Towards filling the gap between AOSE methodologies and infrastructures: requirements and meta-model

Fabiano Dalpiaz*, Ambra Molesini†, Mariachiara Puviani‡ and Valeria Seidita§
*Dipartimento di Ingegneria e Scienza dell'Informazione
Università di Trento
Email: dalpiaz@disi.unitn.it
†Dipartimento di Elettronica, Informatica e Sistemistica
Università di Bologna
ambra.molesini@unibo.it
‡Dipartimento di Ingegneria dell'Informazione
Università di Modena e Reggio Emilia
mariachiara.puviani@unimore.it
§Computer Science and Artificial Intelligence Laboratory
Università degli Studi di Palermo
seidita@dinfo.unipa.it

***Abstract*—Many different methodologies have been proposed in Agent Oriented Software Engineering (AOSE) literature, and the concepts they rely on are different from those adopted when implementing the system. This conceptual gap often creates inconsistencies between specifications and implementation. We propose a metamodel-based approach that aims to bridge this gap, resulting in an integrated meta-model that merges the best aspects of four relevant AOSE methodologies (GAIA, Tropos, SODA and PASSI). The meta-model assembly followed a well-defined process: for each methodology to be integrated in the meta-model, we elicited the requirements, identified a set of process fragments, thoroughly compared the concepts belonging to the various fragments, and finally composed the meta-model.**

I. Introduction

The trend towards agent-oriented software engineering (AOSE) is motivated by the need for a new engineering paradigm to face the increasing complexity and openness of computational systems. Object-oriented software engineering is adequate for the development of a variety of systems, but it falls short when applied to the development of open complex systems. This class of systems introduces the need for a new computing paradigm based on distributed intelligent units – *agents* –, whose characteristics are intrinsically different from objects [1]. This paradigmatic shift involves both the conceptual and the technical levels of the development cycle, ranging from the requirements analysis to the implementation and the deployment over an infrastructure. The work we present here is in the context of the "Methodologies for the Engineering of complex Software systems: Agent-based approach" (MEnSA) project[1], which aims at filling the conceptual gap between AOSE methodologies and multi-agent systems (MAS) infrastructures.

This gap is well known: Molesini et al. [2] examined this problem and proposed a case study concerning the SODA methodology [3]. Integrating an AOSE methodology with a MAS infrastructure requires to compare and relate the concepts, to provide a set of methodological guidelines, and to introduce a set of new concepts acting as a glue to make the integration successful. This task is not trivial, and one of the main reasons that make it complex is the difference in perspectives of methodologies and infrastructures developers. AOSE methodologies follow a top-down approach starting from a real world problem and moving towards a solution (the architecture of a MAS); thus, the concepts and techniques developed are mainly suitable for the use at analysis and design phases. On the other hand, the developers of MAS infrastructures follow a bottom-up approach starting from already existing programming paradigms, often an object-oriented one, and build upon it to form higher level programming constructs that make the development of the agent-based software easier.

MEnSA's "filling the gap" objective requires a complex process, made up of several sub-tasks, whose common element is the usage of a meta-model approach, and will ultimately produce an integrated methodology. This paper is focused on the work we have done concerning the integration of a number of AOSE methodologies; the integration of infrastructures is ongoing, and it will be presented in future publications. The AOSE methodologies taken into consideration by MEnSA are GAIA [4], Tropos [5], SODA [2], and PASSI [6]: they mainly differ in the typical scenarios they are designed to support, and in the phases they better cover. For instance, Tropos exploits a well established technique for requirements analysis (goal

[1]http://www.mensa-project.org

modeling), SODA provides an exhaustive characterization of the environment, PASSI has an extensive coverage of the implementation phase, and GAIA is well suited for the modeling of organizational aspects.

Our approach is founded on the work done by Cossentino et al. [7], [8] and starts from the definition of a set of requirements for the meta-model we want to assembly. Then, we elicit a set of fragments fulfilling the identified requirements, define a semantic conceptual map to precisely relate concepts belonging to various methodologies, and finally compose the fragments into an integrated meta-model.

This paper is structured as follows: Section II discusses the requirements we identified to lead the assembly of the meta-model; Section III describes the selected fragments and presents the conceptual map to compare the methodologies; Section IV focuses on the meta-model, describing the current version of meta-model; Section V terminates the paper by proposing conclusions and future work.

