically conceived design tool.

The PASSI MAS meta-model (Bernon *et al.*, 2005) is organized in three different domains: the Problem Domain (where requirements are captured), the Agency Domain, which represents the transition from problem-related concepts to the corresponding agent solution (that is a logical abstraction), and the Solution Domain (where the implemented system will be deployed).

The Problem Domain (Figure 4) deals with the user's problem in terms of scenarios, requirements, ontology and resources; scenarios describe a sequence of interactions among actors and the system. Requirements are represented with conventional use case diagrams.

The system environment is depicted in terms of concepts (categories of the domain), actions (performed in the domain and effecting the status of concepts) and predicates (asserting something about a portion of the domain elements); the environment also includes resources that can be accessed by agents.

The Agency Domain includes the agent that is the real centre of this part of the model; each PASSI agent is responsible for accomplishing some functionalities descending from the requirements of the Problem Domain. Each agent during its life can play some roles; these are portions of the agent's social behaviour characterized by some specificity such as a goal, or providing a functionality/service and in so doing it can also access some resources. The Service component represents the service provided by a role in terms of a set of functionalities (including pre- and post-conditions as well as many other details mostly coming from the OWL-S specifications), and can be required by other agents to reach their goals. Agents could use portions of behaviour (called tasks) or communications to actuate the role's aims.

https://doi.org/10.1017/S0269888905000421 Published online by Cambridge University Press

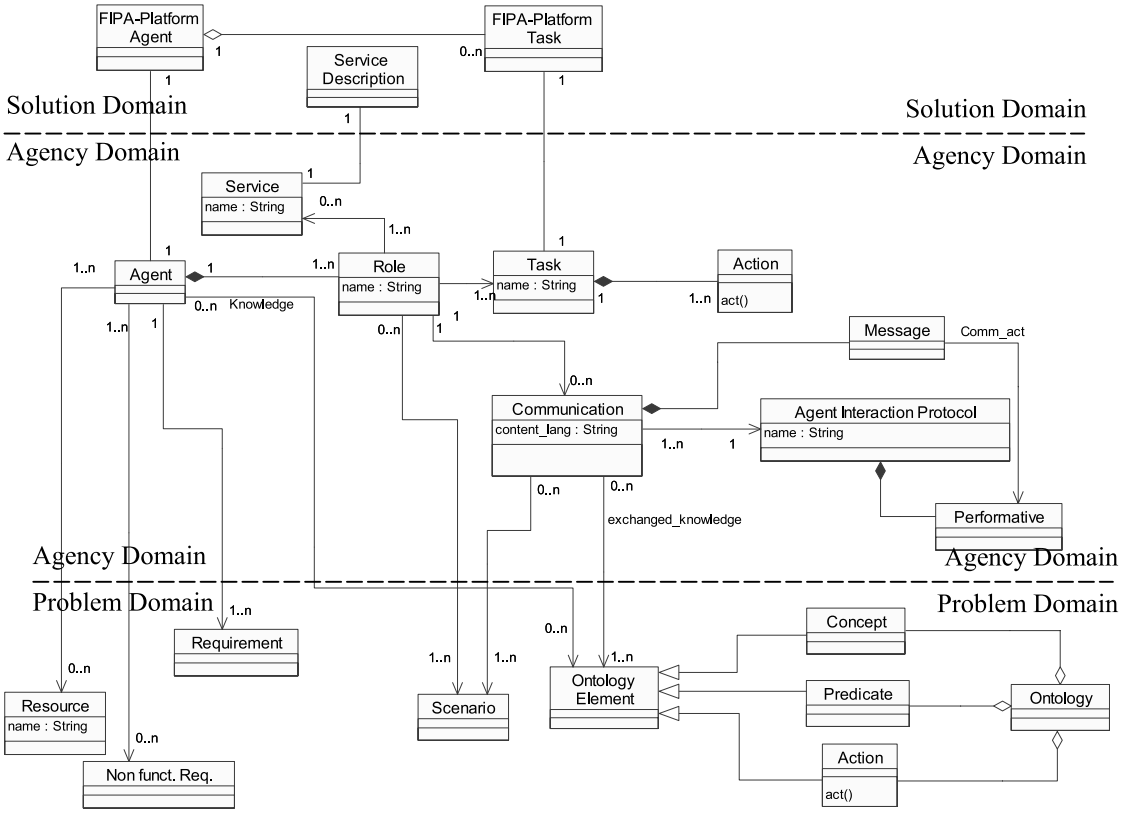


Figure 4 The MAS meta-model adopted by PASSI

In PASSI, the term task is used with the significance of the atomic part of the overall agent behaviour and, therefore, an agent can accomplish its duties by differently composing the set of its own tasks.

A PASSI communication is composed of one or more messages expressed in message content language and following an agent interaction protocol (AIP) composed of performatives (the predefined semantics of the message content; Searle, 1969). This structure is the consequence of the choice of adopting FIPA specifications for the systems to be designed with PASSI.

