Agent-Oriented Software Engineering Technical Forum Group

Report of the TFG Activity

26, November 2005

Massimo Cossentino, ICAR - Consiglio Nazionale delle Ricerche, Italy, cossentino@pa.icar.cnr.it
Juan Pavón, Universidad Complutense Madrid, Spain, jpavon@sip.ucm.es
Carole Bernon, Université Paul Sabatier, IRIT, France, bernon@irit.fr

1 Introduction

The Agent-Oriented Software Engineering Technical Forum Group (AOSE TFG) has been established with the purpose of creating a path towards integration and interoperability of existing methodological approaches for the development of Multi-Agent Systems (MAS). 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 to specify 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 reference points by the whole community. Although this work presents several challenges, all of us 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, (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.

Complementary to the definition of such a common meta-model for MAS, AOSE TFG has also proposed a framework for the evaluation of agent-oriented methodologies. This was set up in order to gain knowledge of the different methodologies from the pragmatics and the underlying theories. In the end, the results of evaluating methodologies can be useful for the selection of method fragments in a method engineering approach.

This report describes the evolution of the activities of the AOSE TFG along the different meetings, which have driven the work in these issues. Section 2 describes the activity of the first TFG meeting (in Rome). Section 3 describes the activity of the second TFG meeting (in Ljubljana). Section 4 describes the activity of the third TFG meeting (in Budapest). Finally, Section 5 draws some conclusions on the overall TFG activity.

2 TFG 1 - Rome Event

The AOSE TFG activity in Rome was divided into two different sessions, both of them scheduled for Friday, (2nd July): the first session has been held in the morning and dealt with methodologies and MAS (multi-agent system) meta-models. In the second session (in the afternoon), we discussed about methodologies evaluation, ideas for future activity of this TFG and future trends in AOSE in general have been discussed too.

More than forty people attended the two different sessions of the AOSE meeting and seventeen talks have been presented.

In the following the discussed topics, presented talks and debates outcomes will be presented in their chronological order.

2.1 Methodologies

More than twenty different design methodologies can be found in literature for the design of MASs. This can be seen as a proof that agent designers in accomplishing their different tasks, and solving specific problems in distinct production environments, often prefer to setup a methodology specifically tailored for their own needs instead of reusing existing ones. What seems to be widely accepted is that a unique specific methodology cannot be general enough to be useful to everyone without some level of personalization.

In this scenario we can identify two contrasting elements: first, in an interesting paper on object-oriented software (development) processes, Fuggetta (in "Software Process: a Roadmap", 2000) states that the research in this field is stuck and most technologies developed by the software process community have not been adopted by the industrial world; second, the AgentLink community, in its roadmap for agent based computing, embodying the feelings of a large part of the agent research and industrial community, has identified the designation of a standard in design methodologies as an essential demand.

In order to accomplish this request, during this first meeting of the AOSE TFG, we focused our attention to the exploration of existing methodologies and the study of possible unification strategies.

2.1.1 Methodologies Overview

Several methodologies (eight) have been presented during the meeting; they gave an idea of the relevant effort spent by the European agent community on this topic.

2.1.1.1 IDE-eli

The Integrated Development Environment for Electronic Institutions (IDE-eli) is a set of tools aimed at supporting the engineering of multi-agent systems as electronic institutions. Software agents appear as the key enabler technology behind the electronic institutions vision. Thus, electronic institutions encapsulate the coordination mechanisms that mediate the interactions among software agents representing different parties. IDE-eli allows for engineering both electronic institutions and their participating software agents. Notably, IDE-eli moves away from machine-oriented views of programming toward organizational-inspired concepts that more closely reflect the way in which we may understand distributed applications. It supports a top-down engineering approach: firstly the organization, secondly the individuals. IDE-eli is composed of:

- **ISLANDER.** A graphical tool that supports the specification of the rules and protocols in an electronic institution.
- **AMELI.** Software platform to run electronic institutions. Once an electronic institution is specified with ISLANDER, it is ready to be run by AMELI without any other programming effort.
- **aBUILDER.** Agent development tool.
- **SIMDEI.** Simulation tool to animate and analyse specifications created with ISLANDER prior to the deployment stage.

Many more application areas may be tackled with the aid of the IDE-eli tools. In general terms, the electronic institutions approach is deemed as appropriate in complex domains where multiple partners are involved, and a high degree of coordination and collaboration is required. Thus, electronic institutions look promising to support workflow management, monitoring and management of shop-floor automation, or supply network management issues.

2.1.1.2 INGENIAS

The purpose of INGENIAS is the definition of a methodology for the development of MAS, by integrating results from research in the area of agent technology with a well-established software development process, which in this case is the Unified Software Development Process. This methodology is based on the definition of a set of meta-models that describe the elements that form a MAS from several viewpoints, and that allow to define a specification language for MAS. The viewpoints are five: agent (definition, control and management of agent mental state), interactions, organization, environment, and goals/tasks. These meta-models are the basis for building the tools for the creation and management of MAS specifications, and code generation to different target platforms.

2.1.1.3 MetaDIMA

MetaDIMA aims at bridging the gap between existing agent architectures with their development tools and agent-based methodologies. The idea is to start from existing architectures and tools and to elaborate meta-models in order to formulate the necessary knowledge about the development process.
MetaDIMA is inspired by the Model-Driven Architecture (MDA), proposed by OMG, which aims at separating application logic from the underlying technologies to improve reusability and development process.
The aim of this project is to illustrate the use of MDA to define a new multi- agent system development process that is based on a library of meta-models. These meta-models may be used by the designer to easily describe the multi-agent models which are automatically transformed in order to generate a runnable multi- agent system.

2.1.1.4 Agile PASSI

In Agile PASSI, the primary requirement is related to not distracting developers from their main goal (producing a system that solves some kind of algorithmic problem, for instance in robotics) with a long design process.
This could be achieved by using an agile process that supports a light (manual) design phase while it encourages the reuse of existing contributions in form of patterns; supporting design tools automatically produce a consistent documentation at different levels of abstraction. Agile PASSI takes advantage of the work done with PASSI and reuses a couple of its most important features: (i) the identification of agents as a set of functionalities expressed in form of use cases, and (ii) the central role of ontology description in analyzing the agent solution.

2.1.1.5 RIO

The RIO meta-model is centred on two concepts: role and organization. A role is an abstraction of a behaviour or a status in an organisation, an organization is a set of roles where each role interacts with at least one another in the organization.
In RIO, formal specifications (obtained with ObjectZ and statecharts) are linked to implementation and the designer work is supported by tools that allow the simulation of the system and its verification.

2.1.1.6 ADELFE

ADELFE aims at covering all the phases of a classical software design: from requirements to deployment. A well-known process, the RUP (Rational Unified Process), has been tailored to take into account specificities coming from the design of Adaptive Multi-Agent Systems (AMAS). Only the requirements, analysis and design phases require modifications in order to be adapted to AMAS, all the others appearing in the RUP remain the same.

Three main tools are integrated into the ADELFE toolkit: an interactive tool which describes the process and helps the designer to apply it, OpenTool, a graphical modelling tool, and a tool which analyses if using the AMAS technology is useful to implement the target system.

2.1.1.7 MaSE

The Multi-agent Systems Engineering (MaSE) is a general purpose methodology for developing multi-agent systems that is founded on basic software engineering principles. MaSE divides the development process into two major phases: the analysis phase and the design phase. Each phase is composed of a set of stages.
The development process adopted in MaSE is iterative. This means that any stage can be repeated many times until the final design is achieved and at any stage a designer may go back to a previous stage.
MaSE is supported by a CASE tool (called agent Tool) that has the capability of performing the analysis and design activities, validating the various models, generating automatic transformation of models, and generating skeleton code. A specific class library is available, called agentMOM, that can be used for code generation.
MaSE also supports the specification of mobility aspects and the definition of ontologies.

2.1.2 Unification Proposals

In the unification direction two different talks have been proposed. In the first talk, A. Garro presented his approach on the composition of a new methodology starting from fragments of other existing methodologies. In the second talk, J. Pena presented a fragment from the MacMAS methodology dealing with the analysis of the acquaintance organization in a MAS. These contributions will now be presented more in details.

2.1.2.1 A Methods Integration Approach

This work refers to the FIPA Methodology Technical Committee activity and it consists in a quite open approach that allows the composition of elements coming from a repository of fragments of existing design processes that could be expressed in terms of a standard notation.
Specifically dealing with the methods integration problem in this contribution two different approaches have been considered to obtain methods integration: (i) guided by a (MAS) meta-model; (ii) guided by a development process.
In the first approach (guided by a MAS meta-model), while building his own methodology, the designer has preliminary to identify the elements that compose the meta-model of the MAS he is going to build; then, he has to choose the method fragments that are able to produce the identified meta-model elements. The second approach (guided by a development process) focuses on the instantiation of some software development process that completely covers the development of MAS. Given a specific problem and/or an application domain, the process will be instantiated by selecting, for each phase, suitable method fragments, chosen from agent-oriented methodologies proposed in the literature or ad-hoc defined.

2.1.2.2 An Acquaintance Organization Fragment

The methodology fragment proposed by J. Pena consists in a systematic, iterative and incremental approach for the development of the acquaintance organisation (interaction organisation) of a MAS structuring models in a set of abstraction layers. It is specially tailored to deal with complexity derived from interactions, thus, we focus all the modelling process on them. This fragment is a part of MacMAS that is a fractal methodology where a model is refined by its bottom layer models and abstracted by its top layer model

2.2 MAS Meta-models

In the object-oriented context the construction of a new methodology and the execution of the design process, rely on a common denominator, the universally accepted concept of object and related meta-model of the object-oriented system. The situation, in the agent-oriented context, is quite different since there is not a commonly accepted definition of the concept of agent and related meta-model of the multi-agent system (a structural representation of the system elements like agent, role, behaviour, ontology, etc. that compose the actual system together with their relationships).

In this case, one of the first steps of a new methodology composition process, should therefore consist in an accurate analysis of the MAS meta-models in order to identify which elements will compose the system that will be built. Its availability will not only be helpful for defining the whole methodology but it will help at a practical level too: the same fact of clearly declaring the structure of the system will enable the development of design tools that are able to check for design correctness and find parts that are not completely/coherently defined. In this view, the final result of the design activity should, obviously, consist in a model of the system - an instantiation of the MAS meta-model – that solves the faced problem.

