Author(s): Beydoun, Ghassan; Low, Graham; Henderson-Sellers, Brian; Mouratidis, Haralambos; Gomez-Sanz, Jorge; Pavón, Juan; Gonzalez-Perez, Cesar.
Article Title: FAML: a generic metamodel for MAS development
Year of publication: 2009
Citation: Beydoun, G. et al (2009) 'FAML: a generic metamodel for MAS development', IEEE Transactions on Software Engineering 35 (6) 841-863
Link to published version:
http://doi.ieeecomputersociety.org/10.1109/TSE.2009.34
DOI: 10.1109/TSE.2009.34

FAML: A Generic Metamodel for MAS Development

Ghassan Beydoun, Graham Low, Brian Henderson-Sellers, Haralambos Mouratidis, Jorge J. Gomez-Sanz, Juan Pavón, and Cesar Gonzalez-Perez

Abstract—In some areas of software engineering research, there are several metamodels claiming to capture the main issues. Though it is profitable to have variety at the beginning of a research field, after some time, the diversity of metamodels becomes an obstacle, for instance to the sharing of results between research groups. To reach consensus and unification of existing metamodels, metamodel-driven software language engineering can be applied. This paper illustrates an application of software language engineering in the agent-oriented software engineering research domain. Here, we introduce a relatively generic agent-oriented metamodel whose suitability for supporting modeling language development is demonstrated by evaluating it with respect to several existing methodology-specific metamodels. First, the metamodel is constructed by a combination of bottom-up and top-down analysis and best practice. The concepts thus obtained and their relationships are then evaluated by mapping to two agent-oriented metamodels: TAO and Islander. We then refine the metamodel by extending the comparisons with the metamodels implicit or explicit within five more extant agent-oriented approaches: Adelfe, PASSI, Gaia, INGENIAS, and Tropos. The resultant FAML metamodel is a potential candidate for future standardization as an important component for engineering an agent modeling language.

Index Terms—Modeling, metamodel, multiagent systems.

1 Introduction

The advances of Model-Driven Development have motivated the application of modeling techniques to different fields. This has been seen by some researchers as an opportunity to embody the knowledge of particular aspects of their research in the form of metamodels. As a result, there are a number of metamodels developed for similar or overlapping domains of software engineering. For instance, there are various metamodels for aspect-oriented programming (e.g., [1], [2], [3]), software architectures (e.g., [4], [5], [6]), and multiagent systems (MASs) (main focus of this paper) (e.g., [7], [8]). It would be useful if the different metamodels within the same domain of software engineering (e.g., domain of MAS) could be combined into one, or at least be subsumed by one, metamodel. This clearly requires an assumption that metamodels share a number of common concepts, although they may not be expressed in exactly the same way.

For any given domain, the benefits of metamodel unification may include: Domain concepts are easier to apply for newcomers (concepts would be present in the single metamodel instead of having to look for them in a dispersed collection of extant ones); increased portability of models across supportive modeling tools (they would refer to the same metamodel); better communication between researchers (they could use the same frame of reference, i.e., the unified metamodel); and research could focus on improving and/or realizing the unified metamodel instead of being spread across a number of extant metamodels.

Synthesizing a unified metamodel from extant metamodels for a given domain is a challenge: Various extant metamodels may contain an apparent disparate variety of concepts. Certain concepts in one metamodel may be contradictory to concepts used in another as researchers may be biased to concepts supporting their particular view which is reflected in their own. Hence, a unified metamodel is forced to take into account as many of these issues as possible while maintaining internal consistency. While this is extremely difficult, we hypothesize that considerations of ambiguity, generality, and extensibility can help solve most of these problems.

Ambiguity permits a researcher (or developer) to interpret a model in the most convenient way and to deal with a degree of contradiction. Ambiguity here refers to inevitable variations in interpreting natural language definitions of metamodel concepts, be it in FAML or other existing metamodels, i.e., it is implicit that many of the

- *G. Beydoun is with the School of Information Systems and Technology, Faculty of Informatics, University of Wollongong, NSW 2522, Australia. E-mail: ghassan@uow.edu.au.*
- *G. Low is with the School of Information Systems, Technology and Management, Australian School of Business, University of New South Wales, Sydney 2052, Australia. E-mail: g.low@unsw.edu.au.*
- *B. Henderson-Sellers is with the School of Software, Faculty of Engineering and Information Technology, University of Technology, Sydney, PO Box 123, Broadway 2007, Australia. E-mail: brian@it.uts.edu.au.*
- *H. Mouratidis is with the School of Computing and Technology, University of East London, Docklands Campus, 4-6 University Way, E16 2RD, Room Number EB.G.25., London. E-mail: haris@uel.ac.uk.*
- *J.J. Gomez-Sanz and J. Pavón are with the Ingenieria del Software e Inteligencia Artifical, Facultad de Informatica UCM, Universidad Complutense de Madrid, Madrid 28040, Spain. E-mail: jjgomez@sip.ucm.es, jpavon@fdi.ucm.es.*
- *C. Gonzalez-Perez is with the Heritage Laboratory (LaPa), Spanish National Research Council (CSIC), Instituto de Estudios Gallegos Padre Sarmiento, San Roque 2, Santiago de Compostela, Galici 15704, Spain. E-mail: cesar.gonzalez-perez@iegps.csic.es.*

Manuscript received 31 Mar. 2008; revised 20 Jan. 2009; accepted 9 Feb. 2009; published online 15 May 2009.
Recommended for acceptance by R. Laemmel, A. Winter, and D. Gasevic.
For information on obtaining reprints of this article, please send e-mail to: tse@computer.org, and reference IEEECS Log Number TSESI-2008-03-0121.
Digital Object Identifier no. 10.1109/TSE.2009.34.

extant metamodels are inherently ambiguous since their semantics are expressed in natural language. While ambiguity is useful in reconciling multiple and disparate sources into a hybrid definition, aiming for consistency and generality in the end result may often be more efficacious. Generality permits specialization of a concept to be more relevant to a specific line of research. In other words, a (general) concept can stand for many things, including those that interest a particular researcher. Thus, generality provides extensibility to the metamodel. It permits researchers willing to use the unified metamodel an ability to extend concepts to incorporate their specific issues. Any new concepts would be considered for inclusion in revisions of the unified metamodel when their utility for a number of researchers is demonstrated. Therefore, using generality and extensibility, it is possible to iteratively construct a new unified metamodel from a combination of existing metamodels.

In the synthesis of our metamodel, we solely focus on the metamodel element of modeling languages (together with the abstract syntax—a.k.a. semantics—that any such metamodel defines) in the context of software agents and multiagent systems (MASs). Agents are highly autonomous, situated, and interactive software components. They autonomously sense their environment and respond accordingly. In a given MAS, coordination and cooperation between agents that possess diverse knowledge and capabilities facilitate the achievement of global goals that cannot be otherwise achieved by a single agent working in isolation. Indeed, MASs have been shown to be highly appropriate for the engineering of open, distributed, or heterogeneous systems such as market simulations and e-commerce trading environments [9], as well as for building computational models of human societies in order to study emergent behavior [10].

In the MAS literature, MAS metamodels have traditionally been the subject of study of MAS methodologies. MAS methodological approaches are typically made of, inter alia, a process-focused portion and a modeling language (to describe intermediate and final workproducts[1]). A selection of these approaches can be found in [11]; notable examples include Gaia [12], Adelfe [13], Prometheus [14], INGENIAS [15], PASSI [16], and Tropos [17]. This variety of methodologies has brought benefits in terms of research, since several ways of developing MAS have been investigated. However, it has caused difficulties. The most evident is making it harder for developers to use MAS technology as they have to choose from the large number of extant agent-oriented methodologies—often requiring the necessary acquisition of new skills to understand the methodology and its associated unique modeling language. Each specific application domain can actually be thought of as being circumscribed by a modeling language delimiting the domain of MAS development.

Assuming that there are indeed common concepts in MAS development relating to input and output workproducts, the paper shows how a relatively generic metamodel (and hence, a modeling language) for agents can be built. This metamodel would be as relevant to agent modeling as the UML metamodel has been for object-oriented modeling. Since a blind merge will almost inevitably create major semantic problems [18]; here, we aim to create a unifying metamodel by identifying common concepts in an iterative cycle consisting of both a bottom-up and a top-down analysis: Identifying common elements in existing modeling languages provides the bottom-up perspective, complemented by the top-down semantic evaluation of necessary agent-focused concepts, and aided by an intelligent use of generality. As such, it is a domain-specific (i.e., agents) application of software language engineering.

Since the concepts needed for agent-oriented software engineering extend and modify those for object-oriented software engineering, it is generally agreed that a direct, simple use of UML as an AO modeling language is inappropriate [19], [20], [21], [22]. This paper therefore does not pursue a UML-based modeling language such as a UML profile, as done effectively in AUML [23], [24] and AML [25]. Semantic problems appear when the basic element in Agent-oriented Software Engineering (AOSE), i.e., the agent, is constrained to be a subtype of the UML standard's Class construct [26]—although future work will evaluate the possibility of agent being a subtype of (UML) Component [27] or of Classifier [28].[2] The concrete proposal of this paper is the metamodel FAML (FAME[3] Agent-oriented modeling Language),[4] which was initially described in an early prototype in [18]. Later work [29] added some concepts specifically for security (not part of the relatively generic metamodel discussed here). The future goal is to create a complete modeling language (including concrete syntax, i.e., a notation) based on this metamodel that can describe the basic features of agents in an MAS. Our expectation is that, when this language is available, it will be usable to describe any workproduct produced by any methodology (since all methodologies aim to produce an MAS that is based on those needed workproducts).

The paper is structured as follows: Section 2 describes the actual process creating the initial version of our metamodel and illustrates it in terms of runtime and design-time concepts for both agent internal and agents' situational aspects. Section 3 describes the metamodel validation and shows how it can be used to generate other extant approaches, providing external validation and refinement of the metamodel leading to the final version described in Section 4. Section 5 discusses related work on agent modeling languages. Section 6 concludes the paper with a discussion of our findings and future work.

2 Creating the Initial FAML Metamodel

The FAML creation process started with an abstract representation of what may generally be expected from

1. Workproducts are artefacts of interest to the endeavor. For example, these artifacts may be inputs or outputs of the processed focused portion of the methodology. They may be described graphically or textually, thus, requiring a variety of languages such as English (natural language), AUML (modeling language), and C# (implementation language).

2. The proposal expounded in [28] is most interesting, but unfortunately was only published after we had received reviewers' comments on our original submission. An integration of that work on MAS-ML and our work on FAML must, therefore, be a topic for future work.

3. Framework for Agent-Oriented Method Engineering (FAME) is the project name under which FAML has been developed.

4. Although FAML was named initially as a modeling language, the derivation of the important metamodel MUST, in our opinion, be completed before a notation is added. Here, we focus on that metamodel as a prelude to future work on notation derivation.

MASs (e.g., [30], [31]) and continued with the study of their generic common features. Where the MAS required an agent feature that was outside the scope of the three essential definitional properties (*autonomy*, *situatedness*, and *interactivity*) or that was not commonly used, such a feature was considered as methodology-specific and outside the scope of FAML. The incorporation of these concepts, however, would still be possible for interested researchers if an extension mechanism, perhaps akin to the notion of profiles or metamodel extensions that are possible with UML, were to be used. Researchers could continue with their individual research while using the FAML metamodel. Workproducts in any methodology can be successfully generated using the metamodel, even if it requires additional, nondefinitional concepts (e.g., Adelfe, which assumes *adaptive* agents [13], or the security concerns discussed in [29]). In the creation process of FAML, internal consistency was guaranteed by ensuring that the semantics of the concepts were consistent. This sometimes required sacrificing coverage and generality, i.e., the definition of some concepts was narrowed to ensure internal consistency. In some cases, coverage and generality are opposing and trade-off decisions had to be made.

2.1 Metamodel Creation

To construct FAML, the set of concepts to be used was first determined. This described frequently occurring concepts in any MAS and the relationships among them. These concepts and their definitions were rooted in the existing literature related to MAS and MAS methodologies. Because of the sheer size of this literature, it was not possible to ensure that the initial set was comprehensive. There was a continuous incorporation of new references during the metamodeling process. Thus, the metamodel creation process was iterative. Using these iterations between top-down and bottom-up perspectives, we identified commonly used concepts that developers often use and methodologists agree upon. The FAML creation process, which was the main concern in [18], consisted of the following steps undertaken (iteratively):

Step 1: The set of general concepts relevant to any MAS and its model was determined. As noted earlier, some problem-specific concepts were omitted, e.g., terms specific to robots (e.g., actuators) or to single-agent systems were excluded. They could be included by specializing more general FAML concepts. Literature from the following areas was relevant: agent software engineering (e.g., [16], [26], [32], [33]), AI (e.g., [34], [35], [36]), distributed AI (e.g., [37], [38], and cognitive science (e.g., [39]). The output of this step was a list of concepts pending succinct definition.

Step 2: Candidate definitions were short-listed. Sources with a clear definition were given greater weight than those with only an implicit definition that can be subject to personal interpretation. Widespread occurrence of any particular definition was also taken into account leading to adopting a set concepts grounded in what other people in the agent community widely agreed. This output was a short list of candidate definitions corresponding to the list of concepts selected in Step 1.

Step 3: Differences between definitions were reconciled to ensure an internally consistent set of metamodel terms. Definitions were chosen based on consistency with earlier choices when possible; otherwise, hybrid definitions created from multiple sources were introduced. Where contradictory use of concepts between two or more sources occurred, the solution was to select the usage that was most coherent with the rest of the set of chosen concepts trying at all times to maintain generality. For example, "Task" had been defined as *a set of behavior*s in [39] and as "*behavior* but with the significance of atomic part of the overall *agent behavior*" in [40]. The definition decided for FAML is "specification of a piece of behavior that the MAS can perform," which is a hybrid definition that encompasses both interpretations since nothing is stated about atomicity. The "Task" concept in FAML includes the possibility of decomposing a task if required. Conversely, researchers requiring that a "Task" be atomic would not use task decomposition. The final output of this step was a refinement of the list of concepts obtained in Step 1 with their corresponding definitions modified (if necessary) to ensure internal consistency.