The Implementation Domain describes the structure of the code solution in the chosen FIPA-compliant implementation platforms and it includes three elements: (i) the FIPA-Platform Agent (the implementation class for the agent entity represented in the Agency Domain); (ii) the FIPA-Platform Task (the implementation structure available for the agent's Task); (iii) the ServiceDescription component that is the implementation-level description (for instance, an OWL-S file) of each service already specified in the Agent Domain. This description is also useful to enable the system openness and the reusability of its components (agents).

3.5 RICA

The RICA (Role/Interaction/Communicative Action) approach (Serrano & Ossowski, 2004) integrates relevant aspects of Agent Communication Languages (ACLs) and organizational models and it is itself based on the concepts of communicative roles and interactions. RICA describes a conceptual framework that guides the designer from the specifications of the organizational model of the agent society to the definition of the ACLs of the MAS.

The organizational model is specified in terms of entities such as agents, roles and interactions, while the specification of the ACL is based on the definition of communicative actions and protocols.

https://doi.org/10.1017/S0269888905000421 Published online by Cambridge University Press

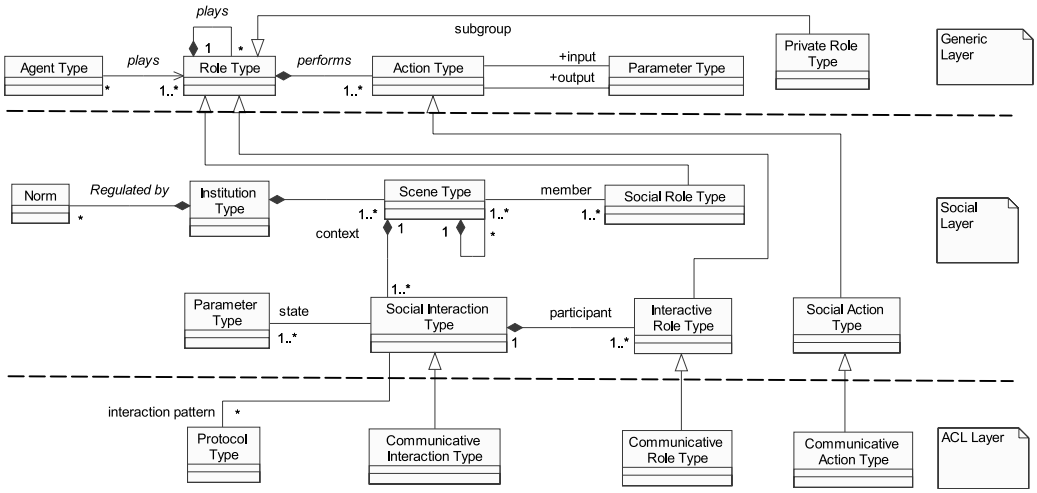


Figure 5 The MAS meta-model adopted by RICA

RICA is essentially made up of two major components: a meta-model (defining the language used to specify the organizational and communicative entities), and a specification of the structure and behaviour of these entities. The RICA MAS meta-model (Serrano & Ossowski, 2004; Serrano *et al.*, 2005) is organized in three different layers: the first deals with the generic elements of the system (agent, role and action types); the second includes social concepts like norms, and institutions; the last layer is devoted to agents' interactions via communications.

In more detail, in Figure 5 we can see that the RICA agent can play several roles; some roles are called 'private' thus meaning that they do not require interactions with other agents but only with the environment. In playing its roles the agent performs some actions characterized by input and output parameters.

In the social layer, generic role types are specialized in social and interactive role types. Social role types participate in scenes (represented by Scene Type in the model) that can be regarded as meeting points used to study interactions. Scene Types are used to build Institution Types that are regulated by Norms. Interactive Role Types regard the agent's social interactions (represented by the Social Interaction Type element) where each agent acts as a participant.

In the third layer, we can find the specialization of some elements of the previous layer according to a communicative direction; this produces such elements as Communicative Interaction Type, Communicative Role Type, Communicative Action Type and Protocol Type.

3.6 Tropos

Tropos (Castro *et al.*, 2001; Bresciani *et al.*, 2004) is an incremental design methodology that aims at modelling both the MAS and its environment. The process covers the early phases of the analysis, where it is characterized by the adoption of a goal-based approach.

From the beginning (Early and Late Requirements analysis), the designer deals with domain stakeholders (modelled as social actors) and the goals they want to reach, resources they can share and plans they may perform. Actors are related by mutual and intentional dependencies that are consequences of their goals. In the next phase (Architectural Design), actors are mapped to a set of software agents while in the final Detailed Design, agents' capabilities and interactions are defined in detail.