II. Requirements and principles for assembling the meta-model

In order to obtain a good meta-model – and a good methodology – we followed a path similar to that adopted in the engineering of a (software) system and proposed in [7], [9]. After defining the requirements for our product (the meta-model), we identified the fragments that better contribute to the satisfaction of the requirements.

In this section we illustrate how the integrated meta-model was conceived (Section II-A), and describe the requirements that led to a new methodology (Section II-B).

A. Assembling a meta-model

A meta-model describes the structure of all the elements that should be designed when following a specific methodology. Relationships between elements have specific meanings, and they should reflect the phases in the methodology. Different methodologies are built according to specific design philosophies, and comparing their meta-models is not a trivial task: often, the described concepts and relationships share the name but have different semantics.

Previous experiences in meta-models creation (e.g., [8], [9]) made it clear that this activity is much more than the mere selection and composition of concepts from the existing meta-models. Different composition patterns can be encountered:

1) selected elements from existing meta-models present the same name but have different meanings. This is the most common and difficult situation to be faced; a deep analysis of the collected elements has to be done, possibly some new elements have to be introduced, some others have to be modified in order to fill the presented differences;
2) selected elements have the same meaning but different names (the opposite of the previous case): renaming some elements is necessary;
3) all the selected elements present totally disjoint names and definitions, requiring just a simple composition; this is the best situation we could encounter, though the most unusual.

Given the consistency and coherence problems enumerated above, an integrated meta-model normally needs to be completed by concepts and relations acting as glue, introduced to ensure the important features of the original methodologies are not lost. After a sufficient refinement of the meta-model, it is possible to start the new methodology definition by assembling a set of selected process fragments according to the chosen life-cycle. If the selected fragments do not completely cover all the life-cycle phases and the requirements, new fragments will be selected, modified (if needed) and added.

B. The methodology requirements

The definition of a new methodology has to start by specifying the requirements to be satisfied. For the construction of a new integrated methodology, we decided to start from the requirements, choose the more suitable fragments belonging to existing methodologies, and assemble them in a proper way. The evaluation of the resulting integrated methodology is the verification of the extent to which the requirements are satisfied.

Now we list and describe the requirements and sub-requirements for our integrated AOSE methodology:

1) **Fill the gap between design and implementation**:
   a) Transformational approach from requirements elicitation to design and implementation, which refines high-level abstractions into low-level more concrete entities.
   b) Support for traceability: the path from each requirement to the corresponding source code should be clear and easy to identify.
   c) Powerful abstractions during the design phase are needed to provide an appropriate design of the system; they should be close to the infrastructure-level abstractions, but attention should be paid to avoid too fine-grained designs.
2) **Good requirements elicitation and analysis**:
   a) Support for both functional and non-functional requirements.
   b) Support for both goal-oriented and functional-oriented analysis.
3) **Different abstractions in the different phases** should make the comprehension of the design process easier.
4) **Enabling an easy transition towards the new methodology** for designers who are expert with one or more of the input methodologies.
5) **Precise and compact modeling constructs for the concept of agency**:
   i Agent: the definition of what an agent is and what it is supposed to do during its lifetime.
   ii Agent's rationale: the rationale an agent follows to achieve its objectives, that is the general reasoning principles leading the agent's behavior.

iii Situated agents: the environment where agents live requires an explicit representation throughout the whole methodology.
iv Social agents: agent-to-agent and agent-to-environment interactions are essential to engineer a multi-agent system.

Considering these requirements we started the analysis of the four selected methodologies (Tropos, Gaia, SODA, and PASSI), and we discovered more specific and detailed requirements. These requirements are listed below:

1) **Transformational process**: this need comes from requirements 1 and 3. The model-driven engineering paradigm [10] will be therefore adopted. Transformations between the elements of different domains should be clearly defined. For example, the notion of agent exists in different development phases and methodologies with (slightly) different meanings. Following a transformational approach, we can define several types of agent (requirements agent, design agent, ...), and define the way a certain agent transforms (or refines) into another one.
2) **Layering**: the management of different abstraction levels simplifies the design (requirements 1 and 3). SODA supports layering by means of the zooming and projection mechanisms. Zooming makes it possible to pass from an abstract layer to another, while projection projects the entities of a layer into another [3].
3) **Goal-oriented analysis** should be performed **before functional-oriented analysis**. The latter should start from results of the former. The goal-oriented analysis stands as a basis for Tropos, where agents are defined in terms of the functional and non-functional goals they want to achieve. Functional-oriented analysis is then used by eliciting the tasks to be executed to achieve the goals.
4) **Interaction**:
   a) Agent interactions should support *semantic communications* for removing or minimizing the ambiguity of messages contents.
   b) An ontology should be used to model agent knowledge in order to provide a conceptual background to all the agents belonging to a MAS.
   c) Compliance with FIPA ACL (Agent Communication Language) [11] specifications at the communication level is necessary.
   d) Agent interactions with the environment should be explicitly modeled.
   e) Indirect interactions (e.g., blackboard-based) should be supported.
5) **Organizational rules** proved to be a useful approach for modeling some social aspects. Gaia and SODA are examples of methodologies based on (organizational) rules to constrain and direct the agents behaviors.
6) **Environment and topology modeling** can be done by adopting abstractions like SODA's *artifacts and workspaces* in order to explicitly distinguish between active entities (agents) and passive entities (artifacts), and for organizing the conceptual places – workspaces – structuring the environment.
7) **Non-functional requirements should be explicitly modeled** (requirement 2). Tropos is the first AOSE methodology supporting explicit modeling of non-functional requirements, through the concept of soft-goals.
8) **Agent plans** should be modeled but they should not constrain the agent architecture to a specific kind of agent. In other words, the methodology should provide an abstract representation of plans, which can be realized into several implementations.

III. Selected fragments and conceptual map

The starting point for the meta-model creation is the requirements we described in the previous section. Given this, the approach we used to devise a meta-model is the following.

- Firstly, we have derived a set of fragments (Section III-A) satisfying the requirements identified in Section II. These fragments represent the core of the meta-model, which should be analyzed and refined in order to provide a better integration of the fragments.
- Secondly, we built a glossary of terms relevant to the fragments identified in Section III-A. For space reasons the dictionary is not reported here and it can be found in [12].
- Then, based on the glossary, in Section III-B we defined a conceptual map of terms, identifying synonyms and similar terms, and pointing out existing conflicts.
- Finally, on the basis of all the previous work, we defined the first version of the meta-model in Section IV.

A. Selected fragments

Each of the studied methodologies has some strong points, and should give a significant contribution to formulate the final MEnSA methodology. Here we list the coarse-grained fragments we have initially chosen from each methodology, which we extracted from the FIPA TC repository of fragments [13]:

1) **Tropos**
   a) Early requirements phase:
      i) Organization description.
      ii) Analysis.
   b) Late requirements phase:
      i) System identification.
      ii) Environment description.
2) **Gaia**
   a) Analysis phase:
      i) System *roles* identification.
      ii) Role model elaboration.
   b) Design phase:
      i) Service model development.

3) **SODA**
   a) Architectural Design:
      i) MAS Organisational model.
   b) Detailed Design:
      i) MAS Interaction model.
   c) Environment model.
4) **PASSI**
   a) Agent Society:
      i) Domain Ontology design.
      ii) Communication Ontology description.
   b) Agent Implementation:
      i) Multi-agent system design.
   c) System Requirement:
      i) Agent Identification.

B. Linking methodologies: a conceptual map

In order to propose an integrated meta-model, we built a conceptual map for eliciting *synonyms* (or, at least, similar concepts), and *inter-level relations* between concepts used at different abstraction levels. The conceptual map has been built on the basis of the MEnSA glossary [12], which has provided a complete and accurate semantic matching schema connecting the fragments' abstractions.

The conceptual map is shown in Figure 1; we used different colors to depict concepts belonging to fragments coming from distinct meta-models. The concepts are tied by two types of graphical links that represent two different relationships:

- non-directed links represent **horizontal relations**, which relate two concepts that are "synonyms". Identifying concepts sharing the same definition is very unlikely, and the resulting integration would be loose and nearly useless. Therefore we decided to extend the equivalence relation to include those concepts having a similar definition and whose usage in practice is equivalent. Horizontal relations (h) are not transitive: $h(c1, c2) \wedge h(c1, c3) \nrightarrow h(c2, c3)$.
- directed links point out **vertical relations**, which create "inter-level" links (top-down) between concepts belonging to different abstraction levels. We define *abstraction* as the development phase a concept belong to. Since we use h-relations to express similarity (and not only sameness), $h(c1, c2) \wedge v(c1, c3) \nrightarrow v(c2, c3)$.