Step 4: Chosen concepts were designated into two sets:[5] design-time concepts and runtime concepts. We recognize that it is intuitive to model the "system as developed" by the software engineers as well as the "system as being executed" inside the computer. For example, in OO software development using UML [29], classes are the focus of the design activities and created by the developer whereas objects come to exist inside the computer at runtime. When using UML, the runtime elements (the objects) that occur inside the computer when the program is executed are less tightly defined because of the inbuilt bias toward classes rather than objects. In FAML, we want to go beyond this OO limitation in modeling runtime objects and provide explicit support for both design-time artefacts (the "classes"-equivalent in AOSE, i.e., agent definitions) and runtime artifacts (the "objects"-equivalent in AOSE, i.e., agents themselves). This differentiation,[6] perhaps considered arbitrary by some, can also add some indication of which stage in the development lifecycle a particular concept is likely to be useful—in comparison with an all-embracing modeling language (ML) like UML [29], which tends to confuse developers simply because it does *not* give any stage indication of its utility. Concepts in each of the two sets were further designated into two scopes, agent-external and agent-internal-related scopes, since a key differentiator for AOSE is the tight coupling between the software entity (here "agent") and its environment [42]. This allowed the separate modeling of the internal aspects of agents ensuring their successful functioning as single entities, and the modeling of aspects of an MAS that relate to interactions of individual agents within a collaborative system. Agent-internal concepts cover various types of existing agents. However, agent-external concepts cover MAS common concepts including all those that can be attributed to the essential features of agents (situatedness, interactivity, and autonomy). Less common

5. While one reviewer recommended splitting analysis from design by adding a third layer, this needs careful thought and testing beyond the constraints imposed for this current paper.

6. Effectively, we discriminate between conceptual versus machine-dependent; thus, design could be said to include "analysis," as is often done in AO methodologies (see, e.g., [41]).

	Agent-external	Agent-internal
Design time	System-level	Agent Definition-leve
Run time	Environment-level	Agent-

Fig. 1. The 2×2 matrix that is used to define four typical FAML metalevel diagrams (after [29]).

system concepts specific to certain types of MAS were omitted. For instance, *adaptivity* is a nondefinitional property of agents in that not all MASs require this property. Thus, some concepts describing an MAS in Adelfe's metamodel are deemed too specific to be included in the FAML metamodel—Adelfe's [13] adaptive system design requires learning agents.

The four designated categories are depicted in Fig. 1. The final output of this step was four categories of concepts, although the relations between concepts in each of the categories and between the categories were yet to be fully identified (see Step 5).

Step 5: Relationships within both the design-time and runtime sets and relationships interfacing the categories, from Step 4, were identified. This led to the initial metamodel.

Steps 1-5 are generic, i.e., they do not depend on the particularities or nature of any specific agent-oriented methodology. Their output represents the initial agent-oriented metamodel, which is then subjected to further evaluation and validation (Section 3).

2.2 Initial FAML Metamodel

The intention is that FAML will provide a set of generic concepts useful to a modeling language, while not necessarily providing all required details demanded by every specific agent-oriented ML. Some details are left to each individual methodology-specific ML case to be included using, for instance, specializations of FAML concepts or making an intelligent use of cardinalities. For example, if a given methodology is geared toward simulation systems composed of reactive agents, then a concept such as Plan would not be needed. This is supported with the optional (0..1) cardinalities seen in the metamodel diagrams. Instead of Plan, the association between events and actions can be utilized. This association is possible since an Action Specification includes references to preconditions and postconditions, which ultimately refer, among others, to the events produced in the system.

In connecting all filtered and synthesized concepts into a coherent metamodel, it is ensured that the set of terms is self-contained. Concepts have relationships only to other concepts within the same set. At the system level, only relations and concepts that apply due to the definitional properties of agents (autonomous, situated, and interactive) are included. As noted before, each of the two layers of FAML metamodel, the design-time and runtime layers (Fig. 1), may have two scopes: an agent-external or agent-internal scope. Four information categories exist within FAML: system level (design-time, agent-external), agent definition level (design-time, agent-internal), environment level (runtime, agent-external), and agent level (runtime, agent-internal). FAML is thus presented using four different views to clearly group classes into these four different areas of concern: design-time agent-external classes, design-time agent-internal classes, runtime agent-external (environment) classes, and runtime agent-internal classes.

The central FAML concept at design-time is *System* (i.e., an agent-oriented system), whereas the central concept at runtime is *Environment* (i.e., the context where agents reside). The concept of *Agent* also plays a central role, especially in marking the boundary between agent-internal concepts and agent-external concepts. Some classes in FAML are related to the internals of agents, while others are related to their externals. Agent-external aspects at design-time are called system level, they belong to the overall system rather than a specific agent. Agent-external aspects at runtime are called environment level, they belong to the overall environment rather than a specific agent. At the same time, and since *Agent* is a runtime concept, agents only exist at design-time as *AgentDefinitions* (i.e., specifications of the contents of agents), and therefore, agent-internal aspects at design-time are called agent definition level while agent-internal aspects at runtime are simply called agent level.

2.2.1 Design-Time Aspects

At design-time, the concept of *System* (i.e., an agent-oriented system) plays a central role (Fig. 2—see also Table 1). To start with, a system satisfies the requirements, which can be functional or nonfunctional. *Tasks* are derived from requirements. The initial metamodel used in this paper differs slightly from that originally proposed in [18] in that tasks are derived from requirements (both functional and nonfunctional) and not just from functional requirements as proposed in [18]. The restriction that nonfunctional requirements are derived from performance measures is removed as this was regarded as being too restrictive.

Roles also define a system—they can be related to each other directly to specify that a role specializes from another role, or indirectly to specify different kinds of relationships such as incompatibilities. Each role can be related to different tasks in the context of either being responsible for the task or simply as a collaborator.

The agents that will exist at runtime are described with agent definitions (Fig. 2 linked to the details shown in Fig. 3—see also Table 1). To initialize the concept *AgentDefinition*, the metamodel provides an attribute (InitialState). The agent definition has both a system generic function, which serves to initialize all the agents related to the system, and a role-specific function, which serves to initialize an agent when it joins a particular role at runtime.

A system is also composed of facet[7] definitions, i.e., definitions of aspects of the environment with which agents can interact. A facet definition can also be associated with

7. "Facet" may or may not be the best name to be used to identify environment-related concepts in the metamodel. It has the unique advantage of being a shorter word than "environment-related."

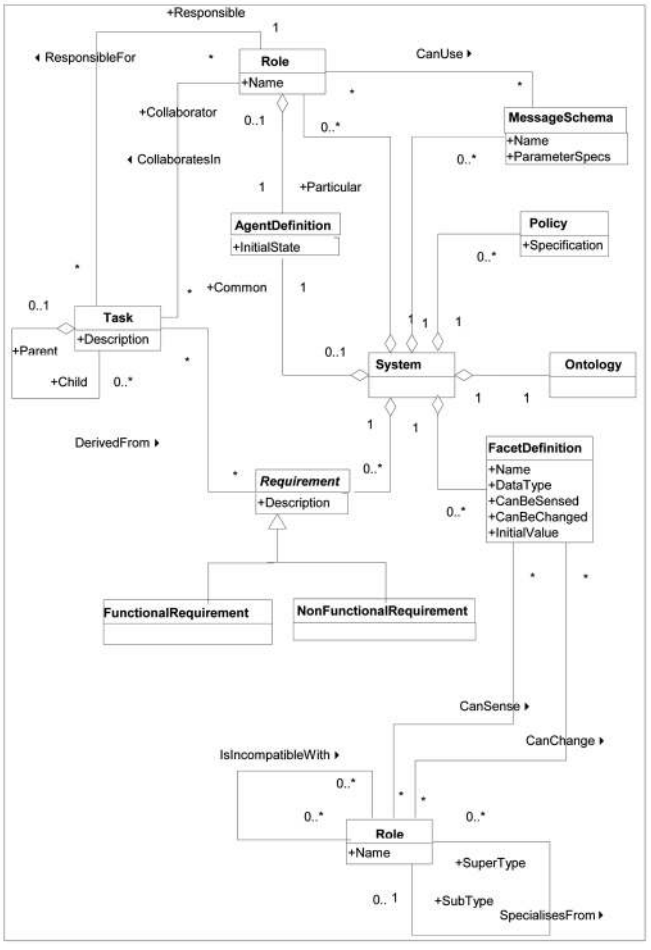

Fig. 2. Design-time agent-external classes adapted from [18]. (The diamond notation is used here to denote a generic whole-part relationship.)

each role to specify whether agents playing that role may sense or change that facet. Importantly, the environment may change independently of the agent system [54]. Facets should not be expected to remain constant between agent interactions. In fact, we envisage facets as behaving very much like variables of a physical environment, which often fluctuate from one moment to the next, with each individual measurement of them yielding different results. For instance, consider an MAS for community-based Web searching [31] in which an example of facets is details of records in the database of history of Webpage hits (at an individual Web node). These facets can change independently of the system (e.g., Webpages being removed) and are sensed by some agents depending on their roles. For example, an agent playing the role of a history manager will be able to access these, whereas an agent playing the role of a facilitator may not be able to access them.

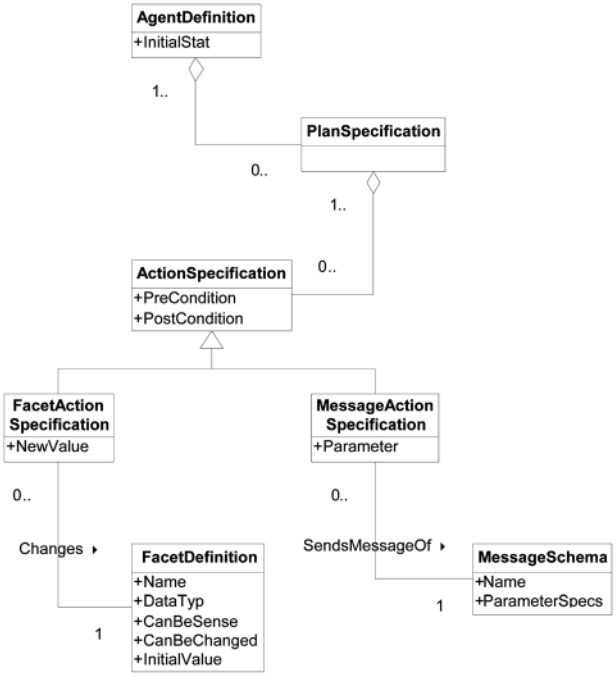

Fig. 3. Design-time agent-internal classes [18].

At the Agent Definition level, an agent definition consists of an initial state plus a number of plan specifications (Fig. 3). Each plan specification is composed of a number of action

TABLE 1
Initially Identified Design-Time Concepts and Their Definitions

Concept	Definition
Action Specification	Specification of an action of an individual agent, including any preconditions and post-conditions.
Agent Definition	Specification of the initial state of an agent just after it is created.
Environment Statement	A Boolean statement about the environment.
Facet Action Specification	Specification of a facet action in terms of the facet definition it will change and the new value it will write to the facet.
Facet Definition	Specification of the structure of a given facet, including its name, data type and access mode.
Functional Requirement	Requirement that provides added value to the users of the system.
Message Action Specification	Specification of a message action in terms of the message schema and parameters to use.
Message Schema	Specification of the structure and semantics of a given kind of messages that can occur within the system.
Non-Functional Requirement	Requirement about any limits, constraints or impositions on the system to be built.
Ontology	Structural model of a given domain.
Plan Specification	An organized collection of action specifications.
Policy	Rule that specifies an arrangement of events expected to occur in a given environment.
Requirement	Feature that a system must implement.
Role	Specification of a behavioural pattern expected from some agents in a given system.
System	Final product of an agent-oriented software development project.
Task	Specification of a piece of behaviour that the MAS can perform.

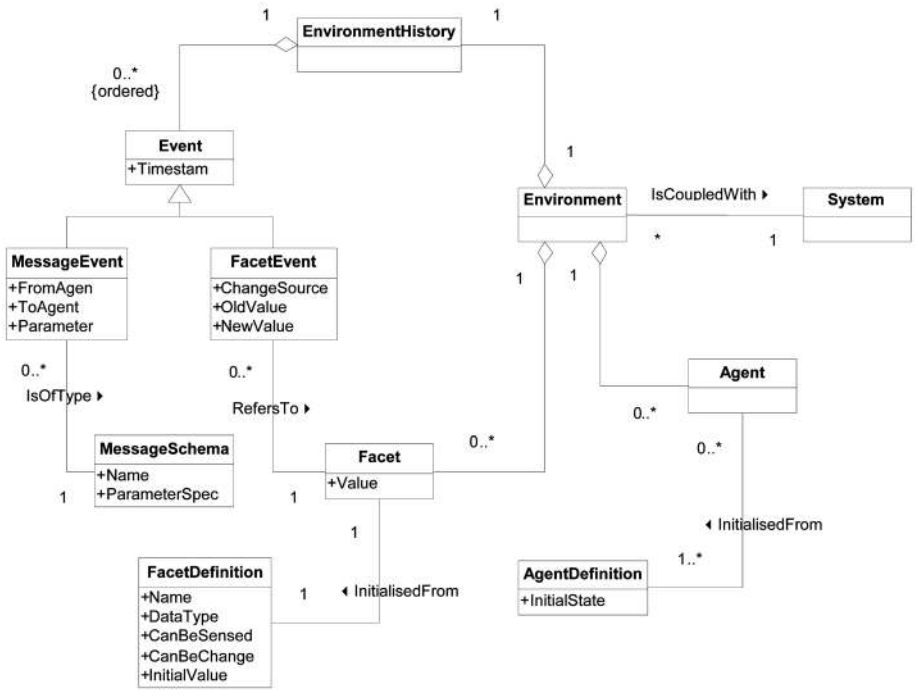

Fig. 4. Runtime agent-external classes adapted from [18].

specifications, which can be facet action specifications (which specify how to change a facet of a given kind) or message action specifications (which specify how to send a message using a given schema). Note that the term "action" is used in the context of a single agent, while the term "task" is used (see above) in the context of the complete system.

2.2.2 Runtime Aspects

At runtime, the *environment* is the central concept (Fig. 4—see also Table 2), which is tightly coupled with the design of the system. An environment is essential for agents to exist and function [42]. Environments host agents and have facets (specific aspects with which the agents can interact). The recognition of the environment as an explicit abstraction with facets as points of interactions is consistent with recent findings and emerging views articulated in [43], [44]. Additionally in FAML, an environment always has an environment history, composed of all the events that have ever occurred in that environment. Events, in turn, can be message events (i.e., events corresponding to a message being sent) or facet events (i.e., events corresponding to a facet being changed).