The situation up-to-date is that the lack of a unique MAS meta-model leads each methodology to have its own concepts and system structure. Starting from the proposed contributions (four talks) during this meeting our group tried to establish a strategy that could bring to the identification of a common MAS meta-model that could be widely adopted.

There were three contributions on already adopted meta-models and one unification proposal. They are presented in the following.

2.2.1 MAS Meta-models of Existing Methodologies

2.2.1.1 INGENIAS

The INGENIAS agent meta-model defines an isolated agent. The agent concept underlying this meta-model is the one defined by Newell in "The Knowledge level" (1982). An agent is therefore seen as a program that exists at the *knowledge level*. It has a physical body with which it can act in the environment, a knowledge body which contains whatever the agent knows at some time, and a set of goals. Also, an agent behaves according to the principle of rationality (*if an agent has knowledge that one of its actions will lead to one of its goals, then the agent will select that action)*. Following this definition, an agent has goals and there should be some association type between agent tasks and goals.

The agent has a mental state which is used to decide what to do next; this mental state is managed by the *MentalStateManager* that is in charge of adding/removing knowledge as well as consistence maintenance. Agent's mental state consists of control entities and information entities. Control entities specify what is expected from the agent, whilst information entities describe the state of the world as seen by the agent.

On the other side, the INGENIAS agent plays roles in several workflows in the system. The association of an agent to a role means that the agent acquires all the properties and responsibilities assigned to the role (goals and interactions in which the role participates).

Social aspects in this approach are abstracted in the organization concept. An *organization,* in the INGENIAS approach, characterizes a group of agents that work together towards a common goal (*purpose*). The organization may consist of

only one agent or several groups of cooperating agents, which form part of *organizational structures* that establish relationships among them.

2.2.1.2 MacMAS

In the MacMAS approach interactions are the central concept of each methodology fragment, and of the whole MAS meta-model.
This has been done to improve the ability to deal with systems with a high interaction degree. An interaction can relate an arbitrary number of roles (multi-Role Interactions) and can be refined by several finer–grain interactions.
This allows performing a layered description of the system which improves the possibility of dealing with complexity through an iterative and incremental process.
MacMAS is completed by a set of automatic algorithms to compose and decompose interactions that also helps in the transition between layers.
Interactions are related to systems goals and this enhances the traceability between requirements and final models.

2.2.1.3 Software Agents vs. Software Components

This work is based on the hypothesis that given a component, we can ascribe mental qualities, i.e., knowledge and goals, to it; this brings to the consideration that components described in terms of their mental qualities are agents.
In this approach, the goal consists in enhancing the possibility of reusing significant parts of a MAS, this can be done by adding some constraints on software infrastructures to better support reusability. Particularly infrastructures should be a means for: (i) transferring knowledge and goals (mainly goals), and not only for moving data, (ii) supporting agents finding each other transparently, i.e., without an explicit request (from the agent or from the programmer), (iii) allowing agents finding problem solvers and not just task executors, (iv) enabling choosing a problem solver on the basis of what it can do and not on how it can describe its capabilities.
In other words some “intelligence” should be moved to the infrastructure but still considering that the problem should not be moved there, i.e. the system designer should not program the infrastructure.

2.2.2 Unification Proposal

One contribution in this direction has been proposed by P. Turci Her talk started from the consideration that some of the agent-oriented methodologies that exist nowadays are dealing with specific kinds of agents or multi-agent systems; this is, for instance, the case of ADELFE, Gaia and PASSI. ADELFE is devoted to cooperative agents and adaptive MAS, while Gaia aims more at creating social organisations and PASSI, the more general one, considers the whole life-cycle from the problem domain to the agent-based solution and the final level code implementation but it limits its scope to FIPA-compliant systems. These differences are reflected in the meta-models elaborated by respecting authors to express the concepts used in their design activities.
In the talk, these meta-models have been compared in order to begin a unification work. One of the most interesting result is the discovery that all of the three (meta-)models share common concepts such as agent and interaction protocol, while other elements are present only in some of them: this is the case of ADELFE and Gaia that share the communication and environment notions, while Gaia and PASSI have in common notions like role and service. Some concepts are only appearing in one of the three meta-models, for instance, responsibilities in Gaia, ontology in PASSI or representation (of others) in ADELFE.

Putting these different meta-models together, comparing their similarities and differences has enabled enriching them mutually as well as unifying the different used concepts and the work resulted in the unified MAS meta-model that has been presented at the end of the talk.

2.3 Methodologies Evaluation (Guidelines and Techniques)

There were two presentations concerning the evaluation of methodologies. One more generic, from A. Sturm ("Evaluation Techniques for Agent-Oriented Methodologies") and the other referring to a concrete aspect, which is the way methodologies deal with complexity, specially for the modelling of interactions in MAS (J. Peña and R. Corchuelo, "Guidelines, Techniques and Modelling Artefacts at the Analysis Stage of AOSE Methodologies to Deal with Complexity").

2.3.1.1 Dealing with Complexity at the Analysis Stage

The first presentation in this section proposed a RFC (Request for Comments) about guidelines, techniques and modelling artefacts at the analysis stage of AOSE methodologies to deal with complexity. The complexity here is related to the interactions between agents and it is closely linked with the organizational aspects. The proposed approach is specially tailored to deal with complicated organizations and is based on three principles (initially identified by N. Jennings):

- Abstraction that defines simplified models of the system emphasising some details while avoiding others in order to focus the designer's attention,
- Decomposition that applies the "divide and conquer" principle to limit the designer's scope to a portion of the problem,
- Composition that identifies and manages inter-relationships between the different sub-systems. It is then possible to group together various basic components to treat them as higher-level units of analysis.

These principles have to be taken into account at the modelling process level which is not the case in most of the existing methodologies. In order to do that, a set of features that should be provided by AOSE methodologies at the analysis stage is proposed. These features are divided into:

- Techniques that are procedures to transform models,
- Modelling artefacts that are icons to graphically represent the system,
- Guidelines that indicate the best practices to deal with complexity by using the previous techniques and modelling artefacts.

The RFC presents the three kinds of principles according to static and dynamic aspects:

- The acquaintance aspect models the relationships between agents in the system from the interaction point of view.
- The behaviour aspect models the order of apparition of these relationships over time.

To evaluate an analysis stage, this RFC can be found on-line via a form in which the different features are asked and can be weighted according to an agreement level (which ranges from 0 to 100) and the ability to conquer complexity (with the same range).

2.3.1.2 Evaluation of Agent-Oriented Methodologies

The second presentation of this session was based on the fact that some comparisons between different methodologies (according to different criteria depending on the goal of the work) can be found in literature but no guideline or tool really exist to address advantages or drawbacks of methodologies or even to compare them.
Considering that a methodology is an entire set of guidelines and activities (lifecycle process, concepts and models…), the presented evaluation framework was based on four categories:

- Concepts and properties. Since no consensus exists within the agent community yet, the basic concepts related to agents and MASs were determined (autonomy, reactiveness, proactiveness and sociality) and then used as properties to evaluate whether an agent-oriented methodology is close to these concepts. Another evaluation determines if a methodology supports building blocks (e.g. agent, belief, desire, intention, message, role, task…), the associated notations and to what extent.
- Notations and modelling techniques. Some properties (e.g. accessibility, analyzability, modularity…) were considered to evaluate how a methodology deals with symbols used to represent elements within a system and how it deals with models depicting aspects of a system at different levels of abstraction.
- Process. The development process is evaluated according to the development context and the lifecycle coverage (from requirements to implementation and tests). The evaluation should also consider whether this methodology provides a detailed description of the various activities of the development lifecycle in order to ease its use.
- Pragmatics. The evaluation of a methodology must take into account how this methodology deals with practical aspects of deploying and using it in a real project or organization. In this case, issues such as available resources, required expertise, language suitability, domain applicability and scalability have to be considered.

Finally, based on academic experiments, some results were given concerning the evaluation of four selected methodologies: Gaia, MaSE, Tropos and OPM/MAS.

Taking into consideration the goals of the AOSE TFG, it seems that evaluation is one of the first tasks we should address. Different techniques are being proposed: feature-based comparison, meta-modelling, metrics, ontological evaluation, and experimentation with case studies. How to organize a common activity in the AOSE TFG towards evaluation stays as an open issue in this meeting, but it is recognized as one of the issues to address in the next ones.

2.4 Open Directions in AOSE

In the last few years, together with the increasing acceptance of agent-based computing as a novel software engineering paradigm, there has been a great deal of research related to the identification and definition of suitable models and techniques to support the development of complex software systems in terms of multi-agent systems. These researches, which can be roughly grouped under the term “agent-oriented software engineering”, are endlessly proposing a variety of new metaphors, formal modelling approaches, development methodologies and modelling techniques, specifically suited to the agent-oriented paradigm.
However, the research in the area of agent-oriented software engineering is still in its early stages, and several challenges need to be faced before agent-oriented software engineering can fulfil its promises, becoming a widely

accepted and a practically usable paradigm for the development of complex software systems. One possible way to identify and frame the key research challenges in the area of agent-oriented software engineering is to recognize that such challenges may be very different depending on the "scale of observation" adopted to model and build a software system.

At one extreme, the micro scale of observation is that where the system to be engineered has to rely on the controllable and predictable behaviour of (a typically limited number of) individual agents, as well as on their mutual interactions. There, the key engineering challenges are related to extending traditional software engineering approaches toward agent-oriented abstractions. Although the use of AUML and of a traditional "waterfall" process model – as promoted by current researches – may have been of some use in the first years of research, the current understanding is that brand new modelling and notational tools, as well as possibly brand new software process models may be needed, to exploit in full the potentials of the agents paradigm.

At the other extreme, the macro scale of observation is the one where a multi-agent system is conceived as a multitude of interacting agents, for which the overall behaviour of the system, rather than the mere behaviour of individuals, is the key of interest. In this case, a discipline of agent-oriented software engineering should focus on totally different problems, and should be able to develop novel "systemic" approaches to software engineering, possibly getting inspiration from areas such as complex systems sciences and systemic biology.

In between, the meso scale of observation is that where the need of predictability and control typical of the micro scale clashes with the emergence of phenomena typical of the macro scale. Therefore, any engineering approach at the meso scale requires accounting for problems that are typical of both the micro and the macro scale, and possibly for new problems specific to the meso scale. These include: identifying the boundaries of a systems – which may be challenging in the case of open multi-agent systems; electing trust as a primary design issue; identifying suitable infrastructures for multi-agent systems support.