In contrast to the other MAS meta-models, Tropos introduces the concept of actor as a generalization of the agent (Bertolini *et al.*, 2005) (Figure 6). This is the direct consequence of the early requirements phase, where actors are initially identified and then translated to possible agents during the architectural design. The role of Tropos is an abstract characterization of the behaviour

https://doi.org/10.1017/S0269888905000421 Published online by Cambridge University Press

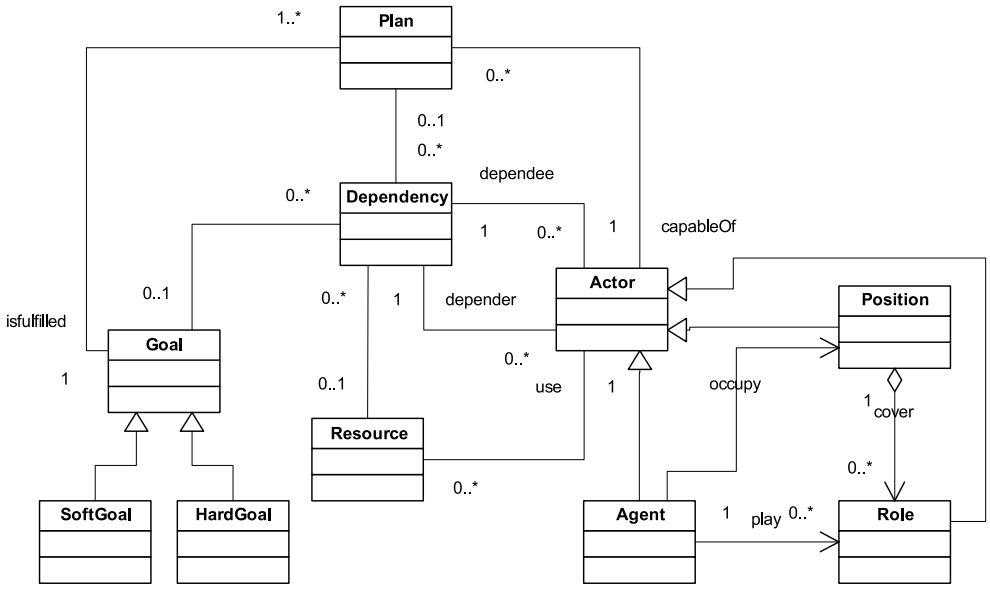


Figure 6 The MAS meta-model adopted by Tropos

of a social actor and a set of roles compose a *position*. The concept of position occurs in the architectural design phase, where agents are used to occupy previously identified positions.

In more detail, we can say that in Tropos, an actor (Bertolini *et al.*, 2005) models an entity with strategic goals and intentionality. An actor can represent a physical or a software agent as well as a role or a position. Goals (representing actors' strategic interests) can be of two different kinds: hard goals and soft goals, the latter have no specific criteria for deciding whether they are satisfied or not and are often used to model non-functional requirements. An actor can achieve his goals by adopting a plan and/or using environment resources. Actors can have a dependency on one another in order to satisfy their own goal or access a resource and their goals can be decomposed in terms of sub-goals that can be related with AND/OR relationships.

There are two other important elements of the Tropos meta-model: the means/end analysis (reporting that a means can satisfy a goal by using plans, resources or other goals) and the contribution (showing that a goal, plan or resource can contribute to the achievement of some objective).

3.7 The first proposal of a unifying MAS meta-model

Starting from the analysis of ADELFE, Gaia and PASSI MAS meta-models a unifying MAS meta-model, first presented in Bernon *et al.* (2005), is shown in Figure 7. This meta-model is guided by the aim of creating societies without (ADELFE) or with predefined organizations (Gaia), in accordance with the growing interest for open systems in which an organization cannot always be given during the design phase. To achieve this result the generic agent is enriched with all the properties an agent may have, being cooperative or not. Furthermore, this generic agent is composed of Gaia-like roles complemented by some PASSI features (tasks and a FIPA-compliant communication structure). This generic agent has two choices: belonging to an organization or following cooperation rules. Agents are implemented (at code level) in the PASSI way. The proposed meta-model is also characterized by the possibility of identifying in it the three domains (problem, agency, solution) discussed in the PASSI approach.

From the experience of merging these three models we learnt that their composition adds some significant improvements to the new structure since they complement each other in several aspects. For example, the ADELFE representation that the agent has of its environment, the Gaia environment and the PASSI ontology, naturally relates by representing the fact that an agent has

https://doi.org/10.1017/S0269888905000421 Published online by Cambridge University Press