Looking at the agent-internal (Fig. 5—also Table 2), an agent may play a number of roles at any point in time. In

TABLE 2
Initially Identified Runtime Concepts and Their Definitions

Concept	Definition
Action	Fundamental unit of agent behaviour.
Agent	A highly autonomous, situated, directed and rational entity.
Belief	An environment statement held by an agent and deemed as true in a certain timeframe.
Desire	An environment statement held by an agent which represents a state deemed as good in a certain timeframe.
Environment	The world in which an agent is situated.
Environment History	The sequence of events that have occurred between the environment start-up and any given instant.
Environment Statement	A statement about the environment.
Event	Occurrence of something that changes the environment history.
Facet	Property of the environment with which agents can interact.
Facet Action	Action that results in the change of a given facet.
Facet Event	Event that happens when the value of a facet changes.
Intention	An actively pursued, committed desire.
Message	Unit of communication between agents, which conforms to a specific message schema.
Message Action	Action that results in a message being sent.
Message Event	Event that happens when a message is sent.
Obligation	Behaviour expected from an agent at some future time.
Plan	An organized collection of actions that can be executed to pursue a particular intention.
Role	Specification of a behavioural pattern expected from some agents in a given system.
System	Final product of an agent-oriented software development project

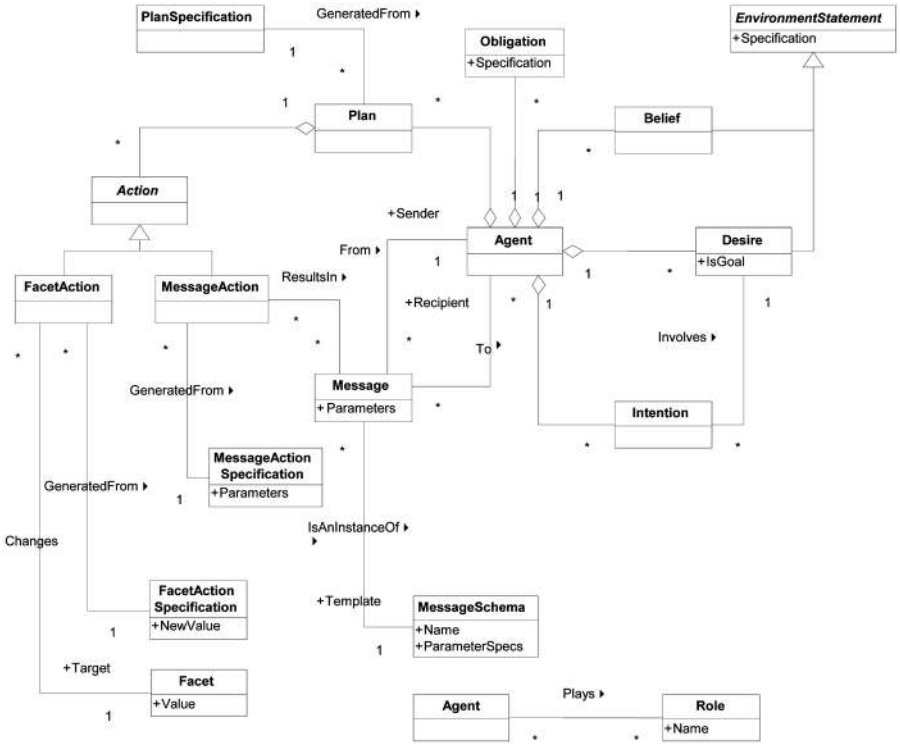

Fig. 5. Runtime agent-internal classes [18].

this version of FAML [24], an agent is shown to hold a collection of beliefs, desires, and intentions, including basic concepts supporting a BDI approach. This approach is not compulsory, since agents can exist without having any beliefs, desires, and/or intentions.[8] An agent can have a plan, which is, in turn, composed of actions. These can be facet actions (this results in a facet changed) or message actions (this results in a message sent). Agents may send messages to each other. A message always has a single sender, but can have multiple recipients.

3 Validation of FAML Metamodel

This phase is concerned with external consistency and completeness of the metamodel. This ensures external consistency with the concepts promoted by extant methodologies and that FAML can generate the majority of the modeling language components promoted in each different extant AO methodology. Generally, internal consistency (as ensured in Step 3 of the FAML creation process—Section 2) is a strong indicator of external consistency as demonstrated and highlighted in [45]. However, the goal of this validation is to undertake an external consistency check against the explicit and implicit metamodels of methodologies available in the literature. The validation phase has the following steps:

Step 1: We revisit an initial validation (reported in [18]) against two extant explicit metamodels, TAO [46] and Islander [47], and improve the coverage of the metamodel by adding concepts from both of these, which were chosen for our initial validation since, at the time of writing, their authors were among only a handful able to provide explicit descriptions of their underpinning metamodels that can be easily examined outside the context of a methodology. Moreover, they are proved metamodels that were used to lead to modeling actual, deployed, and commercial MASs. Resulting enhancements introduced into the FAML metamodel are described in Section 3.1.

Step 2: Crosscheck FAML concepts from Step 1 to ensure coverage of key modeling concepts included in the feature analysis framework for AOSE developed by [48],[9] which also reports an evaluation of the support offered by GAIA, TROPOS, Passi, Adelfe, INGENIAS, MAS-CommonKADS, Prometheus, MaSE, RAP, and MESSAGE for those key modeling concepts. This validation step refines key FAML concepts from Step 1 and is detailed in Section 3.2.

Step 3: Crosscheck FAML concepts from Step 2 against the complete metamodels of three well-known, yet differing methodologies: Gaia [50], PASSI [51], and Adelfe [52]. This validates and refines key FAML concepts from Step 2. Details of this step are given in Section 3.3.

Step 4: Empirically test the metamodel from Step 3 by attempting to generate the ML component of INGENIAS and TROPOS. This may also result in further enhancement of FAML. Details of this step are given in Section 3.4.

As we progress through the steps of the validation process, the expected decrease in the number of refinements of FAML will be a strong indicator of its completeness. While actually assessing the coverage of our metamodel in the validation steps, we examine how and to what extent

8. This apparent BDI emphasis is removed during the validation phase (see Section 3.2).

9. This work takes and extends the feature analysis validation reported in [49], which considered the original FAML metamodel and not the enhanced metamodel from validation step 1, nor did it suggest how the additional concepts might be implemented.

FAML accommodates the semantics of major corresponding concepts in the target metamodel, taking note of the following:

- Only general concepts relevant to modeling the large majority of MAS are included. Some methodology-specific concepts are omitted from FAML (as shown in Section 2).
- It is not mandatory to utilize every element of FAML. Hence, not all aspects of our metamodel are needed for every methodology-specific ML.

Comparisons between concepts of FAML and those of different metamodels underlying various agent-oriented methodologies are clearly difficult. Wherever possible, we adopt a holistic approach to any comparison, noting that different metamodels may have different ways to express the same concept or may even use the same name for different concepts.

3.1 Validation Step 1: Using Metamodels from TAO and Islander to Validate and Refine FAML

We iteratively revisit an initial validation of FAML from [18], using Islander[10] [47] and TAO [46][11] and add to FAML newly identified concepts to better describe TAO and Islander. The additional concepts enhance FAML's coverage and permit more detailed description about interagent relations, including relationships between roles and organizations of agents.

The dialogical framework of Islander offers an evolved role specification mechanism that allows various relationships between roles to be expressed, including authority relationships. To allow FAML to describe these, we add to it a new concept, *Role Dependency*, which can be refined into various authority relationships between roles. We also add to FAML a new concept of *Role Compatibility* to describe any association relationship between roles available in Islander. We deem other evolved expressive concepts of Islander as too domain specific. For instance, Islander performative structures specify interactions at a level of detail not available in FAML. These are specific to Electronic Institution MASs (e.g., Auction Websites) targeted by Islander. Hence, their inclusion in FAML would not serve our motivation to create a methodology-independent metamodel of an agent-oriented ML as discussed before. The TAO[12] refinement of the FAML concept, *FunctionalRequirement*, centers on the notion of Organization. Every TAO organization (Fig. 6) owns some agents and is coupled with a set of roles played by the agents. Each TAO organization has also goals that are decomposed into agent goals allocated to roles. The initial version of FAML [18] had an association between tasks and roles, as well as an association between roles and agent definitions. That version did not explicitly specify which agents cooperate in a given task. To rectify this, the concepts of *Organization* (Fig. 14) (and its associated *Organization Definition* (Fig. 12)) are added to FAML in order to specify which collection of agents cooperates toward a system goal. The new FAML concept, *SystemGoal*, can be subdivided and related to organizations, and these, in turn, have agents belonging to them. The inclusion of system goals (and tasks) in the design-time system-level classes also provides support for those methodologies that utilize a goal-oriented approach such as i* [54] to requirements elicitation. Similar to TAO, FAML has now a link between organization definitions and roles. Although a difference remains in TAO, the notion of organization additionally contains refinements of our notions of *Task*, which remains in FAML associated with *SystemGoals* and indirectly associated with our notion of *OrganizationDefinition*.

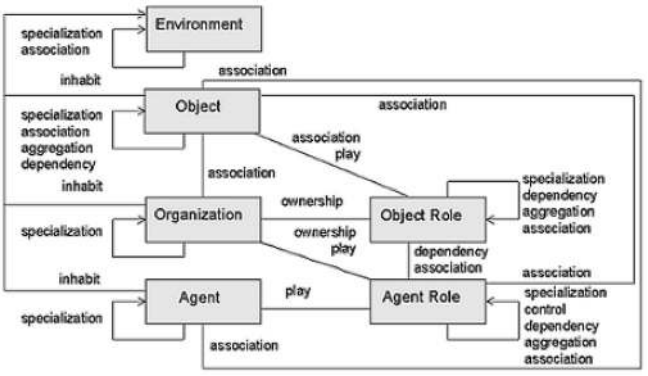

Fig. 6. TAO metamodel showing MAS-ML abstractions and relationships.

The new FAML concept, *OrganizationDefinition*, also better captures the concept of *Dialogical Framework* in Islander, which describes the roles and their relationships as a set of cooperative units that may involve a number of agents. Islander and TAO's support for the revised FAML concepts after this validation stage is shown in Tables 3 and 4, respectively.

3.2 Validation Step 2: Using Feature Analysis

Tran and Low [48] developed an evaluation framework using various feature analysis frameworks applied to evaluating AOSE methodologies [55], [56], [57], [58], identifying and integrating four categories of evaluation criteria: *Process-Related*, *Technique-Related*, *Supportive Features-Related*, and *Model-Related*. Only concepts represented by Model-Related Criteria refer to the workproducts of a methodology. They examine capabilities and characteristics of a methodology's models and notational components as well as their support for agent characteristics. We use features only from the Model-Related Criteria (considered by Tran and Low [48]) to examine the coverage of FAML for key modeling concepts since FAML aims to support the workproducts rather than processes or techniques involved in creating them (Table 5). Tran and Low [48] reported their evaluation of the support offered by GAIA, TROPOS, Passi, Adelfe, INGENIAS, MAS-CommonKADS, Prometheus, MaSE, RAP, and MESSAGE for these key modeling concepts. This validation step refines key FAML concepts from Step 1.

10. The Islander metamodel is not explicitly described in any published diagrams. The metamodel information used here was from personal communications with the author of Islander [53] and the graphical software interface of the metamodel editor.

11. MAS-ML [28], as an extension to TAO, was published much later (September 2008) and was, therefore, not able to be analyzed in detail for this current paper.

12. The TAO metamodel retains object-oriented design concepts along with its agent-oriented design concepts. In our past and current analyses, we are concerned only with the agent-oriented features of TAO.

TABLE 3
Islander Support for Revised FAML Concepts (After Step 1)

FAML Concept	Corresponding Islander Support
Agent definition	Implicit in: sub-task allocation to agents, message specification assignment and constraints, and association between messages and roles within scenes.
Environment	Implicit in the collection of interactions available to all agents within the system and determined by external agents leaving or entering scenes.
Environment History	Stacks of message (a stack of messages exists for each scene).
Facet Event	External agents entering or exiting a scene
Functional Requirement	Performative Structure (A set of related scenes)
Message	Message
Non-functional requirements	Number of agents per scene, synchronization of agents
Obligation	Obligations (of activities within a scene).
Ontology (domain structure)	Ontology (messages structures)
Organization Definition	Dialogical Framework
Policy	Norms, Constraints
Role Dependency	Authority
Role Compatibility	Implicit: roles are not related
Role Relationship	Subsumed (sub) relation between roles OR Strictly Separated (SSD) relation between roles
Roles	Roles
Task	Scene state

We find that the extended FAML after validation step 1 (Section 3.1) supports most of the features, however, all 10 methodologies analyzed in [48] distinguish between *System Goals* and *Agent Goals* and FAML does not. To ensure its broader applicability, and at the same time, its comprehensibility for developers who are not familiar with BDI, we add to FAML a new concept, *AgentGoal*, and delete both Desire and Intention of an agent (shown in Fig. 15). This new concept can if necessary be used to implement a BDI agent architecture by using an attribute "Committed." If a goal is committed, it can correspond to an intention, an uncommitted goal can correspond to a desire. This notion of commitment is in accordance with the goal and task metamodel of [59]. It makes the concept *Obligations*, shown in Fig. 5, redundant as replaced by the state of commitment of an agent to achieve certain states. Hence, we delete *Obligations.*

FAML was also found deficient with respect to services provided by agents, support for interaction protocols, system architecture, and use cases (Table 5). Support for services and interaction protocol was added. However, support for system architecture and workproducts from specific requirements elicitation techniques (e.g., use cases) was regarded as methodology-specific and outside the general scope of FAML. Particularly, the *Requirements* class in FAML can be used to describe requirements from various elicitation techniques.