3 TFG 2 - Ljubljana Event

During the second meeting, the proposed aims were more especially focused on refining the contributions about MAS meta-models that some supporters presented. This activity led to the identification (and formalization) of a meta-model that members hope should 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.

Section 3.1 provides a complete overlook on the talks and discussion held during the meeting, section 3.2 describes the main sources that have been considered in this unification work, whose result is presented in section 3.3.

3.1 Modelling Languages and MAS Meta-models

Discussion sessions in the second meeting were justified by the challenges derived from the first AOSE TFG meeting in Rome and more especially addressed in the last talk given by F. Zambonelli [33] about open directions in AOSE. 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 talks given in Ljubljana.

3.1.1 Modelling Languages

As discussed in Rome, one of the challenges in the AOSE area was to give 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.

The first talk came from this industrial world with the presentation of AML (Agent Modelling Language) by I. Trencansky [30]. AML is a semi-visual modelling language, versatile and easy to expand, based on the UML 2.0 specifications thus allowing to reuse well-defined concepts and to enable the support of existing CASE tools [12]. 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 multi-agent systems 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 meta-properties 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 implementing AML in CASE tools based on UML 1.* or UML 2.0; in concrete, AML is supported by IBM Rational Rose and LS/TS from Living Systems. Using these AML profiles, a designer is free to customise AML through the definition of extensions to this language. The V.0.9 release of AML is 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 then an important issue. Mario Kusek [20], talked about how 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 where 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 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 than 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 dealt with agent modelling related issues, in the next one we present MAS meta-models related to the different methodological approaches born in the AgentLink AOSE community.

3.1.2 MAS Meta-models

Several studies have been carried on recently about 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 [8][24]. Other researchers and software developing 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 of 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 design 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 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 to significant differences in the way different researchers deal with similar (at least in the name) concepts. We are not here saying that we desire to achieve an unification of all the agent-oriented design approaches since their number is *per se* a richness, 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 of 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 MAS, 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 [17].

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.

During the Rome meeting several MAS meta-models were presented and a first proposal of unification, based on three AO methodologies – ADELFE, Gaia and PASSI, was discussed. Nevertheless, more work had to be done during the second meeting in order to continue the effort of unification and to make a further step towards a reference point for the whole European agent community.

Several MAS meta-models were presented in Ljubljana to enrich the previous discussion, they included the MAS meta-models of ADELFE presented by C. Bernon [4], INGENIAS presented by Jorge J. Gómez-Sanz [15], PASSI presented by L. Sabatucci [13], RICA presented by J. Manuel Serrano [29] and Tropos presented by D. Bertolini [5]. Because working on a unifying MAS meta-model was one of the main aims of the AOSE Technical Forum Group (TFG), all of these talks will be more detailed in a dedicated section (the next one).

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

Finally, Zahia Guessoum [18], talked about an MDA-based approach for MAS meta-models and how 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 using the OMG's MDA (Model Driven Architecture) approach. This approach consists in separating the application logic (described in a PIM – Platform Independent Model) from the underlying technology (described in a PSM – Platform Specific Model). She gave an overview of MetaDIMA, a MDA-based MAS development process, that could be: 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. A first step has been done trying to 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.

3.2 Review of MAS Meta-models

In this section we describe the details of the MAS meta-models discussed and presented during the Ljubljana meeting. They became the basis for the construction of the proposed unified meta-model that is presented in the next section and is the major outcome of this event.

3.2.1 The ADELFE MAS Meta-model

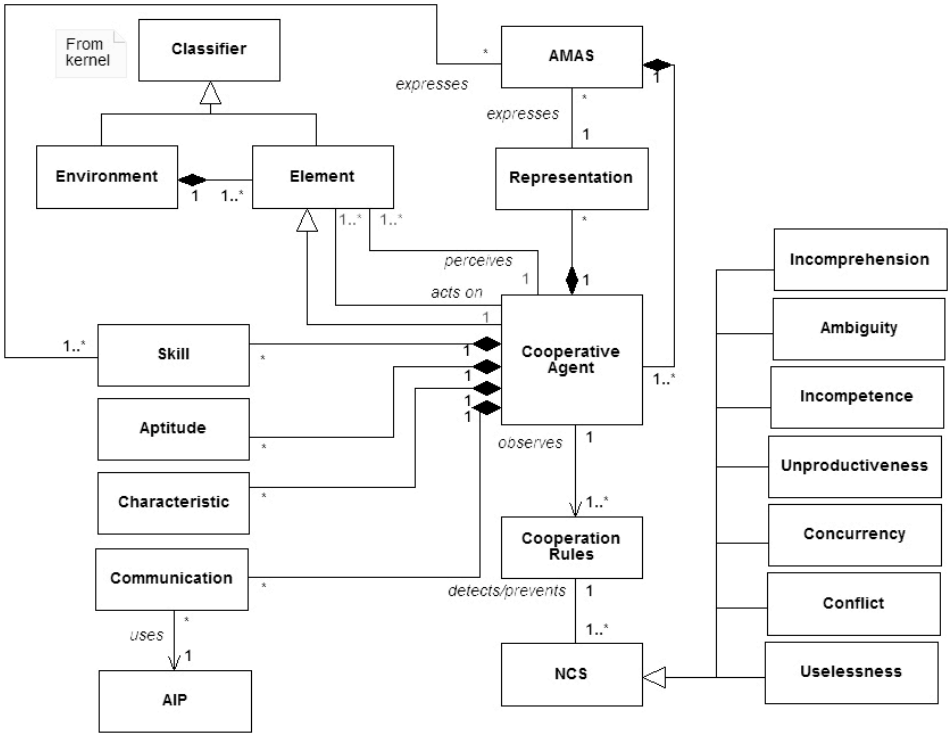

Figure 3-1. The MAS Meta-model adopted by ADELFE.

ADELFE [1][2] is a methodology devoted to the design of adaptive multi-agent systems (AMAS). According to this approach, building a system which realises 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 in having perceptions, taking decisions and then

doing actions (perceive-decide-act). Local cooperation rules enable the agent to detect and solve Non Cooperative Situations (NCS) that play a fundamental part in the ADELFE design. These NCS 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 [3] 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 NCS of different kinds (such as incomprehension and uselessness). The concept of environment is essential for MAS, 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 specialisation of the UML Classifier, more especially an agent can be also viewed as such a specialisation. A cooperative agent has interactions with its 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 multi-agent system.

On the left part of Figure 3-1, we find Skill, Aptitude and Characteristic; skills represent the specific knowledge that enable each agent to realise 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.

3.2.2 Gaia

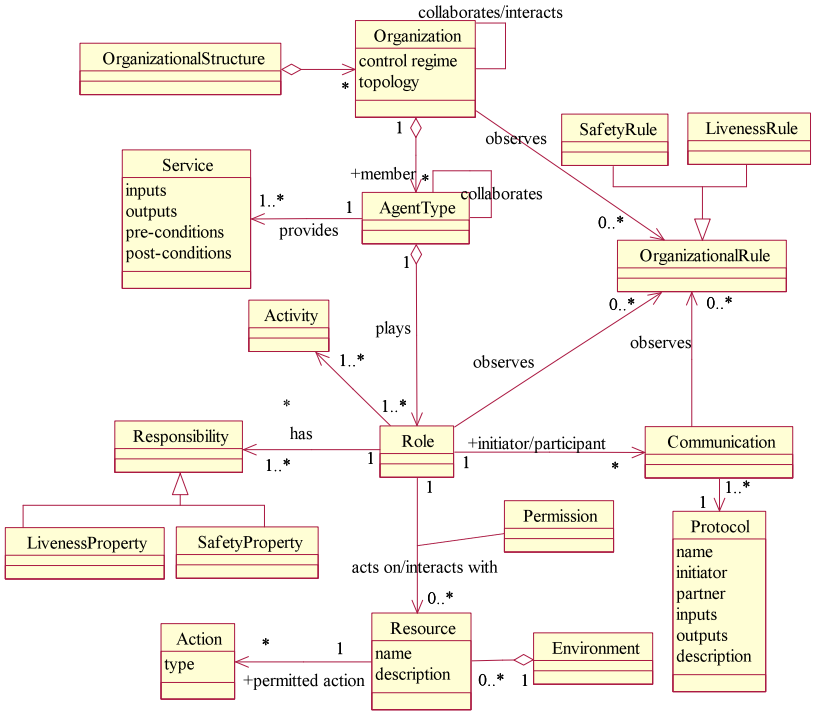

Figure 3-2. The MAS Meta-model adopted by Gaia.

The Gaia methodology evolved from an initial version [31], mainly focused on the design of handle small-scale, closed agent-based systems to the new one [32] 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 or organizational rules.

Having a deeper look at the MAS meta-model for the second, extended version of Gaia (Figure 3-2) [3] 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 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 Gaia MAS meta-model; it explicitly specifies all the entities and resources a multi-agent system 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 presented in [32], this is mainly achieved by introducing the organizational rules and the organization structure.

Organizational rules, specify some constraints that the organisation 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 organisation structure establishes the overall architecture of the system, that is the position of each role in the organisation 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.

3.2.3 INGENIAS

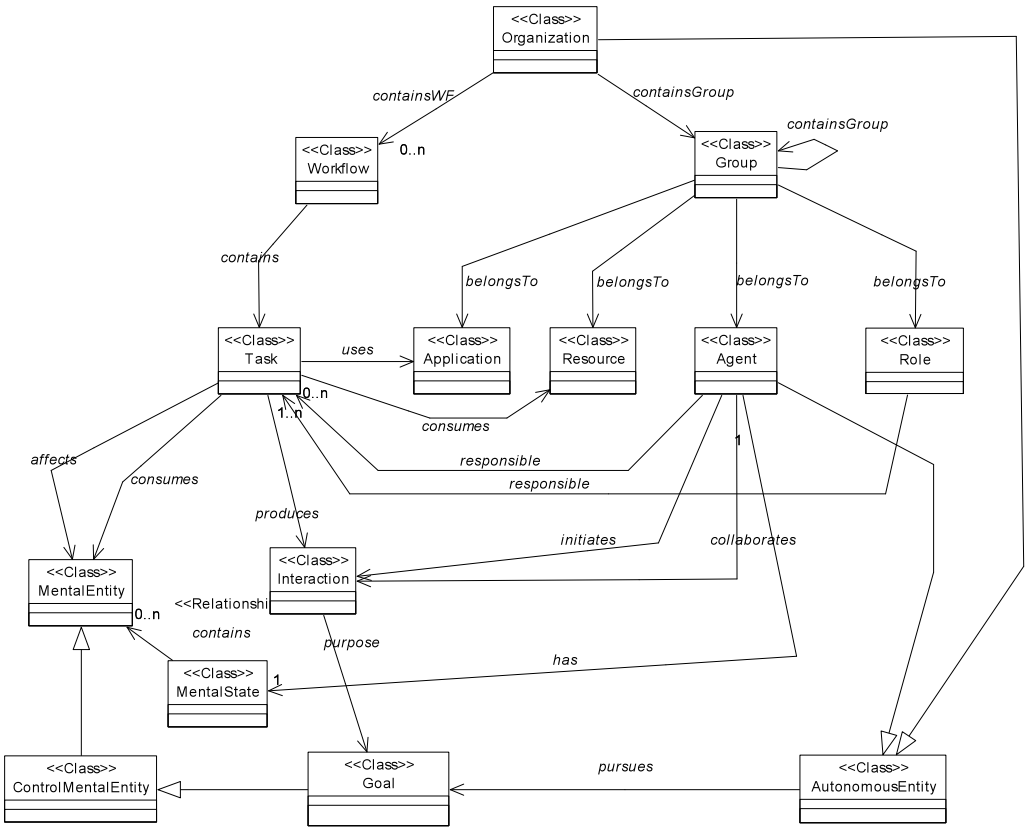

Figure 3-3. Summary of INGENIAS MAS meta-model.