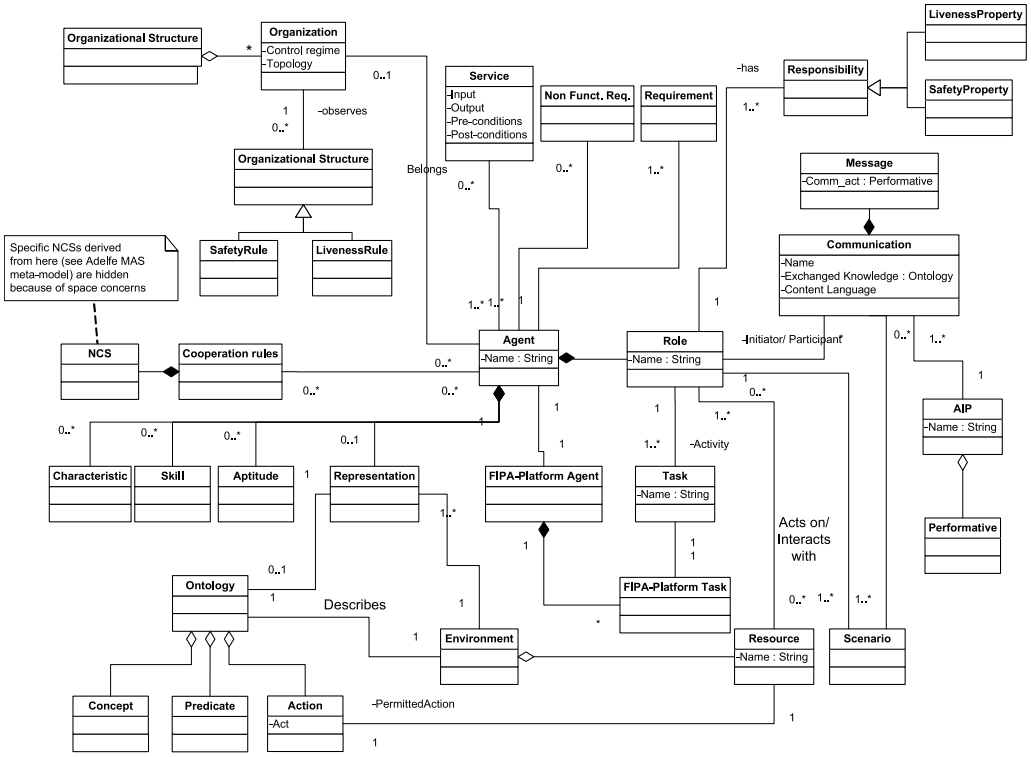


Figure 7 The first proposal for a unifying MAS meta-model

a representation (possibly affected by errors or uncertainty) of the environment expressed in terms of an ontological model of it.

3.8 A comparison of MAS meta-models

The MAS meta-models presented in the previous sub-sections will now be compared with the precise aim of identifying their commonalities as the first step towards a common agreed part of a unified MAS meta-model (MMM), and studying their distinctive differences (that can positively enrich the collection of common elements by introducing ideas coming from just one or a few approaches).

With regard to this comparison, it is worth noting that in their MMM unification proposal (there the term unifying was used to justify the intention of finding a compromise that could summarize the three original MMMs) Bernon *et al.* (2005) proposed a comparison of the different MMMs based on some specific aspects classified according to four different criteria.

- Agent structure: this criterion deals with how each of the meta-models represents its core elements (commonly agents, roles, tasks).
- Agent interactions: how agents of different meta-models are supposed to interact using communications or the environment.
- Agent society and organizational structure: distinct MMMs approach differently the modelling of agent aggregations and cooperation structures.
- Agent implementation: this deals with the way the code-level structure of the agent system is specified.

In this work we are now dealing with a slightly different problem: we are aiming at identifying a unified MMM that could be a common reference point for the AgentLink AOSE community; this means that our attention is initially directed towards the identification of commonalities among the different works we examined; the appreciation of solutions that are not very diffused among the

https://doi.org/10.1017/S0269888905000421 Published online by Cambridge University Press

Table 1 A comparison of the discussed MMMs from the structural point of view

Meta-model	Common components	Peculiarities
ADELFE	Agent (*cooperative*), (*almost*) FIPA communications structure	Environment, cooperation rules, non-cooperative situations (NCSs), skill, aptitude, characteristic
Gaia	Agent (type), role, (*almost*) FIPA communications structure	Organization, service, resource, environment, organizational rules
INGENIAS	Agent, role, task, interaction	Goal, mental state, organization, workflow, group, resource, environment
PASSI	Agent, role, task, FIPA communications structure	Ontology, resources, requirements (functional and non-functional), implementation platform agent and task, service, ServiceDescription
RICA	Agent (*type*), role (*type*), protocol (FIPA communications structure)	Norm, institution type, social/interactive/ communicative role type, action type, social/communicative action type
Tropos	Agent, role, plan (*similar to a task*; Bertolini *et al*., 2005)	Goals, resources, dependency
Unifying MMM	Agent, role, task, FIPA communications structure	Service, environment, characteristic, skill, aptitude, resource, ontology

different MMMs will be part of a second step where specific contributions will be considered to enrich the first-release unified proposal. Because of this different aim, we now prefer to adopt a different comparison method that transversally cuts the previously listed criteria. In Table 1 we reported a list of common and different components of the discussed MMMs.