TABLE 4
TAO Support for Revised FAML Concepts (After Step 1)

FAML Concept	Corresponding TAO Support
Action	Agent rights
Policy	Organization axioms
Message Schemas	Protocols
Organization	Organization
Role	Role
System	Main Organization
System Goal	Goals (within an organization)
Task	Responsibility of an organization

3.3 Validation Step 3: Using Adelfe, PASSI, and Gaia

This validation step is prompted by an integration paper proposing an MAS metamodel unification based on three AO methodologies—Adelfe, Gaia, and PASSI [60]. Our metamodel FAML is compared with each metamodel of those three methodologies, leading to its further refinement and modification. As it turns out, many concepts of the three methodologies are already available in FAML; we only discuss features that are missing or those that eventually lead to some modification of FAML.

The Adelfe methodology is tailored toward building adaptive MASs (Fig. 7), by designing agents with a cooperation-driven social attitude [60]. Agents in adaptive MAS, as targeted by Adelfe, pursue local internal goals (Agent Goals in FAML) and try to cooperate with other agents. The concept of cooperation is embedded within its metamodel. Each Adelfe agent has a set of cooperation rules that allows the agent to detect and resolve noncooperative situations (exceptions) such as incomprehension, ambiguity, incompleteness, unproductiveness, concurrency, conflict, and uselessness. In contrast, FAML accommodates cooperative behavior through shared system goals between agents. The high level of cooperation between Adelfe agents is not found in most methodologies [7]. Adelfe constrains an agent behavior with a cooperative attitude [52] and it deliberately omits a number of concepts. For instance, Adelfe does not support *System Goal* and *Organization* included in FAML. Adelfe uses a modified form of the common Belief-Desire-Intention (BDI) architecture for individual agents, where Aptitude, Skill, and Representation combine the planning intention of an agent with its desire to take an action as well as a description of beliefs about the world. Basic beliefs are captured in *characteristics* of an agent. This analysis of Adelfe suggests adding to FAML a high-level concept, *MentalState,* which supports this modified BDI architecture, but at the same time, remaining consistent with the available support for a traditional BDI architecture through beliefs and agent goals (see Section 3.2

TABLE 5
Support of Concepts by FAML Metamodel

Concepts	Support by FAML metamodel after validation step 1
Agent acquaintance	Role Relationship classes
Agent aggregation relationship	Extension of Agent Definition class
Agent architecture	Agent Definition class
Agent belief/knowledge	Belief
Agent capability/service	Not supported
Agent goal/task	Not directly supported
Agent inheritance relationship	Extension of Agent Definition class
Agent instances deployment	Agent Definition class
Agent instantiation	Agent class
Agent percept/ event	Event class
Agent plan/reasoning rule/problem solving method	Plan class
Agent-role assignment	Role Specification between Role and Agent Definition classes
Content of exchanged messages	Message class
Domain conceptualization	Ontology classes
Environment characterization	Environmental level classes
Environment resource/facility	Environmental level classes
Interaction protocol	Not supported
Inter-agent contract/commitment	Obligation class
Organizational structure/inter-agent social relationship	Organization class Inter-agent social relationship: handled via roles
Role	Role class
System architecture	Not supported
System goal	System Goal class
System task/behaviour	Task class
Use case scenario	Indirectly supported via functional requirement

for more details). The addition of *MentalState* (and its associated initial *MentalStateSpecification*) may also accommodate other deviations from the BDI architecture, e.g., Gaia or PASSI [60].

In PASSI, agents are implemented with an FIPA-compliant platform, allowing PASSI to focus on the Problem and Agency Domains, leaving the Solution Domain to the FIPA infrastructure. The Problem Domain considers the problem in terms of functional requirements, scenarios, ontology, and resources while the Agency Domain considers the elements of the agent solution: agent, role, task, and communication (see Fig. 8). PASSI adopts use cases to express functional requirements. As already noted in Section 3.2, workproducts from specific requirements elicitation techniques (e.g., use cases) are regarded as methodology specific and are not explicitly included in FAML (they are captured in the FAML *Requirements* class). Scenarios are used to describe the sequence of interactions between actors (roles) and the system. PASSI agents have roles that "are portions of the agent's social behavior characterized by some specificity such as a goal, or providing a functionality/service, and in so doing, it can also access some resources" [7]. Communications between roles are composed of a number of messages. PASSI provides a service to the agent society (or organization in FAML) using resources. To enable FAML to model this, the concepts *Resource* and *ResourceSpecification* are added to FAML.

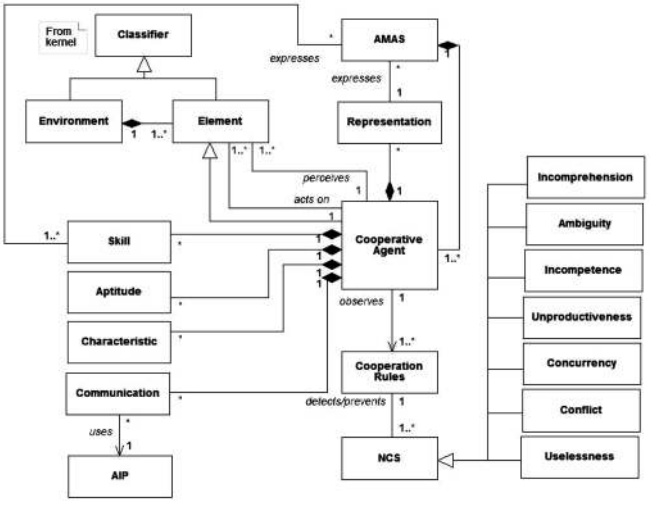

Fig. 7. Multiagent system metamodel for Adelfe [7].

Gaia was originally developed to handle small-scale, closed agent systems [61]. It modeled agents, roles, and interactions without considering their social aspects. It was later extended to model social aspects such as social goals, social tasks, and social organizational rules [62] (see Fig. 9). Gaia does not consider the requirements elicitation phase. Hence, the metamodel does not support a *Requirements* class (or equivalent) as seen in the FAML. The official extension of Gaia [62] emphasizes the social aspects of agent with the organization having organizational rules and these govern the communication between roles and form an organization structure. In FAML, we distinguish between the two concepts: *Organization* (the collection of agents) and the *OrganizationDefinition*. To enable FAML *Policies* to directly describe the organizational rules of Gaia, we add a link between *OrganizationDefinition* and *Policy*. This allows a modeler to express the same information expressed by organizational rules and organization structure in Gaia by *OrganizationDefinition* and associated *Policies*, respectively.

Metamodels of all three (Adelfe, PASSI, and Gaia) support the concept of communication representing a higher level of abstraction than the *Message* concept in

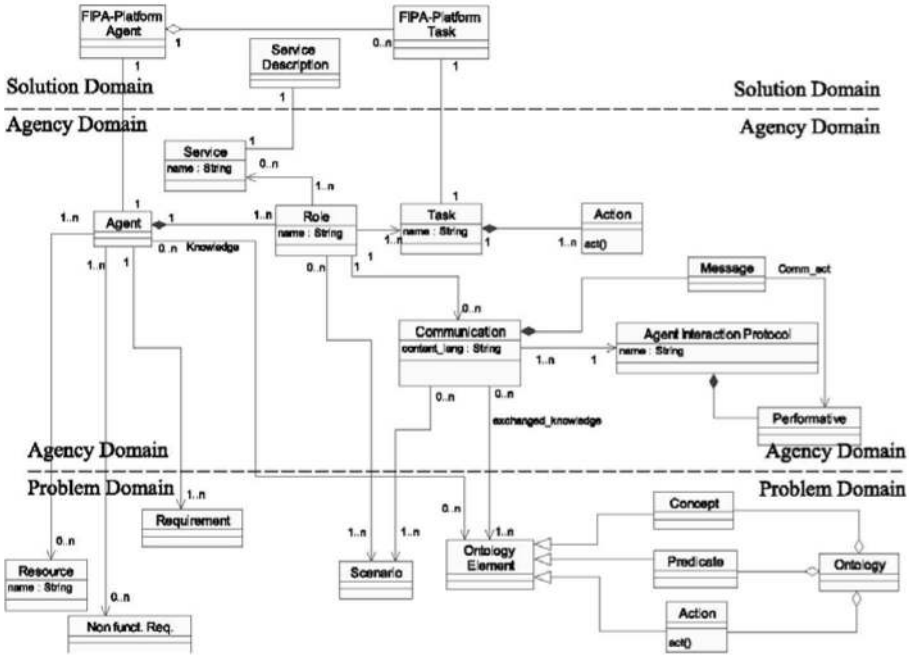

Fig. 8. Multiagent system metamodel for PASSI [7].

FAML. Communication is a domain-related interaction between agents abiding by an interaction protocol. It is knowledge exchanged and is normally composed of one or more messages each associated with a performative (*Message Schema* in FAML). *InteractionProtocol* was added previously (in Section 3.2). To further support communication modeling, we add the concept *Communication*.

3.4 Validation Step 4: by Instantiation against INGENIAS and TROPOS

This validation step describes how FAML can be used to generate the metamodels of two well-known methodologies, INGENIAS [7] and Tropos [63]. This has been undertaken collaboratively with the corresponding cocreators of the metamodels (authors of this paper as well) and provides an excellent vehicle for a further and in-depth validation and analysis.

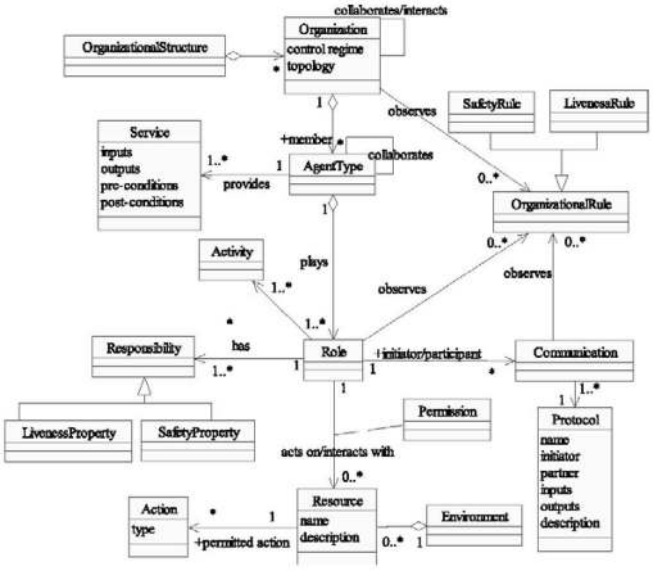

Fig. 9. Multiagent system metamodel for Gaia [7].

3.4.1 Validation against the INGENIAS Metamodel

FAML successfully derives all key concepts in the metamodel of INGENIAS (Fig. 10) in the requirement, analysis, and design phases.

Not all of FAML was used to derive the INGENIAS metamodel, as expected. Details of the INGENIAS derivation from FAML are shown in Tables 6 and 7 and discussed further in this section, with particular emphasis on aspects that required subtle considerations.

INGENIAS, similar to Adelfe, supports a variant of the BDI internal agent architecture. Our new runtime concept *MentalState* (added in Section 3.3) provides support for the INGENIAS Mental State concept. In INGENIAS, mental states are modeled as goals, facts, and beliefs [60]. INGENIAS Agent goals at runtime correspond to committed agent goals in FAML. INGENIAS deviates from standard BDI by not differentiating between facts and beliefs. INGENIAS prescribes to every concept associated with a mental state a set of specific methods for mental state processing (see Fig. 10). This level of details is not available in FAML and deemed as INGENIAS-specific. The INGENIAS concept of Organization is both a runtime and design-time concept. In contrast, FAML differentiates between the design-time concept *OrganizationDefinition* and the runtime concept *Organization*. INGENIAS concepts describing deployment and testing phases (e.g., definition of tests) are not currently considered by FAML.

3.4.2 Validation against the Tropos Modeling Metamodel

Tropos, similarly to other AOSE methodologies, supports the analysis and design activities of the software development

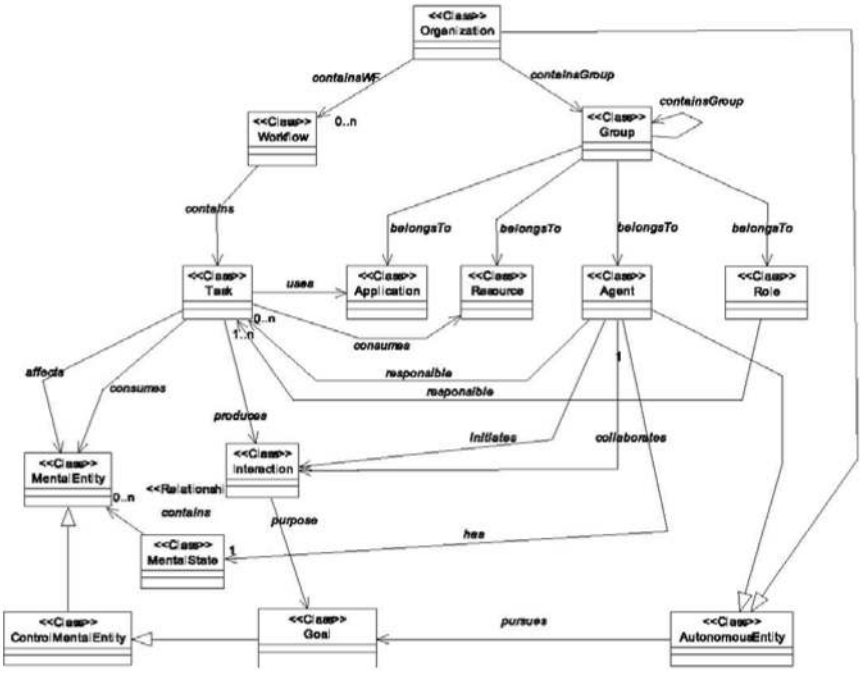

Fig. 10. Multiagent system metamodel for INGENIAS [7].

process, employing an agent-oriented view. However, Tropos is requirements driven and heavily draws on concepts from the requirements engineering discipline such as actors and goals. It was designed to be implementation independent [17]. A number of agent platforms have been used to implement its designs, including JACK [64] and JADE [65]. It can be used independently of any specific agent-oriented implementation (as presented in [9]).

Tropos supports four main stages: the *Early Requirements* analysis, where the focus is on understanding the organizational setting in which the system-to-be will be situated, identifying stakeholders and their intentions, the *Late Requirements* analysis, where the focus is on understanding the interactions and dependencies of the system and its corresponding environment, expressed in a set of functional and nonfunctional requirements, the *Architectural Design*, where the focus is on the architectural specification of the system, and the *Detailed Design*, where the focus is on specifying in detail the various components (agents) of the system. Tropos is supported by a modeling language that is based on the i* framework [51], which facilitates goal analysis, with a grammar consisting of concepts, such as actor, goal, task, and agent as shown partially in the metamodel in Fig. 11. The language is also supported by a visual notation and a formal definition [66] of its grammar.