In order to explain the conceptual map, in Figure 1 we have organized the different relations among concepts in various labeled sets. So, the first diagram chunk (a) concerns non-agentive concepts, which are typically in the system-to-be together with the agents. The Tropos concept *Resource* is horizontally linked to *Function* and *Legacy System* in SODA: more precisely, the former SODA concept is almost equivalent to Tropos resource, whereas the latter is linked because a legacy system defines a set of resources that should be modeled.

Aggregation of agents is examined in (b): Gaia *Organization* is a very high-level view of a set of agents (analysis phase), which is vertically linked to the lower-level concept *Society* of SODA (detailed design). These concepts are useful to

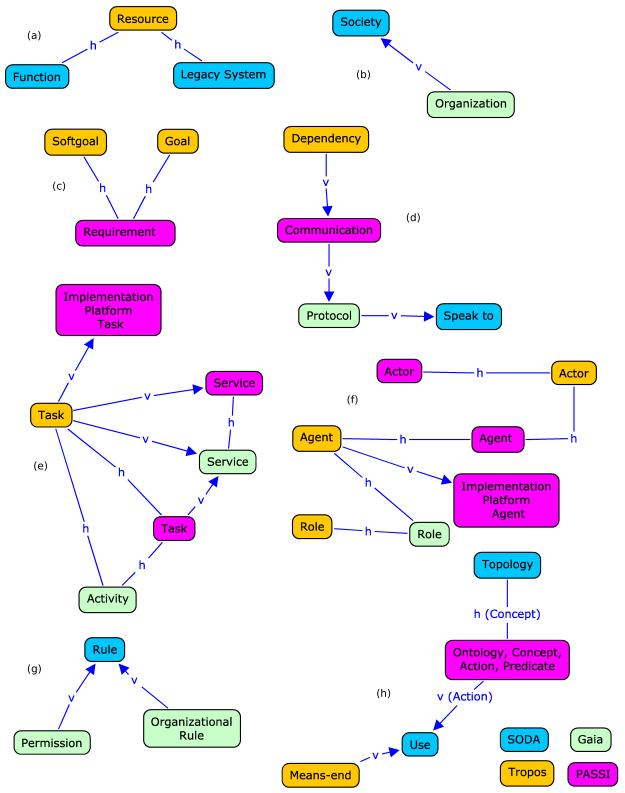

Fig. 1. Conceptual map linking concepts of different methodologies.

support the multi-level definition of the MEnSA meta-model, providing two related abstractions at different levels.

Requirements are considered in (c): Tropos *Goal* and *Softgoal* are horizontally linked to PASSI *Requirement*, the latter representing either a functional or a non-functional requirement.

Sociality of agents is represented in chunk (d). In this particular case, we have been able to point out a well defined hierarchical structure connecting the four examined methodologies. Tropos *Dependency* is used to depict linked actors justifying the reason why they depend on each other (for a goal, a task, or a resource); dependencies are defined during requirements analysis, and in our map they are vertically connected to PASSI *Communication*, which is a design-time concept defining an abstract interaction between two agents. We achieve a lower level of abstraction by linking communication to the Gaia *Protocol*, which defines the way in which roles interact with each other. Protocol is linked to an even lower level to the SODA *Speaks To*, which refers to the act of interaction between agents. It is worth noting that these concepts refer to different types of entity: a dependency involves actors, a communication is between agents, a protocol involves roles, a speaks-to is an atomic interaction act between agents. The meta-model and the derived methodology will handle this heterogeneity by defining exactly where these concepts apply.

Another interesting topic is that related to tasks and services

(e). Tropos *Task* is vertically connected to PASSI *Implementation Platform Task*, the latter being an implementation-level realization of the former (which stands at requirements level). Tropos *Task* is horizontally linked to PASSI *Task*. PASSI *Service* has a definition which is very similar to Gaia *Service*, and hence these two concepts are horizontally linked, providing an abstraction during the design phase. PASSI *Task* is vertically linked to Gaia *Service*. Gaia *Activity* is horizontally linked both to Tropos and PASSI *Task*. From these relations, we can derive a top-down relationship between task (or activity) and service, the former being higher level than the latter.