Elements that can easily be identified as common points of the different MAS meta-models are as follows.

- Agent: it is present in all the methodologies. Gaia and RICA refer to it as *Agent Type*.
- Role: all methodologies but ADELFE include the Role component in their MMMs.
- Task: it is present in INGENIAS, PASSI, and Tropos (with a slightly different meaning: plan; Bertolini *et al*., 2005).
- FIPA communications structure: by this we mean a collection of elements representing agent communications based on messages and following some specific Agent Interaction Protocol (AIP). Several methodologies have good support for this kind of interaction: ADELFE, Gaia, INGENIAS, PASSI, and RICA.

From the analysis of the most characterizing non-common elements of the studied MMMs we can note the following.

- Environment: it is present in ADELFE, INGENIAS, Gaia and the Unifying MMM. In ADELFE it is seen as a possible way of communication for agents, in INGENIAS as application interfaces and resources.
- Organization (and social structures): in Gaia, agents collaborate within organizations that are governed by organizational rules (a similar structure can be found in RICA). Organizations in INGENIAS can be structured in groups (similar to Gaia's organizational structures) and its dynamics are described in terms of workflows.
- Cooperation-related elements: ADELFE has a deep structure about cooperations. It includes cooperation rules and a hierarchy of non-cooperative situations (NCSs); RICA has social roles and actions; in INGENIAS organizations contain workflows describing the collaborative work.
- Mental attitudes and states of agent: the ADELFE agent is modelled in terms of skills, aptitudes and characteristics; the INGENIAS agent is intentional, and has mental states (goals, facts,

https://doi.org/10.1017/S0269888905000421 Published online by Cambridge University Press

beliefs), a mental state manager and a mental state processor; the PASSI agent knowledge is based on an extensive model of domain ontology; Tropos methodology is based on the concept of goal as a problem decomposition entity and each agent acts to reach the goals it has been assigned to.
- Services and resources: Gaia and PASSI define a similar concept of service (PASSI includes a ServiceDescription too as an implementation level transposition of the agent service). Resources are modelled in Gaia, INGENIAS, PASSI and Tropos (in this latter with more details).

The proposed analysis will directly influence the adoption of each single element of the unified MMM described in the next section and the different meanings that each methodology gives to concepts that are similar in their name become a challenge for the definition of the glossary of terms that will complement this unified MMM.

4 Towards a unified MMM: the AgentLink AOSE TFG proposal

In this section we will describe the process that brought us to identify the AgentLink proposal of MMM. The work included the identification of a core subset of MMM components and the clear definition of the meaning of these components by establishing a sort of glossary. Then, defining the relationships between those components, we completed the MMM.

The different talks given during the different meetings of this TFG, as well as the work done in the FIPA Methodology and Modelling TCs, constitute the background of this identification work. The list of sources include:

- existing methodologies (some of them related to authors that are involved in this TFG): ADELFE, Gaia, INGENIAS, PASSI, Tropos and RICA;
- the first reflection on modelling structures that the FIPA Modelling TC proposed (Odell *et al.*, 2005); and
- an attempt to create a unifying meta-model done in Bernon *et al.* (2005).

The second step would be to agree on the components that would be identified and included in a common MAS meta-model.

- In the first live phase, what a MAS meta-model should include will be defined with the definition of concepts like agent, role, task (or plan), communication (message, protocol, performative, etc.).
- In the second live phase, after having launched a discussion during the meeting, concepts such as environment, goal, organization (society, group, institution, etc.), resource, service, would be discussed off-line.
- Other components such as ontology, dependency, action, mental states, organization rules, norms, skills, aptitudes, characteristics (or similar BDI concepts) could be studied later on.

4.1 Identifying the basic components

4.1.1 Defining 'agent'

The starting point for the definition of 'agent' was the definition given by the FIPA Methodology TC[2]. The aim of this discussion was to enumerate the minimal properties an entity must have to be considered as an agent. The essential properties of an agent are its capabilities to act, its autonomy, its communication with others (because the notion of collective is important in many approaches) by interacting, its perception of its environment. Other properties were given, such as proactivity, reactivity, ability to move, or ability to reproduce itself, which could lead to defining different kinds of agents, for instance cognitive or reactive agents. But other agents exist and they are neither

[2] http://www.pa.icar.cnr.it/~cossentino/FIPAmeth/glossary.htm

https://doi.org/10.1017/S0269888905000421 Published online by Cambridge University Press

cognitive nor reactive or may be considered as a mix of these two types. A double inheritance from the 'agent' in the MMM could solve the problem but since it was not mandatory to be exhaustive and define all kinds of agents, this minimal definition for an agent as an entity was agreed:

- capable of acting in its environment;
- autonomous: this means that an agent has control over its own behaviour based on internal or external stimuli;
- can communicate with other agents;
- capable of perceiving its environment.