Our validation showed that FAML successfully generates the key concepts of Tropos [10] (see Table 8). In generating the Tropos metamodel from FAML, we focus on key aspects supported by Tropos such as Agent Structure and Agent Interactions. For Agent Structure, Tropos employs the concept of Actor as a generalization of an agent with its behavior characterized by the concept of Role. The concept of Position is employed to indicate a set of roles. As such, a Tropos agent can occupy a position and play a role. The Tropos concepts Actor and Position are not directly supported by FAML. A Tropos Actor is at a higher level of abstraction than an FAML agent. However, an actor can be defined in an abstract way by combining the concepts of *AgentDefinition* and *Role* and using the relationship between them. Similar to Tropos, this in FAML does not explicitly differentiate between "internal" and "external" actors but it allows in the same way one to capture either internal or external actors (similarly in Tropos). FAML supports the Tropos concept Position via a specialization of *OrganizationDefinition*.

A Tropos agent has intentional elements such as goals and plans, beliefs, as well as capabilities that support the achievement of the intentional elements (Table 8). The FAML concept of *PlanSpecifications*, in association with *AgentDefinition*, generates capabilities (assuming that a capability in Tropos is provided by the complete or partial execution of a plan). The new concept, *MentalStateSpecification* (added as a result of the validation against Adelfe in Section 3.3), can be used to describe basic structures and elements of runtime mental states. This includes beliefs, similar to beliefs as used in Tropos. For agent interactions, Tropos employs both UML and AUML (Agent Unified modeling Language [24]) during its detailed agent design. Capability, plan, and agent interaction diagrams are used to model an agent's goals, beliefs, and capabilities, as well as communication acts among the agents of an MAS. These are all generated by FAML as discussed above. Moreover, at design-time, the FAML metamodel defines a number of concepts such as the FAML *System*, *System Goal*, and *Task*, which have similar counterparts in the Tropos metamodel as MAS, Goal, and Plan, respectively. A point of difference is that the Tropos metamodel does not define the concept of *Requirement*, as in FAML. Instead, Tropos supports the definition of functional requirements, modeled as hard-goals, and nonfunctional requirements, modeled as soft goals.

As mentioned before, Tropos is implementation independent. Its metamodel does not define runtime concepts. A validation of the FAML runtime concepts against Tropos is not possible.

TABLE 6
INGENIAS Support for Design-Time Concepts and Their Definitions

FAML Concept	FAML Definition	INGENIAS Concept	INGENIAS Definition
Action Specification	Specification of an action, including any preconditions and post-conditions.	Task	The ability to transform the internal mental state of the agent as well as the environment in a certain way.
Agent Definition		N/A	
Environment Statement	A Boolean statement about the environment.	Conditional Mental State	A condition to be hold considering information from the mental state of an individual agent or from the environment.
Facet Action Specification		N/A	
Facet Definition	Specification of the structure of a given facet, including its name, data type and access mode.	Application	It represents an entity of the environment the agent can act on. Its definition includes a list of operations with a method-like signature.
Functional Requirement	Requirement that provides added value to the users of the system.	Use case	It represents an expected use of the system
Interaction Protocol	Specification of patterns of communication	Interaction/ Specification	An interaction is the first class representation of an information exchange act among several agents. It pursues a goal which is related to the goal of its participants. The interaction protocol is defined in a Specification entity, which declares which notation and formalism is used to represent the protocol and points out to its content.
Mental State Specification	Specification of a mental state in terms of specified beliefs and intentions.	Mental State	It represents the mental state of an agent. If associated to an agent, it means the initial mental state.
Message Action Specification	Specification of a message action in terms of the message schema and parameters to use.	Interaction Unit	An interchange of information among two agents or roles
Message Schema	Specification of the structure and semantics of a given kind of messages that can occur within the system.	Speech act	An attribute of an Interaction Unit that indicates what speech act is being assumed
Non-Functional Requirement		N/A	
Ontology	Structural model of a given domain.	Mental state entities	There are sets of information entities for defining the relevant knowledge about the world.
Organization Definition	Specification of a collection of roles and agents co-operating towards a system goal.	Organization	Representing a group structure of roles, agents, and resources that work together to achieve some common goals
Plan Resource Specification	Specification of resources that are used in the Plan Specification.	Association between plans/ Resources/ Application	The association means that the plan requires that specific resource or application
Plan Specification	An organized collection of action specifications.	Plan	A set of tasks to be executed within a single agent
Policy		N/A	
Requirement	Feature that a system must implement.	Use Case	It represents an expected use of the system
Resource Specification	A resource specification specifies something that has a name, may have reasonable representations and that can acquired, shared or produced.	Resource/ Application	A plan can be associated to these entities. A resource is an entity of the system whose existence is quantified. An application represents an entity in the environment or the internals of the agent which offers an API.
Role	Specification of a behavioural pattern expected from some agents in a given system.	Role	A reusable behavioural specification. Playing a role implies acquiring the responsibilities associated to the role.
Role Compatibility		N/A	
Role Dependency		N/A	
Role Relationship	Social relationship between two roles for a given purpose.	Sub-ordination relationships	Relationships define among agents and roles indicating that an agent/role must provide a service conditionally or unconditionally to another agent/role. Relationships can be defined as well between groups or organizations. The scope of this kind of relationships covers all agents/roles belonging to each group or organization.
Service	A single, coherent block of activity in which an agent will engage.	Workflow/ goal	A collaboration among several agents to provide a service. The service is usually associated to an organization goal.
System		N/A	
System Goal	Specification of the state of the environment that the system tries to achieve.	Goal	A representation of a state to achieve. It is not specified if the state is an environment state or an agent internal state.
Task	Specification of a piece of behaviour that the system can perform.	N/A	

4 The Resultant, Revised FAML Metamodel

The final revised version of the FAML metamodel is shown in Figs. 12, 13, 14, and 15. As noted before, FAML has two layers: design-time and runtime layers. Each layer has two scopes: an agent-external and an agent-internal scope. FAML is presented in four different diagrams supplemented by two tables (Figs. 12, 13, 14, and 15, and Tables 9 and 10), which, together, clearly group metamodel classes into the four categories: design-time agent-external, design-time agent-internal, runtime agent-external (environment), and runtime agent-internal classes (as per Fig. 1).

TABLE 7
INGENIAS Support for Runtime Concepts and Their Definitions

FAML Concept	FAML Definition	INGENIAS Concept	INGENIAS Definition
Action	Fundamental unit of agent behaviour.	Task	The ability to transform the internal mental state of the agent as well as the environment in a certain way.
Agent	A highly autonomous, situated, directed and rational entity.	Agent	An autonomous entity with goals and the abilities to reach them.
Agent Goal	An environment statement which represents a state pursued by an agent	Goal	Goal concept can be associated to both an organization, meaning then a state of the system, or to an agent, meaning a state of the world using information accessible by the agent
Belief	An environment statement held by an agent and deemed as true in a certain timeframe.	Belief + Fact	Mental entities representing information assumed to be true by the agent.
Communication	Composition of more than one message.	Interaction	An interaction between two or more agents or roles, with at one initiator and at least one collaborator.
Environment	The world in which an agent is situated.	Set of applications	Each application represents an entity of the environment the agent can act on.
Environment History		N/A	
Environment Statement	A statement about the environment.	Conditional Mental State	A condition to be held based on information from the mental state of an agent or from the environment.
Event	Occurrence of something that changes the environment history.	General Event	A mental entity representing something happened in an application
Facet	Property of the environment with which agents can interact.	Application	It represents an entity of the environment the agent can act on. Its definition includes a list of operations with a classic method signature (return result, name of the operation, parameters)
Facet Action		N/A	
Facet Event	Event that happens when the value of a facet changes.	General Event + Perception relationships	Relationships between an agent and an application indicating that the agent should expect a kind of mental entity to be produced by the application.
Mental State	Environment statement and intentions held by an agent at a certain timeframe.	Mental State	The mental state of an agent in a certain moment is an aggregate of mental entities. Whenever you want to express what mental entities an agent is supposed to have at activation at the beginning. This is expressed in an agent model by associating an agent entity to a mental state.
		Mental State Pattern	A configuration of the agent mental state to be achieved.
Message	Unit of communication between agents, which conforms to a specific message schema.	Interaction Unit	An interchange of information among two agents or roles. The interaction unit includes references to the Mental Entities to be considered.
Message Action	Action that results in a message being sent.	Relationship UIInitiates	This indicates that an initiator of the Interaction Unit will deliver information if certain conditions are met.

Design-time agent-external classes (Fig. 12) are concerned with features that can only be perceived by looking at the whole system at design-time. These are typically at the "type" level (as opposed to runtime instances of these concepts that are individualistic). These include the following:

- requirements and their relationships with goals;
- roles, relationships between roles, relationships with message schemata and services;
- tasks, together with their relationships with roles and goals;
- agent definitions and their relationships with roles and organization definitions;
- use of ontologies to define domain application semantics; and
- environment access points and their relationships with roles.

In contrast to the initial version of the metamodel (Fig. 2), classes have been added for *RoleRelationship* (two subtypes), *SystemGoal*, *MentalStateSpecification*, *OrganizationDefinition*, *Service*, and *InteractionProtocol*. Other small changes have been made as detailed in the validation phase (Section 3). The metamodel now also explicitly shows that the environment may change independently of the agent system [54] (via the addition of the attribute CanChange to the *FacetDefinition* concept).[13]

Fig. 13 shows the classes related to the agent internals at design-time.

- Plan specification (if any), which uses a plan resource specification.
- Action specification, which can be a facet action or a message action specification.
- How action specification relates to facet definitions and message schemas.

13. As a result of reviewer comment.

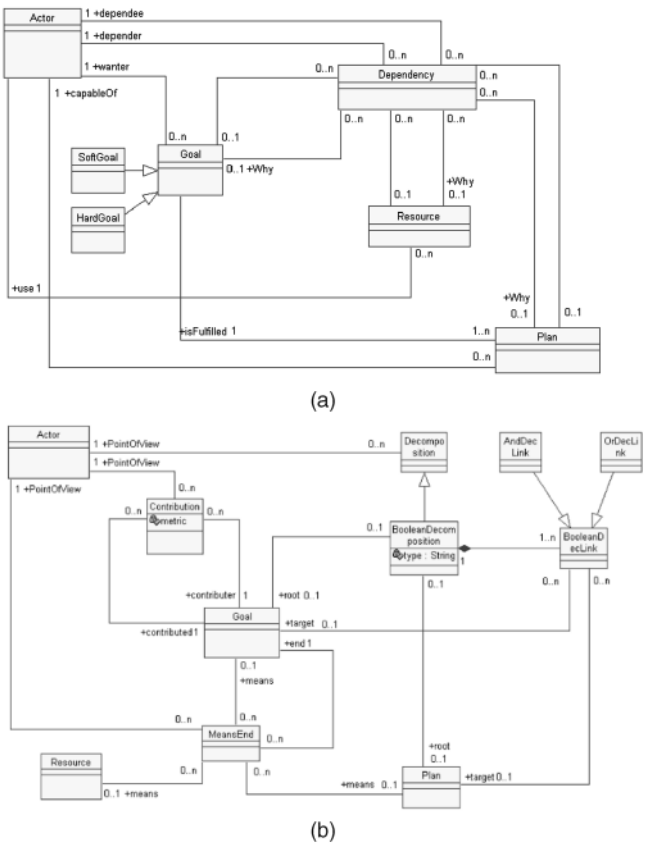

Fig. 11. Multiagent system metamodel for TROPOS. (a) Part of the Tropos metamodel focusing on the actor concept [63]. (b) Part of the Tropos metamodel focusing on the Goal and Plan concepts [63].

The main modification made as a result of the research reported in this paper is the addition of the classes, *PlanResourcesSpecification* and *ResourcesSpecification* (cf., Fig. 3). PlanSpecification is also further detailed to include "GoalCondition," "FailureCondition," and "SuccessCondition" attributes. The first attribute characterizes the goals for which the plan may be applicable. This is a plan-level test that applies before any of the preconditions of actions are used (shown in ActionSpecification). The second new attribute aims to express when a plan will be unable to attain the intended goal. Success condition describes when a plan can be considered to have successfully attained the goal. The use of the information contained within these attributes depends on the chosen agent behavior paradigm. For instance, a failure condition may be satisfied before a plan is finished. Hence, it may make sense to abort the current plan and try another. Using the success condition has similar considerations, e.g., the selected agent behavior paradigm may require a time-out before checking the success of a plan. This would be required if the effects of actions in the environment took some time to be noticed. Details of further development of these attributes are left to designers and may depend on the methodology.

Fig. 14 shows the classes related to the environment in which agents "live" (at runtime). These classes are also at the "type level" and coexist with instances of design-level "types" (for instance, as shown in Figs. 12 and 13 and discussed above). Runtime environment-related classes are concerned with MAS features that exist only at runtime in the environment.

Runtime artifacts supported by FAML are as follows:

- environment history of totally ordered instantaneous events including a message log;
- events of different kinds;
- system access points and relationships with events, resources, and organizations; and
- relationships between agents and the above concepts.

Improvements to the original metamodel (Fig. 4) are the inclusion of classes to represent *Resource* and *Organization* (and its definition).

Finally, Fig. 15 shows the classes related to agent internals at runtime. These classes can only be perceived by considering the internals of agents at runtime. These include:

- plans and actions;
- relationships among actions, messages, and message schemata;
- communication and its relationship with messages and interaction protocol;
- mental states and relationships with agent goals and beliefs; and
- relationships between each of the above and the environment.

Important changes (cf., Fig. 5) are: deemphasis of the BDI architecture for MASs, replaced by classes for *MentalState* and *AgentGoal*; the introduction of a more generic *Communication* class together with an *InteractionProtocol* class. Two runtime corresponding attributes FailureCondition and SuccessCondition are also introduced to Plan. The first describes when a plan cannot attain the goal it is pursuing and the second describes when a plan can be considered to have successfully attained this goal. The evaluation of both of these conditions depends on the designer and may also depend on the methodology. Another attribute PlanDescriptors is introduced to AgentGoal.[14] This new attribute specifies what plans may be suitable to be used toward the goal. This acknowledges that many plans may be suitable for a given goal and that the suitability of a particular plan to a goal cannot be assured at design-time. These new attributes can be used to model the behavior of a deliberative agent at runtime in choosing and abandoning plans as they fail or as goals change.