A crucial part of the conceptual map is the one related to agents, actors, and roles (f), because these are the active entities that glue all the other concepts together. Tropos *Agent* is horizontally linked to PASSI *Agent*, for they are both representations of the same concept. Another horizontal relation is between Tropos and PASSI *Actor*. Moreover, Tropos *Actor* is horizontally linked to PASSI *Agent*. These four concepts are not synonymous, but they are at the same level of abstraction. A third couple of horizontally linked concepts is Gaia and Tropos *Role*, with a further horizontal relation between Tropos *Agent* and Gaia *Role*. There is a vertical relation between Tropos agent and PASSI *Implementation Platform Agent*. This diagram chunk is not very precise: many relations have been identified, but during the meta-model and methodology definition we will have to make some choices and define the selected concepts in a more precise way.

Constraints are an important aspect in multi-agents systems, and our conceptual map contains some related entities and relations in (g). SODA *Rule* is a general design-time concept for regulating agents and their interaction with the environment they live in. Gaia *Permission* and *Organizational Rule* are analysis-level concepts which define the organization in terms of rules and permissions, and are linked vertically to SODA *Rule*.

The last part of the diagram (h) mainly involves the usage of entities by agents. SODA *Uses* is intended to depict a kind of interaction between an agent and an artifact, whereas Tropos *Means-end* is a higher-level abstraction of this behavior, where there is an intentional relation between a goal (or task) and a resource: the latter is the means to achieve the end (the goal or the task). PASSI *Action* is vertically linked to *Use*, whereas PASSI *Concept* is horizontally linked to SODA *Topology*, since a topology is defined in terms of the concepts constituting and regulating it.

IV. MEnSA META-MODEL: A PRELIMINARY VERSION

Starting from the MEnSA glossary and the conceptual map linking concepts of different methodologies, we have created a first version of the MEnSA meta-model. This initial effort is restricted to the phases of requirements and design; the layering is coarse grained, and implementation-level concepts are just sketched. The meta-model we present here slightly refines the initial version described in MEnSA deliverable 1.2 [12].

The key notions around which all the other elements are placed, are *role* and *agent*, which are building blocks for several AOSE methodologies. The meta-model is presented in Figure 2, and we describe it in the next two sub-sections, which refer to (1) requirements phase, and (2) design and implementation phases of the meta-model. The term "phase" is here used in order to represent the logical connection between the three main software process engineering phases and the meta-model elements a designer instantiates while developing each phase.

A. Requirements phase

The requirements phase is mainly based on the fragments of Tropos and Gaia, with some concepts coming from SODA (environment-related), and ontological aspects extracted from PASSI. The meaning of the presented concepts and relations derives from the corresponding methodologies, unless we specified otherwise in the description.

The main notion in the requirements phase in MEnSA meta-model is that of *Requirements Agent*, which is defined in terms of the concepts it connects to (through association links). A requirements agent *plays* one or more *Roles*, and *knows* an *Ontology*. The concept of Role is defined as Tropos' role plus Gaia's rules and permissions, whereas Ontology comes from PASSI. An *Organization* (Gaia) is composed of a set of agents (*made_of* relation between Organization and Requirements Agent), and *has* a set of *Organizational rule*s (Gaia) which define the regulations of the Organization. Every Role *adheres* to the Organizational rules of the Organization where the agent playing that Role lives.

The element Ontology is specified in a slightly different way from PASSI's definition, because we support here a refinement process of the ontology in different development phases. At requirements time, the Ontology is made of a set of *Requirements Concept*s, which can be hierarchically organized (self-transition). This is a coarse-grained representation of an ontology, which will be refined at design-time.

Each Role is *responsible* for one or more *Requirement*s (equivalent to Tropos' abstract goal), and each Requirement can belong to more than one Role. Requirement is specialized (concretized) into *Goal* (hard-goal in Tropos) and *NFR* (Non-Functional Requirement) (soft-goal in Tropos). A Goal can be *and/or-decomposed* into a set of sub-goals, *contribute* to Non-Functional Requirements, and can be *means-end* decomposed. The means to achieve a Goal can be either a *Resource* (in the Tropos sense, which corresponds to SODA Function) or an *Activity* (in Gaia glossary, but we showed this it is horizontally linked to Tropos and PASSI task). Activities, like Goals, contribute to the satisfaction of Non-Functional Requirements, and can be refined through *and/or decomposition*. Resources can be viewed as means to carry out an Activity, in the same manner they are used to achieve Goals.