INGENIAS [23] is both a methodology and a set of tools for development of multi-agent systems (MAS). As a methodology, it tries to integrate results from other proposals and considers the MAS from five complementary viewpoints: organization, agent, tasks/goals, interactions, and environment. It is supported by a set of tools for modelling (graphical editor), documentation and code generation (for different target platforms). INGENIAS adopts the definition of meta-models for MAS from MESSAGE [10], but it refines and extends them, and builds support tools for all stages of the development cycle, from requirements elicitation to implementation and testing.

INGENIAS considers the specification of a MAS from five viewpoints (Figure 3-3):

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 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 responsible (in terms of Roles or Agents).

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 should be an agent that starts an interaction.

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.

   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 used also to express which are the inputs and outputs of the tasks and what are their effects on the environment or an agent's mental state.

5. Environment viewpoint. Defines agent's perception in terms of existing elements of the system. It also identifies system resources and who is responsible of 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 required diagrams, but in 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 (IDK, available at http://ingenias.sourceforge.net), whose base is the INGENIAS meta-model specification.

3.2.4 PASSI

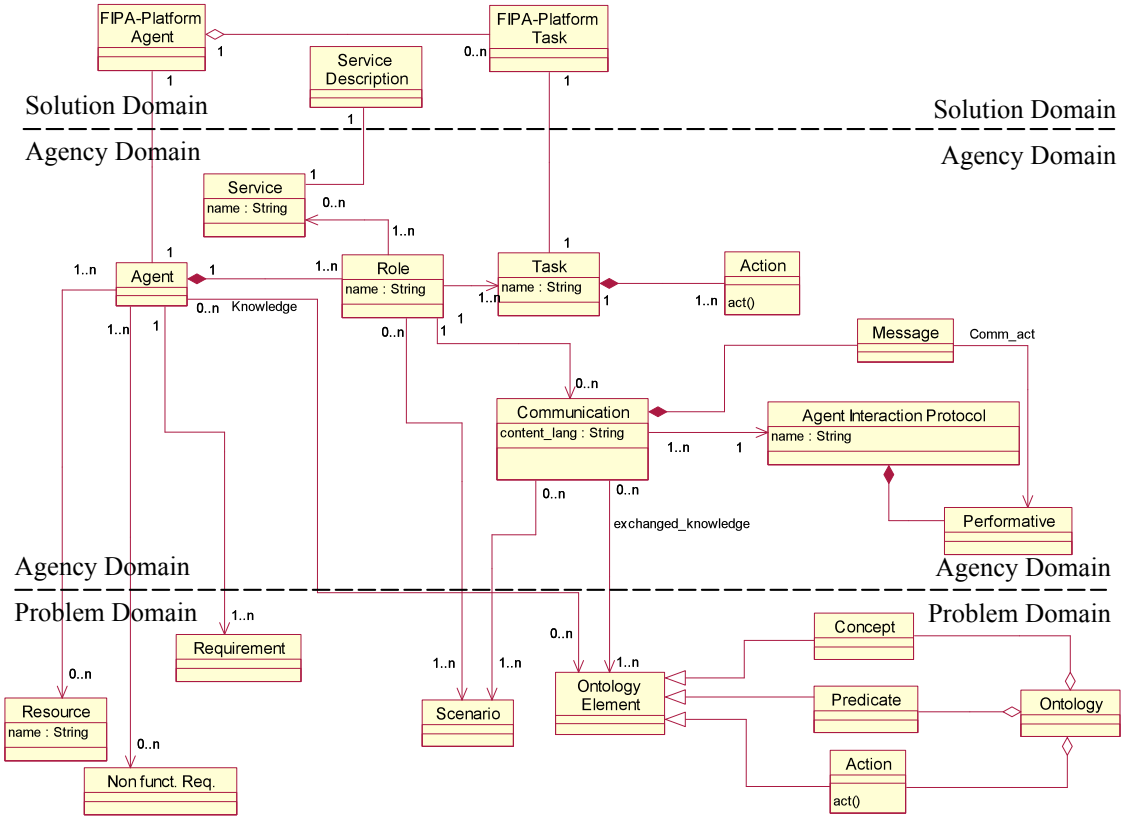

Figure 3-4. The MAS meta-model adopted by PASSI

PASSI (Process for Agent Societies Specification and Implementation) [14][9] is an iterative-incremental process for designing multi-agent systems starting from functional requirements that adopts 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 a specific attention for 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 [3] is organized in three different domains: the Problem Domain (where requirements are captured), the Agency Domain that 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 3-4) 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 operating 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 plays some roles; these are portions of the agent 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.

In PASSI, the term task is used with the significance of atomic part of the overall agent behaviour and, therefore, an agent can accomplishing 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 semantic of the message content [26]). 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).

3.2.5 RICA

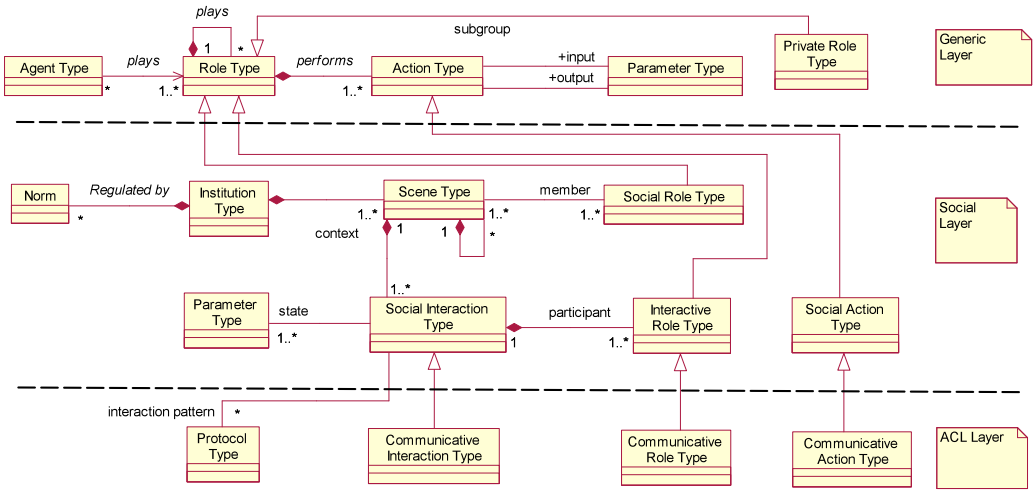

Figure 3-5. The MAS meta-model adopted by RICA

The RICA (Role/Interaction/Communicative Action) approach [27] integrates relevant aspects of Agent Communication Languages (ACL) and Organisational 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 organisational model of the agent society to the definition of the Agent Communication Languages of the multi-agent system.

The organisational 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.

RICA is essentially made of two major components: a meta-model (defining the language used to specify the organisational and communicative entities), and a specification of the structure and behaviour of these entities. The RICA MAS meta-model [19] [27] is organized in three different layers: the first one deals with the generic elements of

the system (agent, role and action types); the second one includes social concepts like norms, and institutions; the last one is devoted to agents' interactions via communications.

More in details, in Figure 3-5, 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 inputs and outputs 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 regards 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.

3.2.6 Tropos

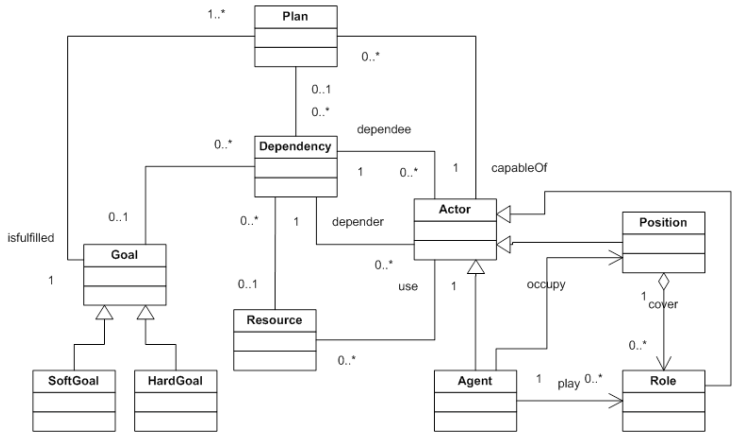

Figure 3-6. The MAS meta-model adopted by Tropos

Tropos [11][7] is an incremental design methodology that aims at modelling both the multi-agent system and its environment . The process covers the early phases of the analysis where it is characterised 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 details.

Differently from the other MAS meta-models, the Tropos one introduces the concept of actor seen as a generalisation of the agent [6] (Figure 3-6). 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. Tropos role is an abstract characterisation of the behaviour 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.

More in details, we can say that in Tropos, an actor [6] 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 other two 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).