Some features such as ability to move or reproduce were not considered as mandatory and concepts like skills or capabilities, for example, are included in the requirement that an agent is able to act. Furthermore, since this definition of a 'generic' agent is sufficient to define a reactive agent (to be reactive is a specialization of being capable of acting in an environment), having a definition for this latter concept and differentiating the cognitive nature of an agent from the reactive one was not considered to be useful at this time. A cognitive agent will be considered as an agent:

- which is proactive: this means, its behaviour is driven by a set of tendencies, in the form of individual objectives, or a satisfaction/survival function which it tries to optimize, or desire, or emotion; and
- which uses a representation of its environment.

4.1.2 Defining 'role'

Starting from the definition used in PASSI, a role has been defined as 'an abstraction of a portion of a social behaviour of an agent'.

4.1.3 Defining 'task'

The discussion for defining the concept of 'task' started from the FIPA TC Modelling definition and the one PASSI gives. For FIPA TC Modelling, a task is a part of the agent behaviour and a behaviour is the observable effects of an operation or an event, including the results, and it specifies the computation that generates the effect of the behavioural features. For PASSI, a task specifies the computation that generates the effect of the behavioural features.

To be general enough, the AOSE TFG participants agreed that a task 'specifies a (set of) activity(ies) that generates some effects'.

A role uses observable and non-observable tasks, being characterized by the former since they are visible; an agent could also do something that is not part of playing a role.

4.1.4 Defining 'environment'

The definition of the environment concept involves several aspects (it is in fact the central topic of another AgentLink III TFG, the Environment TFG) and raises a number of questions.

1. PASSI enlarges the software engineering dichotomy about problem and solution domains by introducing the intermediate agency domain. In the same way, could it be possible to speak about a problem environment (in which the system exists), an agency environment (where agents live), and a solution environment (where object-oriented implementations of agents are deployed)?
2. Should the environment be characterized in a similar way to the one proposed by Russell & Norvig (2002) in which an environment can be accessible or not, dynamic or not, deterministic or not, continuous or not?
3. With regard to the solution domain, could we speak about a 'system outer environment', a 'system inner environment' and a 'controllable or not environment'?

Unfortunately, the scope of this TFG was too narrow to have a discussion about what an environment really is and we decided to limit our work to the definition of an environment as

https://doi.org/10.1017/S0269888905000421 Published online by Cambridge University Press

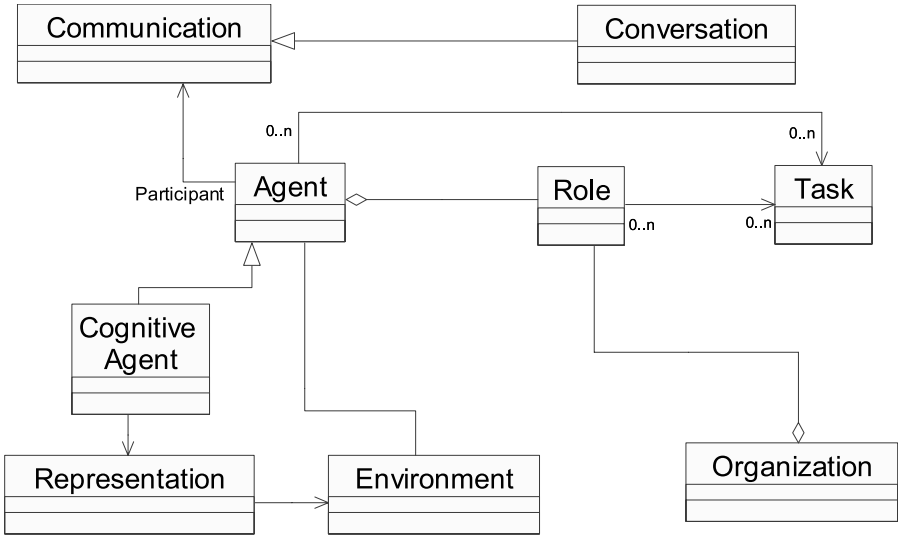


Figure 8 The AgentLink AOSE TFG MMM proposal

'something that an agent can interact with and/or perceive'. This definition goes in the same direction as the one given by the FIPA TC Methodology, in which the environment represents all that is external to the agent, and that includes the social environment and the physical environment.

4.1.5 Defining 'organization'

In Gaia, an organization aggregates agents, and in INGENIAS, an organization is an autonomous entity, with a purpose; the organizational structure is defined with groups.

Organizations can be given by roles (for example, in a soccer match where there is a goalkeeper) but they can also emerge from interactions between agents (for example, in adaptive MASs for which ADELFE is specialized). The discussion concerned the nature of the link between 'agent' and 'organization' because both an agent and an organization can be seen as aggregations of roles. The definition for 'organization' was not completed during the meeting; it is just said to be composed of roles.