Another important concept in the requirements phase is that of *Dependency*: this notion is taken from Tropos, and connects a *depender* role to a *dependee* role *for* a certain *Dependum* (the object of the dependency). A Dependum can be either a

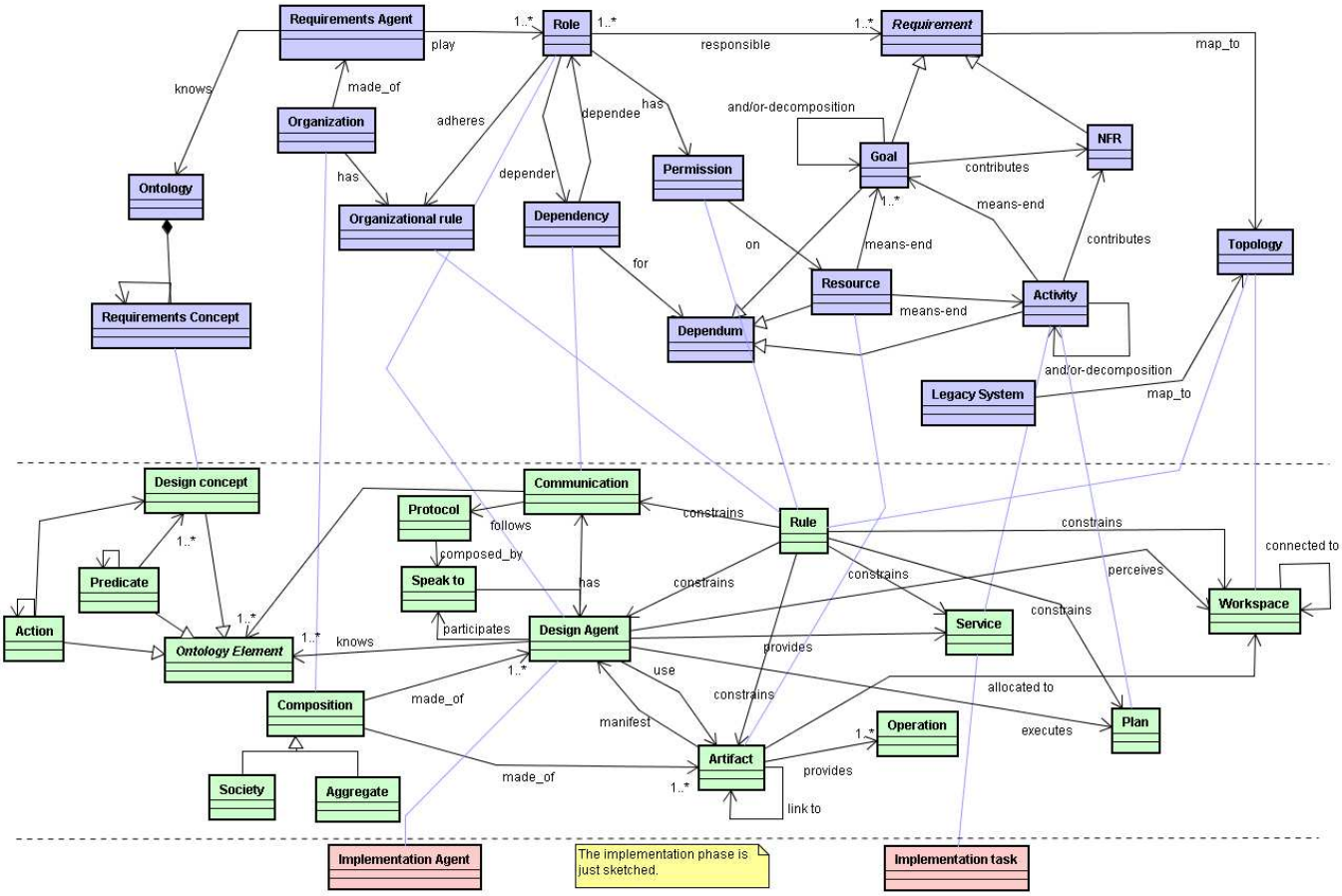

Fig. 2. First version of the MEnSA meta-model.