3.2.7 A First Proposal of a Unifying MAS Meta-model

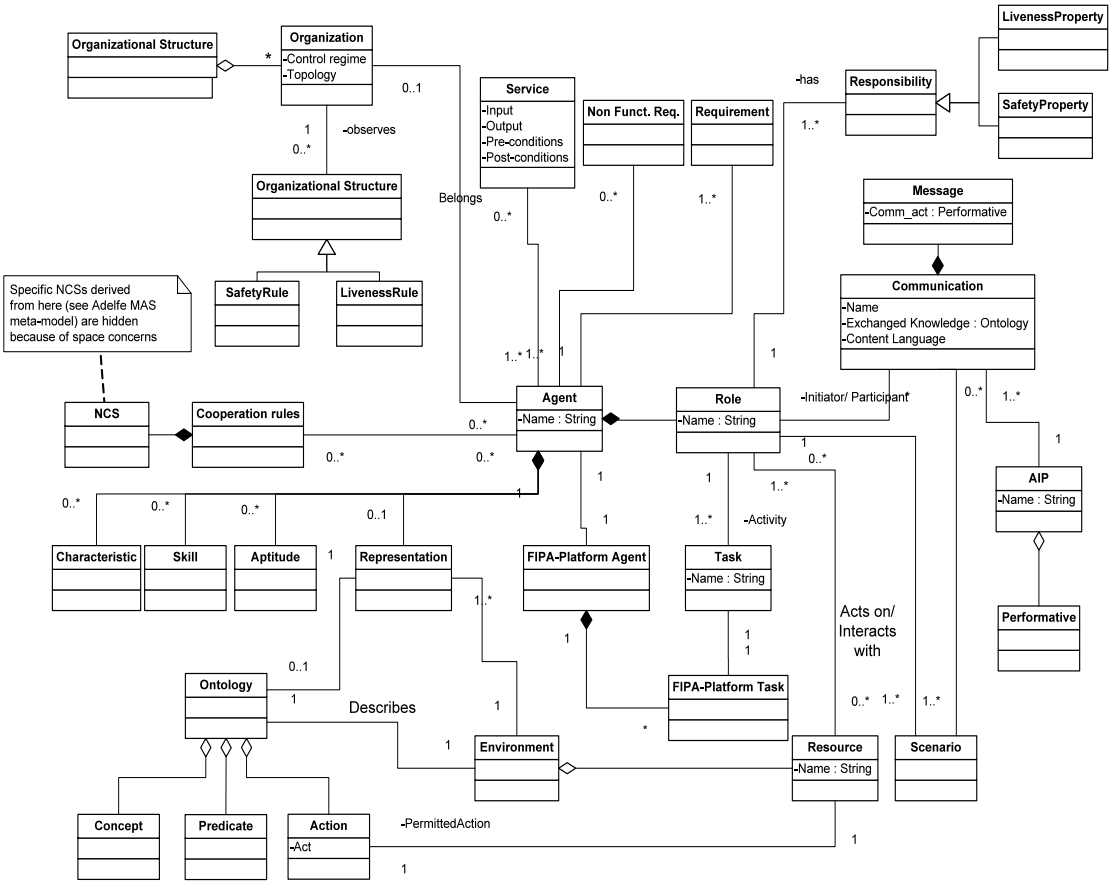

Figure 3-7. A first proposal of unifying MAS meta-model

Starting from the analysis of ADELFE, Gaia and PASSI MAS meta-models, in [3] there is a first proposal of a unifying MAS meta-model, which is presented in Figure 3-7. This metamodel is guided by the aim of creating societies without (ADELFE) or with predefined organizations, 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 relate by representing the fact that an agent has a representation (possibly affected by errors or uncertainty) of the environment expressed in terms of an ontological model of it.

3.2.8 A Comparison of MAS Meta-Models

The MAS meta-models presented in the previous subsections will now be compared with the precise aim of identifying their commonalities as a 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).

Meta-Model	Common components	Peculiarities
ADELFE	Agent (*cooperative*), (*almost*) FIPA communications structure	Environment, Cooperation rules, Non cooperative situations (NCS), 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, Group
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 [6])*	Goals, Resources, Dependency
Unifying MMM	Agent, Role, Task, FIPA communications structure	Service, Environment, Characteristic, Skill, Aptitude, Resource, Ontology

Table 3-1. A comparison of the discussed MAS meta-models from the structural point of view

With regards to this comparison it is worth to note 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) the authors of [3] 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: different MMMs differently approach the modelling of agents 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 slight different problem: we are aiming at identifying a unified MAS meta-model, 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 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 3-1 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:

- Agent: it is present in all the methodologies. Gaia and RICA refer to it as *Agent Type*.
- Role: all methodologies but ADELFE includes the Role component in their MMMs.
- Task: it is present in INGENIAS, PASSI, and Tropos (with a slightly different meaning: Plan [6])
- FIPA communications structure: with this we mean a collection of elements representing agents communications as based on messages and following some specific Agent Interaction Protocol (AIP). Several methodologies have a good support for this kind of interaction means: ADELFE, Gaia, INGENIAS, PASSI, and RICA.

From the analysis of the most characterising non-common elements of the studied MMMs we can see:

- Environment: it is present in ADELFE, INGENIAS, Gaia and the Unifying MMM. In ADELFE it is seen as a possible communication way 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 and its dynamics is described in terms of Workflows.
- Cooperation related elements: ADELFE has a deep structure about cooperation, it includes: Cooperation Rules and a hierarchy of Non Cooperative Situations (NCS); RICA has Social Roles and Actions; in INGENIAS organizations contain Workflows describing the collaborative work.
- Mental attitudes and states of agent: ADELFE agent is modelled in terms of Skills, Aptitudes and Characteristics; INGENIAS agent is intentional, and has a Mental State (Goals, Facts, Beliefs), a Mental State Manager and a Mental State Processor; 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 MAS meta-model 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 complements this unified MAS meta-model.

3.3 Towards a Unified MAS Meta-model: 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 MAS meta-model 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 MAS meta-model.

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:

- Existing methodologies for which authors are involved in this TFG are: ADELFE, Gaia, INGENIAS, PASSI, Tropos and RICA, and are presented above,
- The first reflection on modelling structures that the FIPA Modelling TC proposed [22],
- And an attempt for a unifying meta-model done in [3].

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

- In a 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 a second time, after having launched a discussion during the meeting, concepts such as environment, goal, organisation (society, group, institution, etc.), resource, service, would be discussed off-line,
- Other components such as ontology, dependency, action, mental states, organisation rules, norms, skills, aptitudes, characteristics (or similar BDI concepts) could be studied later on.

3.3.1 Identifying the Basic Components

3.3.1.1 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, what could lead to defining different kinds of agents, for instance cognitive or reactive ones. But other agents exist and they are neither cognitive nor reactive or may be considered as a mix of these two types. A double inheritance from "agent" in the MAS meta-model could solve the problem but since it was not mandatory to be exhaustive and define all kinds of agents, participants finally agreed on this minimal definition for an agent as an entity:

- which is capable of acting in its environment,

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

- which is autonomous: this means that an agent has control over its own behaviour based on internal or external stimuli,
- which can communicate with other agents,
- and which is 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 fact 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 specialisation of being capable of acting in an environment), having a definition for this latter concept and differentiate the cognitive nature of an agent from the reactive one was not considered as 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 an individual objective, or a satisfaction/survival function which it tries to optimise, or desire, or emotion,
- and which uses a representation of its environment.

3.3.1.2 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".

3.3.1.3 Defining "Task"

The discussion for defining the concept of "task" was based on 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 not observable tasks, being characterised by the first ones since they are visible; an agent could also do something that is not a part of playing a role.

3.3.1.4 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 great 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 way similar to the one Russell and Norvig did in their book [25], in which an environment can be accessible or not, dynamic or not, deterministic or not, continuous or not?
3. With regards 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 is really an environment and we decided to limit our work to the definition of an environment as "something that an agent can interact with and/or perceive". This definition goes in the same direction of 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.

3.3.1.5 Defining "Organisation"

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

Organisations 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 MAS for which ADELFE is specialised). The discussion concerned the nature of the link between "agent" and "organisation" because both an agent and an organisation can be seen as aggregations of roles. The definition for "organisation" was not completed during the meeting, it is just said as being composed of roles.

3.3.2 The Unified MAS Meta-model 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 a first draft of a unified MAS meta-model on which AOSE TFG participants agreed and which is shown in Figure 3-8.

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, that 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 the previous discussion on the definition of "organisation", an organisation may be composed of agents but also of roles depending on the concerned methodology. This is expressed on the MAS meta-model by following navigation links from "agent" to "organisation" via "role".

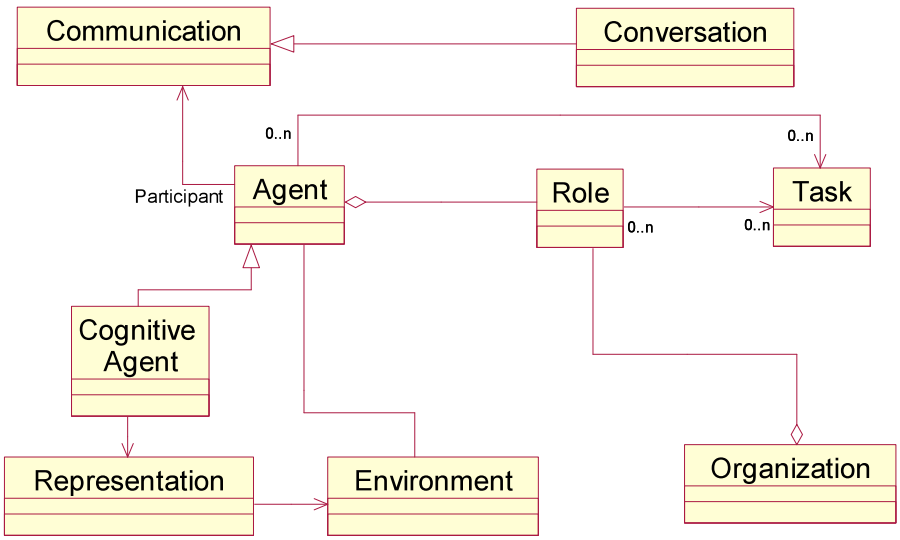

Figure 3-8. The AgentLink AOSE TFG MAS meta-model proposal

4 TFG 3 - Budapest Event

The third AOSE TFG meeting was aimed to complete open activities from the previous two meetings and was enriched by the participation of invited speakers from Australia and USA, who introduced new arguments in the discussion. The main topics of this meeting (apart of talks) were the refinement of the Ljubljana MAS meta-model and the evaluation of several AOSE methodologies through a questionnaire-based approach.

The meeting was organized in three sessions:

1. Talks about agent-related topics from outstanding researchers in the area.
2. Refinement of the Ljubljana MAS meta-model.
3. Methodologies evaluation: results of the questionnaires, comparison and debate.