4.2 The current unified MMM proposal

With the above reported definition of the concepts, the relationships among them could be drawn and in this way it has been possible to build an initial draft of a unified MMM on which AOSE TFG participants agreed and which is shown in Figure 8.

Obviously, the central concept is a generic agent from which a cognitive agent can be derived. An agent is situated in an environment for which it is able to build representations if it possesses cognitive capabilities. An agent can also be part of the environment of other agents, which explains the bidirectional link between these two concepts. An agent communicates with others and to communicate with an intended purpose can use FIPA-compliant conversations. A role is related to an agent (agents play roles); some agents can use tasks without playing any role and therefore it becomes necessary to relate 'agent' to 'task' with a 0 . . .*n* cardinality. Finally, following on from the previous discussion on the definition of 'organization', an organization may be composed of agents but also of roles depending on the concerned methodology. This is expressed on the MMM by following navigation links from 'agent' to 'organization' via 'role'.

5 Conclusions

As far as agent-oriented methodologies have matured in the last years, there is a need for consolidating concepts and methods for agent-oriented development. In the AgentLink AOSE

https://doi.org/10.1017/S0269888905000421 Published online by Cambridge University Press

TFG, the approach towards this goal has been to study MMMs for different methodologies and to find an agreement on basic concepts. This is a work in progress and the proposed unified MMM and concepts definition will be extended. In the end, this will serve as a foundation for future agent-oriented modelling languages and development tools.

Acknowledgment

We would like to thank all the members of the AgentLink AOSE Technical Forum Group for their active participation and contributions during the AgentLink III Technical Fora.

References

Bergenti, F, Gleizes, M-P and Zambonelli, F (eds.), 2004, *Methodologies and Software Engineering for Agent System: The Agent Oriented Software Engineering Handbook*. New York: Kluwer Academic Publisher.

Bernon, C, Camps, V, Gleizes, M-P and Picard, G, 2004, Tools for self-organizing applications engineering. In *Engineering Self-Organising Systems, Nature-Inspired Approaches to Software Engineering (revised and extended papers presented at the Engineering Self-Organising Applications Workshop, ESOA 2003)* (*Lecture Notes in Artificial Intelligence, 2977*). Berlin: Springer, pp. 283–298.

Bernon, C, Cossentino, M, Gleizes, M-P, Turci, P and Zambonelli, F, 2005, A study of some multi-agent metamodels. In *Agent-Oriented Software Engineering V: 5th International Workshop, AOSE 2004* (*Lecture Notes in Computer Science, 3382*). Berlin: Springer, pp. 62–77.

Bernon, C and Gleizes, MP, 2005, ADELFE MAS meta-model. *AgentLink III—AOSE TFG2*, `http://www.pa.icar.cnr.it/~cossentino/al3tf2/docs/adelfe ppt.pdf`.

Bertolini, D, Perini, A, Susi, A and Mouratidis, H, 2005a, TROPOS meta-model. *AgentLink III—AOSE TFG2*, `http://www.pa.icar.cnr.it/~cossentino/al3tf2/docs/tropos bertolini.ppt`.

Bertolini, D, Perini, A, Susi, A and Mouratidis, H, 2005b, The Tropos Visual Language. A MOF 1.4 compliant metamodel. *AgentLlink III AOSE TFG 2*, `http://www.pa.icar.cnr.it/cossentino/al3tfg2/docs/tropos.pdf`.

Bresciani, P, Giorgini, P, Giunchiglia, F, Mylopoulos, J and Perini, A, 2004, TROPOS: an agent-oriented software development methodology. *Journal of Autonomous Agents and Multi-Agent Systems* **8**(3), 203–236.

Brinkkemper, S, Lyytinen, K and Welke, R, 1996, *Method Engineering: Principles of Method Construction and Tool Support*. London: Chapman & Hall.

Burrafato, P and Cossentino, M, 2002, Designing a multi-agent solution for a bookstore with the PASSI methodology. In *Fourth International Bi-Conference Workshop on Agent-Oriented Information Systems (AOIS-2002)* at *CAiSE'02*, pp. 102–118.

Caire, G, Evans, R, Massonet, P, Coulier, W, Garijo, FJ, Gomez, J, Pavón, J, Leal, F, Chainho, P, Kearney, PE and Stark, J, 2002, Agent oriented analysis using MESSAGE/UML. In *The Second International Workshop on Agent-Oriented Software Engineering (AOSE 2001)* (*Lecture Notes in Computer Science, 2222*). Berlin: Springer, pp. 119–135.

Castro, J, Kolp, M and Mylopoulos, J, 2001, A requirements-driven development methodology. In *Proceedings of the 13th International Conference on Advanced Information Systems Engineering (CAiSE'01)*, pp. 108–123.

Cervenka, R, Trencansky, I, Calisti, M and Greenwood, D, 2005, AML: Agent Modeling Language toward industry-grade agent-based modeling. In *Agent-Oriented Software Engineering V: 5th International Workshop, AOSE 2004* (*Lecture Notes in Computer Science, 3382*). Berlin: Springer, pp. 31–46.