Goal (the depender wants the dependee to fulfill that Goal), an Activity (the depender wants the dependee to execute an activity), or a Resource (the depender wants the dependee to provide a Resource).

A Role is defined also by expressing which *Permission*s it *has on* certain Resources. This enables the definition of constraints on the usage of/access to resources.

A Requirement has a relation *map_to* with SODA *Topology*, because the achievement of that requirement depends on the topology definition. The concept of *Legacy System* (SODA) *map_to* a topology, as well.

B. Design and implementation phases

In these phases an additional modeling construct is used to define elements, that is the *realization* links between concepts at different abstraction layers, which define the inter-layer relationships that ensure connections between the various phases.

The concept *Design Agent* realizes the Requirements Agent, and is defined in terms of the associations with other design-time entities. In the context of communication, it *has* a set of *Communication*s active at a certain time (zero or more); every Communication follows a *Protocol*, that is the set of rules that govern the interaction between agents. The Protocol is in turn *composed_by* a number of *Speaks To* elements, which are the elementary (atomic) communication actions, involving two different Design Agents through the association *participates*. A Communication is a top-down realization of Dependency, implementing in the meta-model the vertical link of the conceptual map of Figure 1. Moreover, communication is connected to the abstract class *Ontology Element*, which is made concrete by *Action*, *Predicate*, and *Design concept*. Action and Predicate are connected to Design Concept, as prescribed by PASSI. The difference between PASSI's representation of Ontologies and our specification is in the realization of the requirements concept into a design concept, which enables a refinement of the Ontology during the development cycle. Ontology concept has an incoming association from Design Agent labeled *knows*, which represents the ontological knowledge of an agent.

The Design Agent *provides* a number of *Service*s, which are design-time realizations of the requirement-level concept Activity. The second realization of Activity is the concept of *Plan*, which is a common design-time construct to define the behavior of an agent. An Agent also *perceives* a *Workspace*, which realizes Topology, and can be *connected to* other Topology

entities. A Workspace can be *connected to* other workspaces. Design Agents *use* a set of *Artifact*s, which in turn expose their interface (*manifests* relation between Artifact and Design Agent). Artifacts are the realization of the requirements-time concept Resource. The relation between agents and Artifacts comes from SODA, and it is very important to connect active entities to passive entities in the system. Following SODA meta-model, an Artifact *provides* one or more *Operation*s, can *links to* other Artifacts, and is *allocated to* a Workspace.

Composition is another concept derived from SODA: here it is intended as a design-time realization of Organization. Composition is *made_of* Design Agents and Artifacts, and is specialized by *Society* (a collection of agents and artifacts exhibiting proactive behavior) and *Aggregate* (which exhibits a functional behavior).

The concept of *Rule* is quite important at design-time, for it allows constraining a number of other entities. It is a realization of both Organizational rule, Permission, and Topology; this way it enables control over disparate concerns in the multi-agent system. The concept of Rule is linked, via the association *constrains*, to many other concepts: it governs the Communication between agents, puts constraints on the Design Agent behavior, is encoded into Artifacts to define how they can be used and accessed, constrains both the Services provided and the Plans executed by the agents, and regulates the Workspace where Artifacts are located.

We did not put emphasis to the implementation phase here, because we believe that the definition of this meta-model part will be much easier when infrastructures come into place, providing the suitable abstractions to model this phase. From the methodologies we introduce just two realization: Design Agent into the *Implementation Agent*, and Service into *Implementation Task*, both coming from PASSI.

V. Conclusion and future work

In this paper, we have presented an initial version of an integrated agent-oriented meta-model which aims at being the basis for the creation of a new agent-oriented methodology integrated with MAS infrastructures through a well interrelated set of phases from requirements to implementation. The basis of the meta-model are the fragments selected from the four AOSE methodologies: Gaia, PASSI, SODA and Tropos.

The process we followed for defining the meta-model starts from the identification of the requirements for the target methodology. These requirements helped in the selection of a list of fragments from the four considered methodologies. The next two steps were the construction of a glossary, and the definition of a conceptual map of methodologies abstractions. This map was built to reflect the relations of similarity (at the same level of abstraction, that is at the same development phase) and realization (in the form of "requirements concept X is realized by design concept Y") among the abstractions adopted by each considered methodology.