There were several talks during the first day. In the first one, G. Fortino presented an agent-based approach for the management of distributed workflows. Then L. Padgham presented the Prometheus-ROADMAP methodology. This was followed with a description, by B. Henderson-Sellers, of the FAME project, which deals with a method engineering framework founded on a powertype-based metamodel. After that, J. Odell reported about his most recent industrial experiences on the use of AUML. A. Garcia presented his work on the introduction of aspects in MAS development and discussed both lessons learned and open issues. A. Molesini talked about the importance and role of artifacts in Agent-Oriented Software Engineering in general and more specifically in the SODA methodology. Finally, M. Cossentino reported about the experience in the FIPA Methodology Technical Committee where the method engineering paradigm has been adopted in order to converge towards an agent-oriented design methodology standardization proposal.

The second meeting day was divided into two sessions. In the first one, there has been a discussion about the introduction of new elements in the Ljubljana MAS meta-model. The work started from a proposal by P. Giorgini and M. Cossentino about the introduction of the *Goal* element. The outcome was a refinement in the definition of the *Task* element (initially introduced in the Ljubljana meeting and now defined as: “the activity (set of activities) motivated by some coherent reason”), a definition of Goal as “a specification of a state of the world that may be achieved or

maintained by the agent". An initial definition was drafted for Plan as follows: "A plan specifies how a goal can be achieved". We concluded this session with the agreement that the MAS meta-model is worth of further work and we plan to do it by using a mailing list.

The session about methodologies evaluation was characterized by a very active debate. The discussion mainly was about a comparison presented by Juan Pavón of seven different methodologies on the basis of a questionnaire (that was made available through the AOSE TFG website). The questionnaire was adapted from a paper of Khanh Hoa Dam and Michael Winikoff presented at the AOIS'03 workshop held in Melbourne [34]. The main comments that should be reported from this discussion regard the impact on the evaluation results of the relationship of people who compiled the questionnaire with the methodology they evaluated. Judgements from users of methodologies were often more critical about them (as it was expected); as a consequence, we agreed that the lack of user-filled questionnaire for some methodologies partially limited the goodness of the presented results, but the experience was useful to improve the questionnaire in future editions. This kind of questionnaire will be one of the main tools for an AOSE evaluation framework, which would also include case studies and specific metrics. We discussed how to contribute towards the definition and application of such an evaluation framework.

4.1 Talks on Agent-Oriented Software Engineering Issues

The session has been opened by G. Fortino who presented an agent-based approach for the management of distributed workflows. In this work, he proposed an approach covering modelling and enactment of WFMSs (Workflow Management System) so exploiting the enormous potential of agents for supporting the development of complex systems from their modelling to their implementation. The modelling phase is enabled by a MAS which can be instantiated by a workflow schema based on the Workflow Patterns already presented in literature. The enactment phase is supported by an agent-based framework designed on the basis of the MAS meta-model so allowing a seamless transition from modelling to enactment of distributed workflows. During the talk the structural and behavioural MAS meta-models for both flat and hierarchical workflow management systems were presented. The implementation of this framework is in progress using the JADE development platform.

Lin Padgham presented a methodological approach for development of MAS by integrating the Prometheus and ROADMAP methodologies. Prometheus is an experimented methodology that addresses several phases of the development lifecycle, especially system specification (by identification of requirements and the analysis of system goals), architectural design (by identification of agents and interactions as conversation protocols, and providing a system overview) and detailed design (by definition of process diagrams, agent capabilities, and plan types, close to implementation). ROADMAP is more abstract and high level than Prometheus, with more focus in requirements analysis. It mainly considers the definition of models for goals, roles, agents, and interactions. This can integrate with Prometheus, although a common notation is still under study. A new version of the Prometheus Design Tool (PDT) has been recently released in September, and plans exist to integrate it with Eclipse.

Brian Henderson-Sellers presented the FAME project, which intends to create an agent-oriented, method-engineering based approach to software development. This considers both process and product aspects, including an agent-oriented modelling language, called FAML. As a method-engineering based approach, the project intends to incorporate fragments of several methodologies. It starts with the OPF repository, whose fragments are being translated to be SEMDM-compatible, and is evaluating Tropos, Prometheus, MaSE, Gaia, Cassiopeia, MAS-CommonKADS, AgentFactory, CAMLE and PASSI for new method fragments. As an example, it has been shown how to enhance

Prometheus by Tropos. Concerning the activity of AOSE TFG on MAS meta-models, the FAML proposal should be considered as it structures design and runtime concerns for specifying agent external and internal issues.

James Odell presented recent industrial experiences in the use of AUML as a modelling language. Basically, agent-related concepts appear smoothly in existing software modelling practices, and provide some added value, both from the point of view of the customer (for instance, goals in requirements analysis are explicit, which can be good for managers to understand the software process) and the developer (for instance, as a way to specify and understand high-level processes).

Alessandro Garcia discussed how Aspect-Oriented Software Development (AOSD) can be useful when developing MAS. This has been shown for learning aspects, which could be separated from other agent concerns. And this can be applied to other aspects, such as agent mobility or collaboration among agents. One of the main issues to work on is the definition and implementation of more powerful composition rules that facilitate the combination of different aspects. An interesting conclusion is that the study of aspects can improve the understanding of AOSE abstractions (autonomy, roles, coordination, mobility, etc.) and provide a common framework for the research of effective composition rules and mechanisms in MAS.

Ambra Molesini presented some considerations about the role of artefacts in Agent-Oriented Software Engineering; with the term artefact, A. Molesini refers to objects or tools that agents share and use to support their activities and achieve their objectives; an artefact exposes usage interface, operating instructions and a functionality description. An application of the effect that artefact design has on methodologies has been presented by extending the SODA design process. SODA concentrates on inter-agent issues, like the engineering of societies and infrastructure and one of its three models deals with environment (the other two models are agent and society) that can be considered the base class of all artefacts from an ontological point of view. Several artefact kinds have been introduced in SODA: the Individual artefact in the Agent model, social rules and social artefacts in the Society model and finally resource and social artefacts in the Environment model. The study is deepened by the application of zooming among different abstraction layers of artefacts.

M. Cossentino in his talk reported the experience of the FIPA Technical Committee (TC) Methodology. While FIPA is moving in the IEEE organization, all of its committees are being restructured and therefore it is the right time for a debate about the possibilities of standardization in AOSE and specifically in design methodologies. The approach followed by this group was based on the consideration that it makes no sense to define one unique methodology and to force everyone to adopt it in order to be “FIPA compliant”. The proposed solution is known in classical software engineering as (situational) method engineering. According to this approach, the development methodology is built by assembling pieces of existing processes (called method fragments) from a repository of methods. In this way each designer can obtain the best process for his/hers specific needs. The TC Methodology during its activity produced a definition of method fragment and a collection of fragments extracted from several well-known methodologies. Experiments of methodologies composition using this approach are ongoing and are already published in papers from members of the TC.

Each talk has been followed by an active debate that involved all the participants in a very stimulating experience exchange.

4.2 MAS Meta-models (MMM)

At the beginning of this discussion, a debate on the scope of the MAS meta-model showed that in the current state-of-the-art it is still difficult to find a general agreement on the use of agent-related concepts. In fact, this was already

remarked in the first AOSE TFG meeting, and for this reason we considered relevant the work on this task. During the second AOSE TFG meeting held in Ljubljana there was some agreement on a reduced meta-model and a set of basic definitions of main entities. Here the purpose was to propose extensions and refinement to that meta-model and review the definitions.

The work started from a proposal by P. Giorgini and M. Cossentino about the introduction of the *Goal* element. The outcome was a refinement in the definition of the *Task* element (initially introduced in the Ljubljana meeting and now defined as: "the activity (set of activities) motivated by some coherent reason"), a definition of Goal as "a specification of a state of the world that may be achieved or maintained by the agent". An initial definition was drafted for Plan as follows: "A plan specifies how a goal can be achieved". The complete meta-model as resulting from the initial Ljubljana proposal and following refinement is reported in Figure 4-1 (definitions of its elements in Table 1).

At the end of this session, which was finally quite short taking into account its purpose and the diversity of points of view from participants, there was agreement on the consideration that the MAS meta-model is worth of further work. It is expected to continue this by using a mailing list, moderated by Massimo Cossentino, with interested people who should contact him to be included in the list.

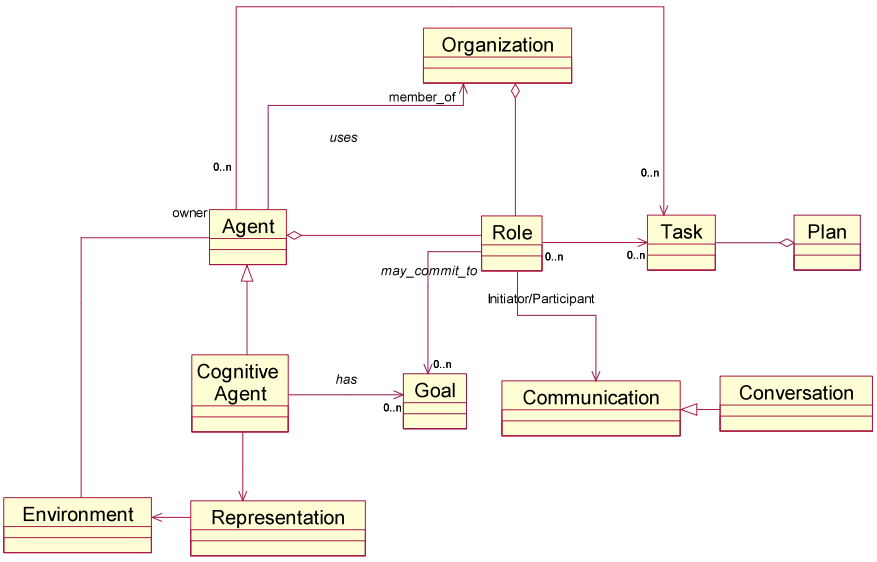

Figure 4-1. The MAS meta-model defined by the AL3-AOSE TFG as an improvement to the result of the previous meeting

Element	Definition	Example
Agent	An agent is an entity: - which is capable of acting in its environment - which is autonomous: this means that an agents has control over its own behaviour based on its internal or external stimuli - which can communicate with other agents - which is capable of perceiving its environment	
Cognitive Agent	A cognitive agent is an agent: - which is proactive: this means an agent behaviour is driven by a set of tendencies (in the form of individual objective, or a satisfaction/survival function which it tries to optimise, or desire, or emotion, …) - which uses a representation of its environment	
Role	An abstraction of a portion of the social behaviour of an agent. It can be characterized by a goal, accomplishing some functionalities/providing a service.	
Task	A task specifies the activity (set of activities) motivated by some coherent reason	
Environment	Environment is something an agent can interact with and/or perceive	
Organization	It is composed of roles[3]	
Goal	A goal is a specification of a state of the world that may be achieved or maintained	A goal is a situation, but not all situations are goals. A set of states of the world is generally not a goal unless there is an agent committed to achieve/maintain this set of states
Plan	A plan specifies how a goal can be achieved[3]	

Table 1. Definitions of the elements composing the MAS meta-model of Figure 4-1.