5 Related Work

While there are many elements of an agent-oriented modeling language in the literature, many of these are implicit by being described in methodology-focused papers, e.g., well-known methodologies such as Gaia [12], [62] and Prometheus [14]. This means that only the notation and the suite of associated diagrams are described without an explicit metamodel. However, concepts described in these methodologies have all been considered and taken into account in our original synthesis of concepts [18]. In

14. While we observed similar attributes in Mental States of Agents in the methodology Ingenias, we construed them as too methodology specific. However, a reviewer's comment triggered this addition as well as the attributes of PlanSpecification for agent-internals at design-time.

TABLE 8
Tropos Support for Design-Time Concepts and Their Definitions

Concept	Definition	Added at validation step
Action Specification	Specification of an action, including any preconditions and post-conditions.	
Agent Definition	Specification of the initial state of an agent just after it is created.	
Environment Statement	A Boolean statement about the environment.	
Facet Action Specification	Specification of a facet action in terms of the facet definition it will change and the new value it will write to the facet.	
Facet Definition	Specification of the structure of a given facet, including its name, data type and access mode.	
Functional Requirement	Requirement that provides added value to the users of the system.	
Interaction Protocol	Specification of patterns of communications that occur in the system	2
Mental State Specification	Specification of the initial mental state in terms of specified beliefs and agent goals	3
Message Action Specification	Specification of a message action in terms of the message schema and parameters to use.	
Message Schema	Specification of the structure and semantics of a given kind of messages that can occur within the system.	
Non-Functional Requirement	Requirement about any limits, constraints or impositions on the system to be built.	
Ontology	Structural model of a given domain.	

FAML Concept	FAML Definition	Tropos Concept	Tropos Definition
Action Specification	Specification of an action, including any preconditions and post-conditions.	Plan	A sequence of activities an agent follows to achieve one or more goals.
Agent Definition	Specification of the initial state of an agent just after it is created.	Agent	Software having properties such as autonomy, social ability, reactivity and pro-activity.
Environment Statement		N/A	
Facet Action Specification		N/A	
Facet Definition		N/A	
Functional Requirement	Requirement that provides added value to the users of the system.	(Hard)goal	A strategic interest of an actor.
Interaction Protocol	Specification of patterns of communication	Interaction Protocol	Specification of possible communication scenarios between agents in a multi-agent system
Mental State Specification	Specification of a mental state in terms of specified beliefs and intentions.	Belief	Actor knowledge of the World
Message Action Specification	Specification of a message action in terms of the message schema and parameters to use.	Communicative act	A set of messages between agents
Message Schema	Specification of the structure and semantics of a given kind of messages that can occur within the system.	Interaction Protocol	Specification of possible communication scenarios between agents in a multi-agent system.
Non-Functional Requirement	Requirement about any limits, constraints or impositions on the system to be built.	Soft-goal	A goal that has no clear criteria for deciding whether it is satisfied or not.
Ontology	Structural model of a given domain.	Tropos models	Graphical models of an actor's goals, plans and dependency relationships.
Organization Definition	Specification of a collection of roles and agents co-operating towards a system goal.	Multi-agent system	Set of agents that have a common goal.
Plan Specification	An organized collection of action specifications.	Plan/Task	It represents, at an abstract level, a way of doing something.
Plan Resource Specification	Specification of resources that are used in the Plan Specification.		
Policy		N/A	
Requirement	Feature that a system must implement.		
Resource Specification	A resource specification specifies something that has a name, may have reasonable representations and that can acquired, shared or produced.	Resource	Physical or an informational entity
Role	Specification of a behavioural pattern expected from some agents in a given system.	Role	Abstract characterization of the behaviour of a social actor within some specialized context or domain of endeavour.
Role Compatibility		N/A	
Role Dependency	Role relationship in which the source role depends on the destination role for a given purpose.	Dependency	Relationship between two actors, which indicates that one actor depends, for some reason, on the other in order to attain some goal, execute some plan, or deliver a resource.

FAML Concept	FAML Definition	Tropos Concept	Tropos Definition
Role Relationship	Social relationship between two roles for a given purpose.	Dependency	Relationship between two actors, which indicates that one actor depends, for some reason, on the other in order to attain some goal, execute some plan, or deliver a resource.
System	Final product of an agent-oriented software development project.	Multi-agent system	Set of agents that have a common goal.
System Goal	Specification of the state of the environment that the system tries to achieve.	Goal	An actors' strategic interest
Task	Specification of a piece of behaviour that the system can perform.	Plan/Task	It represents, at an abstract level, a way of doing something.

addition, an excellent summary of the methodology-implicit modeling language/metamodel is given in [22] and therefore not repeated here.

For those papers that discuss the metamodel element of a modeling language, there are a number of approaches that offer an agent-oriented extension to UML [23], [24], [25] and others, like in this paper, that eschew this route, arguing that UML is an inappropriate basis for an AO modeling language [19], [20]. There is no agreement as to whether "agent-oriented concepts can readily be defined in terms of object-oriented ones" ([67, p. 121]) or whether "to properly support agent-based

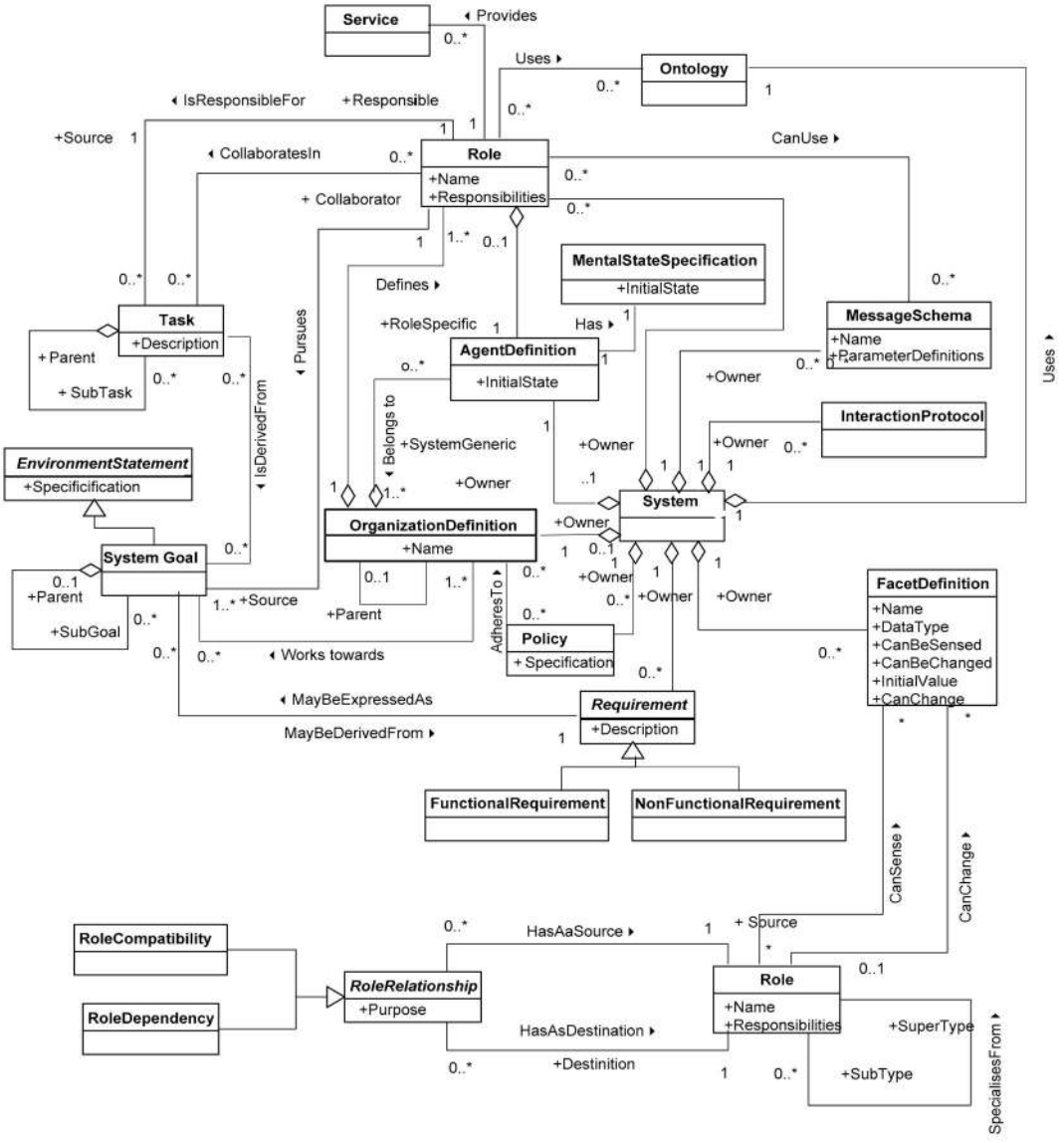

Fig. 12. Design-time agent-external classes. (Note that the duplication of the Role class is only to simplify the layout.)

modeling, it would be necessary to add new concepts and notations in the UML core metamodel" and that "a stereotyped object is still an object" [20]. "The Class metaclass provided by the UML Specification could not be used to define agents" [22].

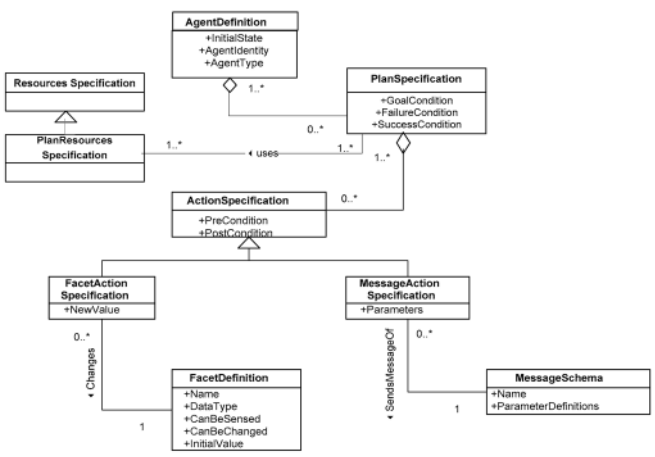

Fig. 13. Design-time agent-internal classes.

Authors who offer a UML profile generally do so for pragmatic reasons. Since UML is well accepted, they argue that this is reason enough to base an agent modeling language on the UML metamodel. Although there are two extension mechanisms offered in the UML documentation (metamodel extension or a UML profile based on stereotypes), only that of a "profile" is encouraged by OMG. The alternative (a "variant") involves extending the metamodel itself, as has been done in the object-oriented context [68], and more recently, the agent-oriented context [22]. These latter authors argue that, while introducing new ideas is easier from a previously understood base [69], the shortcomings of using the profile extension mechanism of UML cannot deal directly with the necessary agent-oriented concepts directly [22]. Consequently, their MAS-ML agent-oriented modeling language [22] introduces agent concepts as additional classes that extend the UML metamodel itself (rather than the indirect extension mechanism of stereotypes

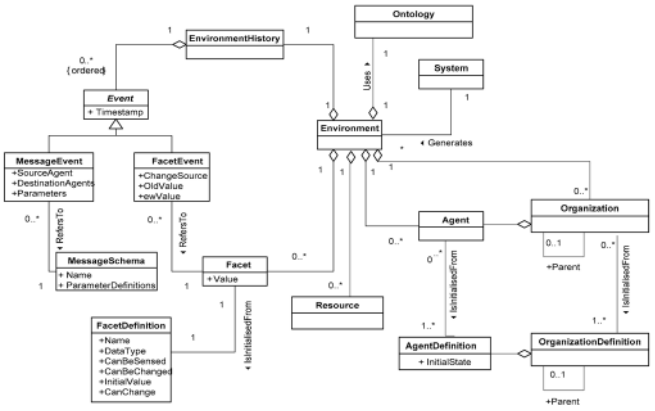

Fig. 14. Runtime agent-external classes.

within a profile). Fig. 16 shows these as five subtypes of UML's Classifier—suggesting a possible future research topic for FAML, perhaps merging its major concepts of Agent, AgentDefinition, Role, Organization, etc., with those of MAS-ML, as shown in Fig. 16.

As well as MAS-ML, which balances original agent-focused classes with the pragmatic advantages of integrating these within the UML metamodeling framework, other independent AO MLs are ANote [20] and the metamodel of the CAMLE approach [70]. ANote has some similarities to FAML but offers less coverage and less detail overall (see Fig. 17, cf., Figs. 12, 13, 14, and 15). CAMLE's metamodel is described only partially in [70] and is focused on the original notion of the "caste"—a set of interactive agents: a concept introduced in [71].

Of comparable scope to FAML is the work of Azaiez et al. [8]. They, too, seek a generic metamodel, having recognized the incompleteness and overspecificity of earlier, particularly methodology-linked, metamodels. Their metamodel (Fig. 18) does not discriminate between design-time and runtime but does partition the conceptual basis between a number of so-called perspectives: environment, agent, action, interaction, and organization. They include a number of concepts purposefully excluded in FAML, such as subtypes of SimulatedEnvironment (itself a subtype of Environment) (their Fig. 2) as well as modeling both reactive and cognitive agents. Nevertheless, their metamodel has many similarities—which we intend to explore both collaboratively and in the context of the recent Request for Proposals for an Agent modeling Language and Profile from the Object Management Group [72].

6 Summary, Discussion, and Future Work

This paper has introduced the application of a process for synthesizing a new metamodel from extant metamodels in a concrete domain. The process included an extensive validation process and should be of interest to researchers in other domains with a diversity of metamodels.