The most immediate work direction is the definition of the meta-model's implementation phase, extracted from a set of MAS infrastructures. This will likely be done by adopting the process presented in Section II.

In addition, the meta-model will certainly be refined as a result of the work on the methodological aspects and the validation phase over a case study.

Another aspect to be considered for refining the presented meta-model concerns the meta-model structure: in the current version we have only two development phases that seem too coarse-grained. We will refine the meta-model splitting the two phases into different and more detailed sub-phases, e.g., the requirements analysis phase could be split into early and late requirements.

All these directions will lead to the creation of the MEnSA methodology, which will be based on the meta-model introduced here. In addition, during the definition of the methodology there will be bidirectional feedbacks between the methodology and the meta-model in order to refine again the meta-model.

Acknowledgements

Part of this work makes use of results produced by the MEnSA project (PRIN 2006) and by the PI2S2 Project managed by the Consorzio COMETA (PON 2000-2006).

References

[1] N. R. Jennings, "On agent-based software engineering," *Artif. Intell.*, vol. 117, no. 2, pp. 277–296, 2000.

[2] A. Molesini, E. Denti, and A. Omicini, "From AO methodologies to MAS infrastructures: The SODA case study," in *Engineering Societies in the Agents World VIII*, ser. LNAI, A. Artikis, G. O'Hare, K. Stathis, and G. Vouros, Eds. Springer, 2008, vol. 4995, pp. 300–317, 8th International Workshop (ESAW'07), 22–24 Oct. 2007, Athens, Greece.

[3] A. Molesini, A. Omicini, E. Denti, and A. Ricci, "SODA: A roadmap to artefacts," in *Proc. of the 6th International Workshop (ESAW 2005), Kuşadası, Aydın, Turkey, 26–28 Oct. 2005. Revised, Selected & Invited Papers*, O. Dikenelli, M.-P. Gleizes, and A. Ricci, Eds., 2006, pp. 49–62.

[4] F. Zambonelli, N. R. Jennings, and M. Wooldridge, "Developing Multi-agent Systems: the Gaia Methodology," *ACM Transactions on Software Engineering and Methodology*, vol. 12, no. 3, pp. 417–470, 2003.

[5] P. Bresciani, P. Giorgini, F. Giunchiglia, J. Mylopoulos, and A. Perini, "Tropos: An agent-oriented software development methodology," *Autonomous Agent and Multi-Agent Systems (8)*, vol. 3, pp. 203–236, 2004.

[6] M. Cossentino, "From requirements to code with the PASSI methodology," in *Agent Oriented Methodologies*, B. Henderson-Sellers and P. Giorgini, Eds. Hershey, PA, USA: Idea Group Publishing, Jun. 2005, ch. IV, pp. 79–106.

[7] M. Cossentino, S. Gaglio, A. Garro, and V. Seidita, "Method fragments for agent design methodologies: from standardisation to research," *International Journal of Agent-Oriented Software Engineering*, vol. 1, no. 1, pp. 91–121, 2007.

[8] M. Cossentino, S. Gaglio, N. Gaud, V. Hilaire, A. Koukam, and V. Seidita, "A MAS metamodel-driven approach to process composition," in *Proceedings of The 9th International Workshop on Agent Oriented Software Engineering (AOSE'08)*, M. Luck and J. Gómez-Sanz, Eds., Estoril, Portugal, 12–13 May 2008.

[9] V. Seidita, M. Cossentino, and S. Gaglio, "Adapting PASSI to support a goal oriented approach: a situational method engineering experiment," in *Proc. of the Fifth European workshop on Multi-Agent Systems (EUMAS'07)*, 2007.

[10] S. Kent, "Model driven engineering," in *IFM*, ser. Lecture Notes in Computer Science, M. J. Butler, L. Petre, and K. Sere, Eds., vol. 2335. Springer, 2002, pp. 286–298.

[11] FIPA, "Home page," http://www.fipa.org/.

[12] MenSA Group, "Deliverable 1.2," http://www.mensa-project.org/request.php?41.

[13] FIPA Methodologies, "Home page," http://www.pa.icar.cnr.it/~cossentino/FIPAmeth/.