Cossentino, M, 2005, From requirements to code with the PASSI methodology. In Henderson-Sellers, B and Giorgini, P (eds.), *Agent-Oriented Methodologies*. Idea Group Publishing, chapter IV, pp. 79–106.

Cossentino, M, Gaglio, S, Sabatucci, L and Seidita, V, 2005, The PASSI and Agile PASSI MAS meta-models compared with a unifying proposal. In *4th International Central and Eastern European Conference on Multi-Agent Systems (CEEMAS'05)* (in press).

Gómez-Sanz, J and Pavón, J, 2002, Meta-modelling in agent-oriented software engineering. In *Advances in Artificial Intelligence—IBERAMIA 2002* (*Lecture Notes in Artificial Intelligence, 2527*). Berlin: Springer, pp. 606–615.

Gómez-Sanz Jorge, J and Pavón, J, 2005, INGENIAS MAS meta-model. *AgentLink III—AOSE TFG2*, `http://www.pa.icar.cnr.it/~cossentino/al3tf2/docs/ingenias slides.zip`.

Guessoum, Z, 2005, MAS meta-models and MDA. *AgentLlink III AOSE TFG2*, `http://www.pa.icar.cnr.it/~cossentino/al3tf2/docs/zahia slovenia.pdf`.

https://doi.org/10.1017/S0269888905000421 Published online by Cambridge University Press

Kusek, M and Jezic, G, 2005, Modeling agent mobility with UML sequence diagram. *AgentLink III AOSE TFG2*, http://www.pa.icar.cnr.it/~cossentino/al3tf2/docs/kusek ppt.ppt.

Luck, M, McBurney, P and Preist, C, 2003, Agent technology: enabling next generation computing. A roadmap for agent based computing. *Agentlink II*, http://www.agentlink.org/roadmap.

Odell, J, Norine, M and Levy, R, 2005, A metamodel for agents, roles, and groups. In *Agent-Oriented Software Engineering V: 5th International Workshop, AOSE 2004* (*Lecture Notes in Computer Science, 3382*). Berlin: Springer, pp. 78–92.

Pavón, J and Gómez-Sanz, JJ, 2003, Agent oriented software engineering with INGENIAS. In *Multi-Agent Systems and Applications III, 3rd International Central and Eastern European Conference on Multi-Agent Systems (CEEMAS'03)* (*Lecture Notes in Computer Science, 2691*). Berlin: Springer, pp. 394–403.

Pavón, J, Gómez-Sanz, JJ and Fuentes, R, 2005, The INGENIAS methodology and tools. In Henderson-Sellers, B and Giorgini, P (eds.), *Agent-Oriented Methodologies*. Idea Group Publishing, chapter IX, pp. 236–276.

Ralyte, J and Rolland, C, 2001, An approach for method reengineering. In *Proceedings of Conceptual Modeling—ER 2001, 20th International Conference on Conceptual Modeling* (*Lecture Notes in Computer Science, 2224*). Berlin: Springer, pp. 471–484.

Russell, SJ and Norvig, P, 2002, *Artificial Intelligence: A Modern Approach* (2nd edn). Englewood Cliffs, NJ: Prentice-Hall.

Searle, JR, 1969, *Speech Acts*. Cambridge: Cambridge University Press.

Serrano, JM and Ossowski, S, 2004, On the impact of Agent Communication Languages on the implementation of agent systems. In *Cooperative Information Agents VIII, 8th International Workshop, CIA 2004* (*Lecture Notes in Computer Science, 3191*). Berlin: Springer, pp. 92–106.

Serrano, JM, Ossowski, S and Saugar, S, 2005, The RICA metamodel: an organizational stance on agent communication languages. *Agentlink III AOSE TFG 2*, http://www.pa.icar.cnr.it/cossentino/al3tfg2/docs/rica serrano.pdf.

Trencansky, I, 2005, Agent Modeling Language (AML). *AgentLink III—AOSE TFG2*, http://www.pa.icar.cnr.it/~cossentino/al3tf2/docs/aml trencansky.pdf.

Wooldridge, M, Jennings, NR and Kinny, D, 2000, The Gaia methodology for agent-oriented analysis and design. *Journal of Autonomous Agents and Multi-Agent Systems* **3**(3), 285–312.

Zambonelli, F, Jennings, N and Wooldridge, M, 2003, Developing multiagent systems: the Gaia methodology. *ACM Transactions on Software Engineering and Methodology* **12**(3), 417–470.

Zambonelli, F, & Omicini, A, 2004, Challenges and research directions in agent-oriented software engineering. *Journal of Autonomous Agents and Multiagent Systems* **9**(3), 253–284.

https://doi.org/10.1017/S0269888905000421 Published online by Cambridge University Press