The main conclusion of the synthesis process is the high degree of knowledge of the area required to produce the FAML metamodel and the number of iterations needed to produce something meeting the needs of the researchers. While it is not possible to prove that we have reached completeness in the validation, the decreasing number of new concepts added at each validation stage, with no new concepts identified from the INGENIAS and Tropos validation step, indicates that we are approaching completeness against existing metamodels. However, we intend to continue this validation process against other metamodels—FAML may require modification as new metamodels are

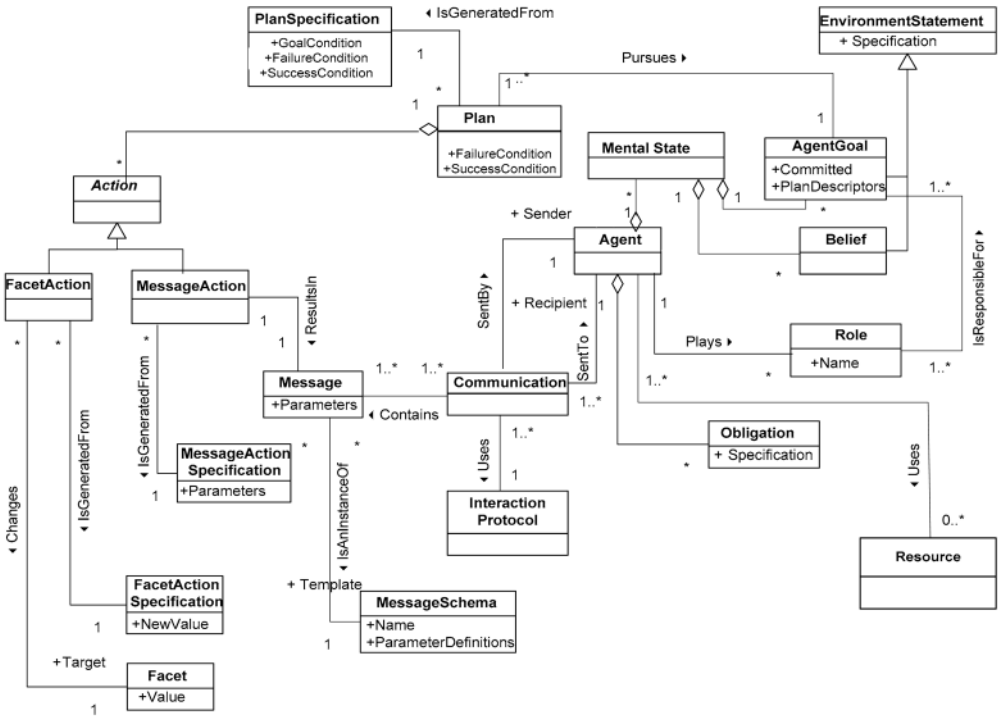

Fig. 15. Runtime agent-internal classes.

TABLE 9
Design-Time Concepts and Their Definitions Used in the FAML Metamodel

Concept	Definition	Added at validation step
Action Specification	Specification of an action, including any preconditions and post-conditions.	
Agent Definition	Specification of the initial state of an agent just after it is created.	
Environment Statement	A Boolean statement about the environment.	
Facet Action Specification	Specification of a facet action in terms of the facet definition it will change and the new value it will write to the facet.	
Facet Definition	Specification of the structure of a given facet, including its name, data type and access mode.	
Functional Requirement	Requirement that provides added value to the users of the system.	
Interaction Protocol	Specification of patterns of communications that occur in the system	2
Mental State Specification	Specification of the initial mental state in terms of specified beliefs and agent goals	3
Message Action Specification	Specification of a message action in terms of the message schema and parameters to use.	
Message Schema	Specification of the structure and semantics of a given kind of messages that can occur within the system.	
Non-Functional Requirement	Requirement about any limits, constraints or impositions on the system to be built.	
Ontology	Structural model of a given domain.	
Concept	**Definition**	**Added at validation step**
Organization Definition	Specification of a collection of roles and agents co-operating towards a system goal.	1
Plan Resource Specification	This is a specification of resources that are used in the Plan Specification.	3
Plan Specification	An organized collection of action specifications.	
Policy	A rule that specifies an arrangement of events expected to occur in a given environment.	
Requirement	Feature that a system must implement.	
Resource Specification	A resource specification specifies something that has a name, may have reasonable representations and that can acquired, shared or produced.	3
Role	Specification of a behavioural pattern expected from some agents in a given system.	
Role Compatibility	Role relationship in which the source role is incompatible with the destination role for a given purpose.	1
Role Dependency	Role relationship in which the source role depends on the destination role for a given purpose.	1
Role Relationship	Social relationship between two roles for a given purpose.	1
Service	A single, coherent block of activity in which an agent may engage.	2
System	Final product of an agent-oriented software development project.	
System Goal	A specification of a state of the environment that the system tries to achieve.	1
Task	Specification of a piece of behaviour that the system can perform.	

considered. Thus, it is an evolving metamodel whose goal is to include as many agent research concepts as possible while keeping internal consistency (among FAML concepts) and external consistency (between FAML and other metamodels). Another concern within each validation step was how different concepts were going to be integrated in the final metamodel. This paper solved this by resolving ambiguities in published definitions by using generality to deal with concepts with a certain degree of contradiction.

Focusing more on FAML, there are other lessons that are worth noting. Taking as reference the last published version of FAML and after the four-step validation, 3 runtime concepts and associated relations are deleted, and 10 design-time, 5 runtime concepts, and associated new relations have been added to FAML as follows:

- Step 1: Preliminary detailed evaluation against the two modeling language metamodels, TAO and Islander, leads to adding five design concepts and one runtime concept. They are, respectively, the following: *SystemGoal*, *RoleRelationship*, *RoleDependency*, *RoleCompatibility*, *OrganizationDefinition* and *Organization*.
- Step 2: Broad cross check against 10 AOSE methodologies leads to adding the following two design concepts and one runtime concept: *Service*, *InteractionProtocol* and *AgentGoal*. This step also leads to deleting three runtime concepts: *Intention*, *Desire*, and *Obligation*.
- Step 3: Detailed check of metamodels of three methodologies, Adelfe, PASSI, and Gaia, leads to adding three design concepts and three runtime concepts. They are, respectively, the following: *ResourceSpecification*, *PlanResourceSpecification*, *MentalStateSpecification* and *Resource*, *Communication*, and *MentalState*.
- The final step successfully generates concepts for the Tropos and INGENIAS metamodels without requiring any further additions to FAML. As anticipated, the rate of adding new concepts dwindles as the validation evolves (see also [73]).

Of the new concepts, we note that a large proportion of the new concepts (7/15) includes container concepts that are formed by composing existing FAML concepts. The composition was deemed important and common enough in the analysis phase of existing methodologies to warrant their addition to FAML. For example, *InteractionProtocol* (a new concept) is a pattern of *Communication* (a new concept), which, in turn, is a sequence of *Messages* (which is an old FAML concept). This suggests that, while we originally succeeded in developing a general model of agent design concepts, we achieved that generality at the expense of

TABLE 10
FAML Runtime Concepts and Their Definitions

Concept	Definition	Added at validation step
Action	Fundamental unit of agent behaviour.	
Agent	A highly autonomous, situated, directed and rational entity.	
Agent Goal	An environment statement which represents a state pursued by an agent	2
Belief	An environment statement held by an agent and deemed as true in a certain timeframe.	
Communication	Composition of more than one message.	3
Environment	The world in which an agent is situated.	
Environment History	The sequence of events that have occurred between the environment start-up and any given instant.	
Environment Statement	A statement about the environment.	
Event	Occurrence of something that changes the environment history.	
Facet	Property of the environment with which agents can interact.	
Facet Action	Action that results in the change of a given facet.	
Facet Event	Event that happens when the value of a facet changes.	
Mental State	Agent goals and beliefs held by an agent at a certain timeframe.	3
Message	Unit of communication between agents, which conforms to a specific message schema.	
Message Action	Action that results in a message being sent.	
Message Event	Event that happens when a message is sent.	
Organization	A collection of agents with specified roles co-operating towards a system goal.	1
Plan	An organized collection of actions that can be executed to pursue a particular agent goal.	
Resource	Something that has a name, may have reasonable representations and that can acquired, shared or produced.	3
Role	Specification of a behavioural pattern expected from some agents in a given system.	
System	Final product of an agent-oriented software development project	

omitting intermediate abstractions. Our validation and the new concepts have fixed this deficiency.

Another proportion of the new concepts (5/15) is added to describe features of classes of some modern and well-known MASs. The new FAML concepts describe services and nonagent resources. The fact that we originally omitted these important features suggests that, while we aimed for having the system-level concepts capture the commonalities between all methodologies, the coverage suffered. Our validation and the new concepts have rectified this deficiency. A remaining small proportion of the new concepts (3/15) is added to refine our role relationships in a way that is common to many MAS reflecting hierarchical human organizations: *RoleRelationship, RoleDependency, and RoleCompatibility.*

In addition, it is illustrative to analyze the result of the validation of FAML, as reflected in the number of new concepts to each of its four metalevels: system at design-time, agent at design-time, system at runtime, and agent at runtime. The validation produced most changes to the design-time agent-external classes metalevel of FAML, adding eight new concepts. In comparison, the validation added two or three new concepts to each of the other metalevels of FAML. Almost half of the new concepts are added to one metalevel. This is most likely due to two factors: First, the focus of the current agent-oriented methodologies is on analysis and design of MASs. Second, there is broad agreement within the AOSE community as to what an agent is capable of doing, since AOSE builds on the view of single agent from AI.

With the extensions to FAML, we expect that its current version is capable of representing metamodels of most existing MAS methodologies. This is evidenced by our successful generation of the metamodels of two important methodologies, Tropos and INGENIAS, without any alterations to FAML. This is an important contribution in developing an agent-oriented metamodel capable of supporting the workproducts produced or consumed in a software development endeavor and facilitating a

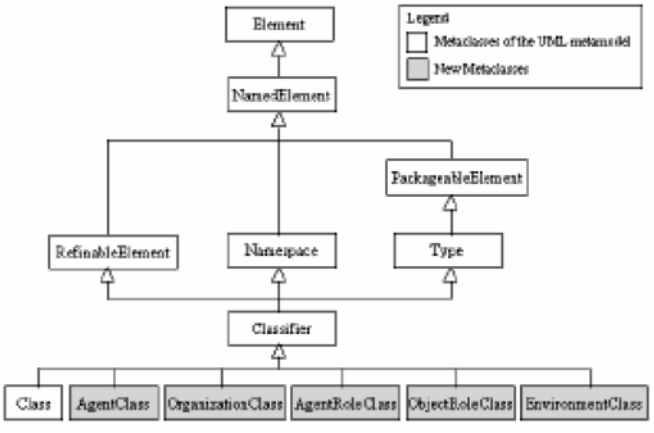

Fig. 16. MAS-ML's metamodel showing five new metaclasses to be added to the UML metamodel (after [22]).

Fig. 17. Conceptual metamodel for ANote (after [20]).

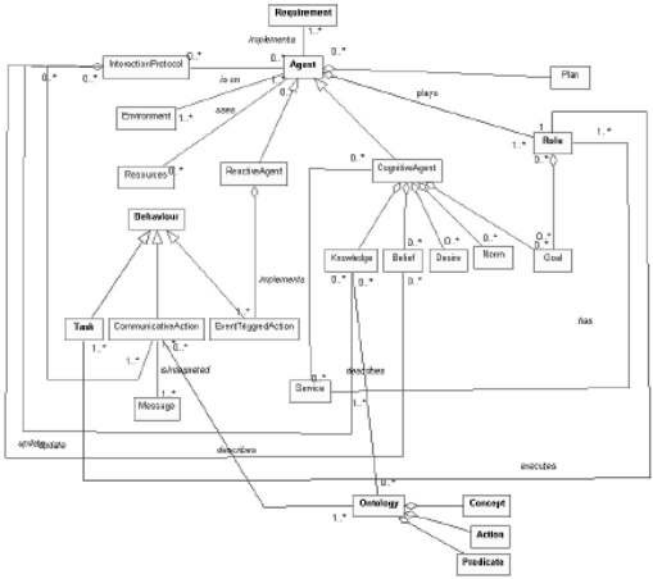

Fig. 18. Final metamodel for the agent perspective proposed by Azaiez et al. (from [8]).

model-driven architecture for AOSE. In a parallel effort, to appear in [29], we are adding security concepts to FAML that are not covered by the current validation, being not sufficiently "generic." As well as extending the current validation for our security concepts, our future work will extend FAML to cover mobile agents and associated features. The final phase of this project will create a notation (concrete syntax) to complement this FAML metamodel (abstract syntax).

REFERENCES

[1] C. Chavez and C. Lucena, "A Metamodel for Aspect-Oriented Modeling," *Proc. Aspect Oriented Modeling with UML Workshop,* 2002.

[2] T. Cottenier, A.v.d. Berg, and T. Elrad, "Modeling Aspect-Oriented Compositions," *Proc. Int'l Conf. Model Driven Eng. Languages and Systems Satellite Events,* pp. 100-109, 2005.

[3] L. Fuentes and P. Sanchez, "A Generic MOF Metamodel for Aspect-Oriented Modelling," *Proc. Workshop Model-Based Development of Computer-Based Systems and Model-Based Methodologies for Pervasive and Embedded Software,* pp. 10-30, 2006.

[4] P. Kruchten, "Software Architecture—a Rational Metamodel," *Proc. Second Int'l Software Architecture Workshop and Int'l Workshop Multiple Perspectives in Software Development on SIGSOFT '96 Workshops,* pp. 5-7, 1996.

[5] M.H. Kacem, A.H. Kacem, M. Jmaiel, and K. Drira, "Describing Dynamic Software Architectures Using an Extended UML Model," *Proc. 2006 ACM Symp. Applied Computing,* 2006.

[6] O. Topçu, M. Adak, and H. Oguztüzün, "A Metamodel for Federation Architectures," *ACM Trans. Modeling and Computer Simulation,* vol. 18, no. 3, pp. 1-29, 2008.

[7] C. Bernon, M. Cossentini, and J. Pavon, "Agent-Oriented Software Engineering," *The Knowledge Eng. Rev.,* vol. 20, no. 2, pp. 99-116, 2005.

[8] S. Azaiez, M.P. Huget, and F. Oquendo, "An Approach for Multi-Agent Metamodelling," *Multiagent and Grid Systems,* vol. 2, no. 4, pp. 435-454, 2006.

[9] J.A. Rodriguez, "On the Design and Construction of Agent-Mediated Electronic Institutions," Artificial Intelligence Research Inst., UAB—Universitat Autonòma de Barcelona, 2003.