4.3 Methodologies Evaluation

This session was organized around the review of the results of the methodology evaluation questionnaire prepared and distributed by Massimo Cossentino, as an extension of a proposal by Khanh Hoa Dam and Michael Winikoff presented at the AOIS'03 workshop held in Melbourne [34]. This review should serve to define a framework for evaluation of agent-oriented methodologies. Such a framework would be useful in several respects, to clarify AOSE concepts, to identify and integrate fragments, and to promote AOSE support tools.

The questionnaire assesses an AOSE methodology against a range of criteria concerning concepts and properties, modelling, process, and pragmatics. Several methodologies were evaluated: ADELFE, Gaia, INGENIAS, OPF, PASSI, Prometheus, and Tropos. From the experience in the filling and comparison of questionnaires we can see that it fulfils most of the issues to be considered when evaluating a methodology. However, some degree of subjectivity in many questions and some divergence in the understanding of some, motivate that the questionnaire should be refined, and probably slightly re-structured. Also, more objective metrics should be defined (although this is a tough issue). Additionally, some standard case studies should be useful to compare the benefits of using different methodology fragments in the development. With this purpose, the interest of having a web site to continue this activity was considered.

Results of questionnaires are compared in tables from Table 3-1 to Table 4-7.

[3] To be further refined

Concept/ Property	ADELFE	Gaia	INGENIAS	OPF	PASSI	Prometheus	Tropos
Autonomy	H	H	H	H	H/H/M	H	L
Mental attitudes	L	N	H	H	L/L/M	M	M
Proactiveness	M	L	H	H	H/M/H	H	N
Reactiveness	H	L	H	H	H/H/H	H	N
Concurrency	H	M	H	L	H/H/M	H	L
Teamwork /roles	L	H	H	H	M/H/H	L	M
Cooperation model	AMAS th.	Teamwork	ALL	ALL	Task del./ Teamwork	none	Negotiation/ Task del.
Protocols support	H	H	H	H	H/M/H	H	N
Communication modes	ALL	Async mess.	ALL	ALL	Direct	N	
Communication language	ALL	ACL like	ALL	ALL	Speech acts	messages	
Situatedness	H	H	H	H	H/M/M	H	H
Environment type	All episodic	Dynamic Conti-nuous	All discrete	ALL	ALL	ALL	Inacc., Non episodic, Dynam.
Concept/Property	Adelfe	Gaia	Ingenias	OPF	PASSI	Prometheus	TROPOS
Other agent features	—	Opennes	Opennes	Opennes	Mobility, openness, security	Plans, agent decisions	Security, Trust, Delegation, Ownership, Dependency, Provision
Non supported features	—	—	Security & Mobility (not explicitly)	Security & Mobility (on going work)	Complex design-time social organiza-tions	Security & Mobility	Dynamic Behavior of Agent
Clear concepts	A	A	SA	SA	SA/N/N	A	A
Overloaded concepts	N	D	D	SD	D/D/N	D	N
More Agent-oriented than OO	A	SA	SA	both	SA/A/A	SA	SA
(Main) Supported agents	Cooperative		BDI (mainly)	BDI (mainly)	Mainly: State-based, rational, reactive	ALL	BDI, Rational
Society of agents modelling	No	SA	SA	(on going work)	A/-/-	No	A
Society structure	-	-	Groups/WF	-	p2p, simple hierarchies, holons	-	Agent Society Pattern, such as Broker, Mediated, Matchmaker

Table 4-1. Concepts/properties-based comparison of methodologies. Keys: N: None, L: Low, M: Medium, H: High, SD: Strongly Disagree, D: Disagree, N: Neutral, A: Agree, SA: Strongly Agree. For PASSI: Creator/PhD Students/Grad. Stud.

Notation	ADELFE	Gaia	INGENIAS	OPF	PASSI	Prometheus	Tropos
Support for static (structure) and dynamic (processing) aspects	SA	A	SA	—	SA/A/A	SA	D
Symbols and syntax well defined	A	N	SA	—	A/A/D	A	N
Well defined semantics	A	D	SA	—	A/N/D	A	A
Clear notation	A	A	N	—	A/A/N	A	N
Easy to use notation	A	A	SA	—	A/A/N	A	SA
Easy to learn notation	N	SA	A	—	N/N/N	NA	N

Table 4-2. Notation-based comparison of methodologies. Keys: SD: Strongly Disagree, D: Disagree, N:Neutral, A:Agree, SA: Strongly Agree, NA: Not Applicable. For PASSI: Creator/PhD Students/Grad. Stud.

Modelling	ADELFE	Gaia	INGENIAS	OPF	PASSI	Prometheus	Tropos
Multiple views	A	N	SA	SA	SA/A/A	—	A
Adequate and expressive	A	N	SA	SA	A/A/N	—	N
Traceability between models and between models and code	A	D	SA	SA	SA/N/A	—	D
Guidelines and techniques for consistency checking	A	SD	N	N	N/N/N	—	D
Supports refinement	SA	N	SA	N	SA/A/A	—	A
Supports modularity	SA	D	A	SA	SA/A/A	—	N
Supports component reusability	SA	SD	SA	SA	SA/A/A	—	SD
Extensible	SA	SD	SA	SA	A/-/-	SA	A
Supports hierarchical modelling and abstraction	SA	D	SA	SA	SA/N/A	SA	A
Other issues							

Table 4-3. Modelling aspects-based comparison of methodologies. Keys: SD: Strongly Disagree, D: Disagree, N:Neutral, A:Agree, SA: Strongly Agree, NA: Not Applicable. For PASSI: Creator/PhD Students/Grad. Stud.

Lifecycle coverage	ADELFE	Gaia	INGENIAS	OPF	PASSI	Prometheus	Tropos
Planning	CE			CEH		CEH	
Requirements analysis	CE	CE	CEH	CEH	CEH/-/-	CEH	CE
Architectural (or agent society) design	CE	CE	CEH	CEH	CEH/-/-	CEH	CE
Detailed (agent) design	CE	CE	CEH	CEH	CEH/-/-	CEH	CE
Implementation			CEH	CEH	CEH/-/-	E	P
Testing/Debugging	H		P	CEH	H	PCEH	
Deployment			P	CEH	CE		
Maintenance				CEH		P	
Death				CEH			

Table 4-4. Lifecycle coverage-based comparison of methodologies. Keys: C: Clear definition of activities, E: Examples given, H: Heuristics given, P: Partial. For PASSI: Creator/PhD Students/Grad. Stud.

Process	ADELFE	Gaia	INGENIAS	OPF	PASSI	Prometheus	Tropos
Addresses Quality Assurance	D	SD	N	SA	N/N/A	A	SD
Estimating guidelines (cost, …)	—	SD	A	N	D/N/A	N	N
Support for decision making (e.g. when to move between phases)	A	SD	SA	A	N/A/A	N	D
Development approach	Iterative/ incremental	Top-down	Iterative/ incremental Transformation & architectural based	ANY	Iterative/ Incre-mental	Iterative/ Incre-mental/ Spiral	Top Down
Supports patterns or reusability	A	SD	D	SA	SA/-/-	N	N
Degree of user implication (i.e. it does requires user-designer communication ?)	Medium	—	Medium	Strong	Weak	—	M

Table 4-5. Process-based comparison of methodologies. Keys: N: None, L: Low, M: Medium, H: High, SD: Strongly Disagree, D: Disagree, N: Neutral, A: Agree, SA: Strongly Agree. For PASSI: Creator/PhD Students/Grad. Stud.

Software tools	ADELFE	Gaia	INGENIAS	OPF	PASSI	Prometheus	Tropos
Diagram editor	OpenTool		IDK editor		PTK		GR-Tool, ST-Tool, TAOM4E
Code generator			IDK code g.		Agent Factory		
Design consistency checker			IDK ATA	Prototype	PTK		GR-Tool, ST-Tool
Project Management	AdelfeToolkit						
Rapid prototyping							
Reverse engineering					Agent Factory		
Automatic testing							
Commercial or research product	OT: comm. AT: free		Research	Research	Research	Research	Research
Adequate level of functionalities	A		A		A/-/-	A	N
Quick and easy to learn	N		A		N/-/-	A	A
Support in raising the quality	A		A		SA/-/-	SA	N
Reduces time to design/implem.	A		SA		SA/-/-	SA	A
Other comments			GPL license, UML/ Ingenias notation	Considering other tools			

Table 4-6. Software tools adopted in the considered methodologies.

Pragmatics	ADELFE	Gaia	INGENIAS	OPF	PASSI	Prometheus	Tropos
Audience	All	All	All	All	All/-/-	All	Grad. st., experts, researcher
Complexity compared to UML/RUP	About the same	About the same	About the same	A lot simpler	About the same/-/-	About the same	Simpler
Resources: Papers	X	X	X	X	X		X
Text books			X	X	X		
Tutorial notes	X		X		X		X
Consulting services			X	X			
Training services			X	X			
Nr.applications built with meth.	1-5	21+	6-20	21+ in OO/ME	21+/-/-	21+	1-5
Were applications real?	Yes	No	No	All	Y/Y/N	Yes	Yes
Any developed by other users?	No	Yes	Yes	Yes	Y/Y/N	Yes	No
Target any specific domain	Complex systems	No	No	All but RT	No/-/-	No	No
Support scalability	Yes	No	Yes	Yes	A/N/-	—	N
Supports distributed systems	Yes	—	Yes	Yes	SA/A/SA	—	N

Table 4-7. Pragmatics-based comparison of methodologies.

5 Conclusions

As far as AOSE has matured in the last years, there is a need for consolidating concepts, modelling languages and methods for agent-oriented development. In the AgentLink AOSE TFG we had a look at major aspects of this discipline by hosting several talks and interesting debates. Main effort has been dedicated to the study of MAS meta-models in order to find an agreement on basic concepts in the AOSE community and to facilitate integration of several agent-oriented models and methods. This is a work in progress and the proposed unified MAS meta-model and concepts definition will be further extended by the TFG members using a mailing list.

Another important aspect is the evaluation of methodologies. During the first meeting there was a review of existing works towards this aim and for the last meeting, several methodologies have been compared on the basis of a questionnaire, which has been filled by many participants of the AOSE TFG. Results are partial and affected by the reduced number of independent users but we think that this first attempt is nonetheless an important step towards methodologies assessment.

The three meetings have been a great success in terms of number of attending people, of participation and technical level but most of all, we think that the AOSE TFG has decisively contributed to create new research collaborations and to strengthen our community.

Acknowledgments

We would like to thank all the AOSE TFG meetings participants, the AgentLink III chairs and staff for their great organization support; we also would like to thank the other TFG chairs for the coordination they did of their groups activities thus enabling several joint meetings, activities and discussions.

References

[1] Bergenti F., Gleizes M-P., and Zambonelli F., Methodologies and Software Engineering for Agent Systems, Editions Kluwer to appear in 2004.

[2] Bernon C., Camps V., Gleizes M-P., and Picard G., Tools for Self-Organizing Applications Engineering, In Di Marzo Serugendo G., Karageorgos A., Rana O.F., Zambonelli F., eds. First International Workshop on Engineering Self-Organising Applications (ESOA), Melbourne, Australia, July 2003, LNCS 2977, Springer Verlag publ., 2004.

[3] Bernon C., Cossentino M., Gleizes M-P., Turci P., and Zambonelli F., A Study of some Multi-agent Meta-models. In Agent-Oriented Software Engineering V: 5th International Workshop, AOSE 2004. Revised Selected Papers. Editors: James Odell, Paolo Giorgini, Jörg P. Müller. LNCS Series, pp62-77, Vol. 3382 / 2005. Jan. 2005.

[4] Bernon C., Gleizes M. P. Adelfe MAS meta-model. Agentlink III – AOSE TFG2, Ljubljana (Slovenia). 28 Feb-2 Mar 2005. Online at: http://www.pa.icar.cnr.it/~cossentino/ al3tf2/docs/adelfe_ppt.pdf

[5] Bertolini D., Perini A., Susi A. and Mouratidis H. TROPOS meta-model. Agentlink III – AOSE TFG2, Ljubljana (Slovenia). 28 Feb-2 Mar 2005. http://www.pa.icar.cnr.it/~cossentino/al3tf2/docs/tropos_bertolini.ppt

[6] Bertolini D., Perini A., Susi A., and Mouratidis H., The Tropos Visual Language. A MOF 1.4 Compliant Meta-model. Agentlink III AOSE TFG 2. Ljubljana (Slovenia). 28 Feb-2 Mar 2005. Available online at: http://www.pa.icar.cnr.it/cossentino/al3tfg2/docs/tropos.pdf

[7] Bresciani P., Giorgini P.,Giunchiglia F., Mylopoulos J., and Perini A., Tropos: An Agent-Oriented Software Development Methodology. Journal of Autonomous Agent and Multi-Agent Systems, 8(3):203 – 236, May 2004.

[8] Brinkkemper, S., Lyytinen, K., and Welke, R., Method Engineering: Principles of Method Construction and Tool Support. Proceedings of the IFIP TC8, WG8.1/8.2 Working Conference on Method Engineering on Method Engineering : Principles of Method Construction and Tool Support. S. Brinkkemper, K. Lyytinen and R. J. Welke Editors, Chapman & Hall, Ltd., Atlanta (Georgia, USA) 1996.

[9] Burrafato P., and Cossentino M., 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) - 27-28 May 2002, Toronto (Ontario, Canada) at CAiSE'02.

[10] Caire, G., Coulier, W., Garijo, F. Gomez, J., Pavon, J., Leal, F., Chainho, P., Kearney, P., Stark, J., Evans, R., and Massonet, P., Agent Oriented Analysis using MESSAGE/UML. In Wooldridge, M., Weiss, G., and Ciancarini. P. (Eds.): The Second International Workshop on Agent-Oriented Software Engineering (AOSE 2001). Lecture Notes in Computer Science, Vol. 2222. Springer-Verlag, Berlin Heidelberg New York (2002) 119-135.

[11] Castro J., Kolp M., and Mylopoulos J., A Requirements-Driven Development Methodology. In Proceedings of the 13th International Conference on Advanced Information Systems Engineering (CAiSE'01), Interlanken, Switzerland, June 2001.

[12] Cervenka R., Trencansky I., Calisti M., and Greenwood D., AML: Agent Modeling Language Toward Industry-Grade Agent-Based Modeling, In Agent-Oriented Software Engineering V: 5th International Workshop, AOSE 2004. Revised Selected Papers. Editors: James Odell, Paolo Giorgini, Jörg P. Müller. LNCS Series, pp31-46, Vol. 3382 / 2005. Jan. 2005.

[13] Chella A., Cossentino M., Sabatucci L., Seidita V. The PASSI and Agile PASSI MAS meta-models. Agentlink III – AOSE TFG2, Ljubljana (Slovenia). 28 Feb-2 Mar 2005. http://www.pa.icar.cnr.it/~cossentino/al3tf2/docs/passi_ppt.zip

[14] Cossentino M., From Requirements to Code with the PASSI Methodology. In Agent-Oriented Methodologies, B. Henderson-Sellers and P. Giorgini (Editors). Idea Group Inc., Hershey, PA, USA. 2005 (in printing)

[15] Gómez-Sanz Jorge J., Pavòn J. INGENIAS MAS meta-model. Agentlink III – AOSE TFG2, Ljubljana (Slovenia). 28 Feb-2 Mar 2005. http://www.pa.icar.cnr.it/ ~cossentino/al3tf2/docs/ingenias_slides.zip

[16] Gómez-Sanz, J. and Pavón, J., *Meta-modelling in Agent Oriented Software Engineering.* Proc. 8th Ibero-American Conference on AI, Iberamia 2002, Seville, Spain, November 12-15, 2002.. Springer-Verlag, LNCS 2527, pp. 606-615.

[17] Gómez-Sanz, J., and Pavón, J., Meta-modelling in Agent Oriented Software Engineering. Advances in Artificial Intelligence - IBERAMIA 2002. LNCS 2527. Springer-Verlag (2002): 606-615.

[18] Guessom Z. MAS Meta-Models and MDA. Agentlink III – AOSE TFG2, Ljubljana (Slovenia). 28 Feb-2 Mar 2005. http://www.pa.icar.cnr.it/~cossentino/ al3tf2/docs/zahia_slovenia.pdf

[19] J. Manuel Serrano, S. Ossowski, S. Saugar. The RICA metamodel: an organizational stance on agent communication languages. Agentlink III AOSE TFG 2. Ljubljana (Slovenia). 28 Feb-2 Mar 2005. Available online at: http://www.pa.icar.cnr.it/cossentino/al3tfg2/docs/rica_serrano.pdf

[20] Kusek M. and Jezic G., Modeling Agent Mobility with UML Sequence Diagram. Agentlink III – AOSE TFG2, Ljubljana (Slovenia). 28 Feb-2 Mar 2005. Online at: http://www.pa.icar.cnr.it/~cossentino/al3tf2/docs/kusek_ppt.ppt

[21] Luck M., McBurney P., Preist C. Agent Technology: Enabling Next Generation Computing. A Roadmap for Agent Based Computing. Online at http://www.agentlink.org/roadmap.

[22] Odell J., Norine M., and Levy R., A Metamodel for Agents, Groups and Roles. In Agent-Oriented Software Engineering V: 5th International Workshop, AOSE 2004. Revised Selected Papers. Editors: James Odell, Paolo Giorgini, Jörg P. Müller. LNCS Series, pp78-92, Vol. 3382 / 2005. Jan. 2005.

[23] Pavón, J., and Gómez-Sanz, J.J., 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 2003. Lecture Notes in Computer Science 2691, Springer Verlag (2003) 394-403.

[24] Ralyte, J., and Rolland, C., An approach for Method Reengineering. Proceedings of the 20th International Conference on Conceptual Modeling, ER2001, Yokohama, Japan, November 27-30 2001

[25] Russel, S. and Norvig P., Artificial Intelligence: A Modern Approach. Prentice Hall Series, 1995.

[26] Searle J.R., Speech Acts. Cambridge University Press, 1969.

[27] Serrano J. M., and Ossowski S., On the Impact of Agent Communication Languages on the Implementation of Agent Systems, Eighth International Workshop CIA 2004 on Cooperative Information Agents, September 27 - 29, 2004, Erfurt, Germany.

[28] Serrano J.M., Ossowski S. On the Impact of Agent Communication Languages on the Implementation of Agent Systems, Eighth International Workshop CIA 2004 on Cooperative Information Agents, September 27 - 29, 2004, Erfurt, Germany

[29] Serrano J.M., Ossowski S., Saugar S. The RICA meta-model. Agentlink III – AOSE TFG2, Ljubljana (Slovenia). 28 Feb-2 Mar 2005. http://www.pa.icar.cnr.it/ ~cossentino/al3tf2/docs/serrano_ppt.pdf

[30] Trencansky, I., Agent Modeling Language (AML). Agentlink III – AOSE TFG2, Ljubljana (Slovenia). 28 Feb-2 Mar 2005. Online at: http://www.pa.icar.cnr.it/~cossentino/al3tf2/docs/aml_trencansky.pdf

[31] Wooldridge M., Jennings N.R., and Kinny D., A Methodology for Agent-Oriented Analysis and Design, In Proceedings of the 3rd International Conference on Autonomous Agents (Agents 99), pp 69-76, Seattle, WA, May 1999.

[32] Zambonelli F. Jennings N., and Wooldridge M., Developing Multiagent Systems: the Gaia Methodology, ACM Transactions on Software Engineering and Methodology, 12(3):417-470, July 2003.

[33] Zambonelli F., Open Directions in AOSE. Agentlink III – AOSE TFG2, Ljubljana (Slovenia). 28 Feb-2 Mar 2005. Online at: http://www.pa.icar.cnr.it/~cossentino/ al3tf1/programme.html

[34] Khanh Hoa Dam, Michael Winikoff, Comparing Agent-Oriented Methodologies, in the proceedings of the Fifth International Bi-Conference Workshop on Agent-Oriented Information Systems held in Melbourne in July (at AAMAS03).