[10] G. Tidhar, C. Heinze, S. Goss, G. Murray, D. Appla, and I. Lloyd, "Using Intelligent Agents in Military Simulation or "Using Agents Intelligently"," *Proc. 11th Conf. Innovative Applications of Artificial Intelligence Papers,* pp. 829-837, 1999.

[11] B. Henderson-Sellers and P. Giorgini, *Agent-Oriented Methodologies.* Idea Group Publishing, 2005.

[12] M. Wooldridge, N.R. Jennings, and D. Kinny, "The Gaia Methodology for Agent-Oriented Analysis and Design," *Autonomous Agents and Multi-Agent Systems,* vol. 3, no. 3, pp. 285-312, 2000.

[13] C. Bernon, M.-P. Gleizes, S. Peyruqueou, and G. Picard, "ADELFE, a Methodology for Adaptive Multi-Agent Systems Engineering," *Eng. Societies in the Agents World,* Springer, 2002.

[14] L. Padgham and M. Winikoff, *Developing Intelligent Agent Systems: A Practical Guide,* vol. 1. J. Wiley & Sons, 2004.

[15] J. Pavon, J. Gomez-Sanz, and R. Fuentest, "The INGENIAS Methodology and Tools," *Agent-Oriented Methodologies,* B. Henderson-Sellers and P. Giorgini, eds., pp. 236-276, Idea Group Publishing, 2005.

[16] M. Cossentino and C. Potts, "A CASE Tool Supported Methodology for the Design of Multi-Agent Systems," *Proc. Int'l Conf. Software Eng. Research and Practice,* 2002.

[17] P. Bresciani, P. Giorgini, F. Giunchiglia, J. Mylopoulos, and A. Perini, "TROPOS: An Agent Oriented Software Development Methodology," *J. Autonomous Agents and Multi-Agent Systems,* vol. 8, no. 3, pp. 203-236, 2004.

[18] G. Beydoun, C. Gonzalez-Perez, B. Henderson-Sellers, and G.C. Low, "Developing and Evaluating a Generic Metamodel for MAS Work Products," *Software Eng. for Multi-Agent Systems IV: Research Issues and Practical Applications,* A. Garcia, R. Choren, C. Lucena, P. Giorgini, T. Holvoet, and A. Romanovsky, eds., pp. 126-142, Springer-Verlag, 2006.

[19] V.T. da Silva, A. Garcia, A. Brandão, C. Chavez, C. Lucena, and P. Alencar, "Taming Agents and Objects in Software Engineering," *Software Engineering for Large-Scale Multi-Agent Systems,* pp. 1-26, Springer-Verlag, 2003.

[20] R. Choren and C. Lucena, "The ANote Modeling Language for Agent Oriented Specification," *Proc. Int'l Workshop Software Eng. for Large Scale Multi Agent Systems 2004,* pp. 198-212, 2005.

[21] D. Dori, "What UML Should Be: Why Significant UML Change Is Unlikely," *Comm. ACM,* vol. 45, no. 11, pp. 82-85, 2002.

[22] V.T. da Silva, R. Choren, and C. Lucena, "MAS-ML: A Multi-Agent System Modeling Language," *Int'l J. Agent-Oriented Software Eng.,* vol. 2, no. 4, pp. 381-421, 2008.

[23] J. Odell, H. Van Dyke Parunak, and B. Bauer, "Extending UML for Agents," *Proc. Agent-Oriented Information Systems Workshop, 17th Nat'l Conf. Artificial Intelligence,* pp. 3-17, 2000.

[24] B. Bauer, J.P. Muller, and J. Odell, "Agent UML: A Formalism for Specifying Multiagent Interaction," *Agent-Oriented Software Eng.,* P. Ciancarini and M. Wooldridge, eds., pp. 91-103, Springer, 2001.

[25] R. Cervenka and I. Trencansky, *AML: The Agent Modeling Language. A Comprehensive Approach to Modeling Multi-Agent Systems.* Birkhäuser, 2007.

[26] V.T.D. Silva and C.J.P.D. Lucena, "From a Conceptual Framework for Agents and Objects to a Multi-Agent System Modeling Language," *Autonomous Agents and Multi-Agent Systems,* vol. 9, nos. 1/2, pp. 145-189, 2004.

[27] J. Odell, "Metamodelling: Future Work," personal comm., 2008.

[28] V. Torres da Silva, R. Choren, and C.J.P. De Lucena, "MAS-ML: A Multi-Agent System Modeling Language," *Int'l J. Agent-Oriented Software Eng.,* vol. 2, no. 4, pp. 381-421, 2008.

[29] G. Beydoun, G. Low, H. Mouratidis, and B. Henderson-Sellers, "A Security-Aware Metamodel for Multi-Agent Systems," *Information and Software Technology,* 2008, doi:10.1016/j.infsof.2008.05.003.

[30] P. Verstraete, B.S. Germain, P. Valckenaers, H.V. Brussel, J. Belle, and H. Hadeli, "Engineering Manufacturing Control Systems Using PROSA and Delegate MAS," *Int'l J. Agent-Oriented Software Eng.,* vol. 2, no. 1, pp. 62-89, 2008.

[31] T. Mine, D. Matsuno, A. Kogo, and M. Amamiya, "Design and Implementation of Agent Community Based Peer-to-Peer Information Retrieval Method," *Proc. Workshop Cooperative Information Agents,* pp. 31-46, 2004.

[32] J. Odell, M. Nodine, and R. Levy, "A Metamodel for Agents, Roles, and Groups," *Proc. Int'l Workshop Agent-Oriented Software Eng.,* J. Odell et al., eds., pp. 78-92, 2005.

[33] P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, and J. Mylopoulos, "A Knowledge Level Software Engineering Methodology for Agent Oriented Programming," *Proc. Int'l Conf. Autonomous Agents,* 2001.

[34] M. Wooldridge, *Reasoning about Rational Agents.* MIT Press, 2000.

[35] G.F. Luger, *Artificial Intelligence: Structures and Strategies for Complex Problem Solving.* Addison Wesley, 2002.

[36] S. Russell and P. Norvig, *Artificial Intelligence, A Modern Approach.* Prentice Hall, 2002.
[37] J. Ferber and A. Drogoul, "Using Reactive Multi-Agent Systems in Simulation and Problem Solving," *Distributed Artificial Intelligence: Theory and Praxis,* L.G.N.M. Avouris, ed., Kluwer, 1992.
[38] E. Durfee and V. Lesser, "Negotiating Task Decomposition and Allocation Using Partial Global Planning," *Distributed Artificial Intelligence,* L. Gasser and M. Huhns, eds., vol. 2, pp. 229-244, Morgan Kaufmann, 1989.
[39] R. Pfeifer and C. Sheier, *Understanding Intelligence.* MIT Press, 2001.
[40] FIPA, *Methodology Glossary—FIPAMG,* http://www.pa.icar.cnr.it/~cossentino/FIPAmeth/glossary.htm, 2003.
[41] S. DeLoach, L. Padgham, A. Perini, A. Susi, and J. Thangarajah, "Using Three AOSE Toolkits to Develop a Sample Design," *Int'l J. Agent-Oriented Software Eng.,* vol. 3, 2009.
[42] J.J. Odell, H.V.D. Parunak, M. Fleischer, and S. Brueckner, "Modeling Agents and Their Environment," *Proc. Int'l Workshop Agent-Oriented Software Eng.,* F.G. et. al., eds., pp. 16-31, 2003.
[43] E. Platon, M. Mamei, N. Sabouret, S. Honiden, and H.V.D. Parunak, "Mechanisms for Environments in MAS: Survey and Opportunities," *J. Autonomous Agents and Multi-Agent Systems,* vol. 14, no. 1, pp. 31-47, 2007.
[44] P. Valckenaers, J. Sauter, C. Sierra, and J.A. Rodriguez, "Applications and Environments for MAS," *J. Autonomous Agents and Multi-Agent Systems,* vol. 14, no. 1, pp. 61-85.
[45] G. Beydoun, A. Hoffmann, J.T.F. Breis, R. Martinez-Béjar, R. Valencia-Garcia, and A. Aurum, "Cooperative Modeling Evaluated," *Int'l J. Cooperative Information Systems, World Scientific,* vol. 14, no. 1, pp. 45-71, 2005.
[46] V.A. da Silva, C. Ricardo, and C.J.P. Lucena, "Using the MAS-ML to Model a Multi-Agent System," *Proc. Workshop Software Eng. for Multi-Agent Systems,* 2003.
[47] M. Esteva, D. de la Cruz, and C. Sierra, "ISLANDER: An Electronic Institutions Editor," *Proc. Int'l Conf. Autonomous Agents and Multiagent Systems,* pp. 1045-1052, 2002.
[48] Q.N.N. Tran and G.C. Low, "Comparison of Ten Agent-Oriented Methodologies," *Agent-Oriented Methodologies,* B. Henderson-Sellers and P. Giorgini, eds., pp. 341-367, Idea Group Publishing, 2005.
[49] G.C. Low, G. Beydoun, B. Henderson Sellers, and C. Gonzalez-Perez, "Towards Method Engineering for Multi-Agent Systems: A Validation of a Generic Meta-Model," *Proc. Pacific Rim Int'l Conf. Multi-Agents,* 2007.
[50] M. Wooldridge, N.R. Jennings, and F. Zambonelli, "Multi-Agent Systems as Computational Organizations: The Gaia Methodology," *Agent-Oriented Methodologies,* B. Henderson-Sellers and P. Giorgini, eds., pp. 136-171, Idea Group Publishing, 2005.
[51] M. Cossentino, "From Requirements to Code with the PASSI Methodology," *Agent-Oriented Methodologies,* B. Henderson-Sellers and P. Giorgini, eds., pp. 79-106, Idea Group Publishing, 2005.
[52] C. Bernon, V. Camps, M.-P. Gleizes, and G. Picard, "Engineering Adaptive Multi-Agent Systems: The ADELFE Methodology," *Agent-Oriented Methodologies,* B. Henderson-Sellers and P. Giorgini, eds., pp. 172-202, Idea Group Publishing, 2005.
[53] C. Sierra, "Communication about Islander Metamodel," personal comm., 2005.
[54] E. Yu, "Modelling Strategic Relationships for Process Reengineering," Dept. of Computer Science, Univ. of Toronto, 1995.
[55] O. Shehory and A. Sturm, "Evaluation of Modeling Techniques for Agent-Based Systems," *Proc. Fifth Int'l Conf. Autonomous Agents,* pp. 624-631, 2001.
[56] S.A. O 'Malley and S.A. DeLoach, "Determining When to Use an Agent-Oriented Software Engineering Paradigm," *Proc. Second Int'l Workshop Agent-Oriented Software Eng.,* 2001.
[57] L. Cernuzzi and G. Rossi, "On the Evaluation of Agent-Oriented Modeling Methods," *Proc. Object Oriented Programming, Systems, Languages and Applications Workshop Agent-Oriented Methodologies,* 2002.
[58] A. Sabas, M. Badri, and S. Delisle, "A Multidimensional Framework for the Evaluation of Multiagent System Methodologies," *Proc. Sixth World Multiconf. Systemics, Cybernetics and Informatics,* pp. 211-216, 2002.
[59] B. Henderson-Sellers, N. Tran, and J. Debenham, "An Etymological and Metamodel-Based Evaluation of the Terms 'Goals and Tasks' in Agent-Oriented Methodologies," *J. Object Technology,* vol. 4, no. 2, pp. 131-150, 2005.
[60] C. Bernon, M. Cossentino, M. Gleizes, P. Turci, and F. Zambonelli, "A Study of Some Multi-Agent Meta-Models," *Proc. Fifth Int'l Workshop Agent-Oriented Software Eng.,* J. Odell, P. Giorgini, and J. Müller, eds., pp. 62-77, 2005.
[61] M. Wooldridge, N.R. Jennings, and D. Kinny, "A Methodology for Agent-Oriented Analysis and Design," *Proc. Third Int'l Conf. Autonomous Agents,* pp. 69-76, 1999.
[62] F. Zambonelli, N. Jennings, and M. Wooldridge, "Developing Multiagent Systems: The Gaia Methodology," *ACM Trans. Software Eng. and Methodology,* vol. 12, no. 3, pp. 417-470, July 2003.
[63] D. Bertolini, A. Perini, A. Susi, and H. Mouratidis, "The Tropos Visual Language. A MOF 1.4 Compliant Metamodel," AgentLlink III AOSE TFG 2, 2005.
[64] M. Winikoff, "JACK (TM) Intelligent Agents: An Industrial Strength Platform," *Multi-Agent Programming: Languages, Platforms and Applications,* R.H. Bordini, M. Dastani, J. Dix, and A. El Fallah, eds., pp. 175-193, Springer, 2005.
[65] F. Bellifemine, F. Bergenti, G. Caire, and A. Poggi, "JADE—A Java Agent Development Framework," *Multi-agent Programming: Languages, Platforms and Applications,* R.H. Bordini, ed., pp. 125-147, Springer, 2006.
[66] A. Fuxman, L. Liu, J. Mylopoulos, M. Pistore, M. Roveri, and P. Traverso, "Specifying and Analyzing Early Requirements in Tropos," *Requirements Eng.,* vol. 9, no. 2, pp. 132-150, 2004.
[67] G. Caire, W. Coulier, F. Garijo, J. Gomez, J. Pavon, F. Leal, P. Chainho, P. Kearney, J. Stark, R. Evans, and P. Massonet, "Agent Oriented Analysis Using Message/UML," *Agent-Oriented Software Engineering II,* M.J. Wooldridge, G. Weiß, and P. Ciancarini, eds., pp. 119-135, Springer-Verlag, 2002.
[68] B. Henderson-Sellers, C. Atkinson, and D.G. Firesmith, "Viewing the OML as a Variant of the UML," *UML '99—The Unified Modeling Language. Beyond the Standard,* R. France and B. Rumpe, eds., pp. 49-66, Springer-Verlag, 1999.
[69] F. Zambonelli and A. Omicini, "Challenges and Research Directions in Agent-Oriented Software Engineering," *Autonomous Agents and Multiagent Systems,* vol. 9, no. 3, pp. 253-284, 2004.
[70] H. Zhu and L. Shan, "Agent-Oriented Modelling and Specification of Web Services," *Proc. 10th IEEE Int'l Workshop Object-Oriented Real-Time Dependable Systems (WORDS '05),* 2005.
[71] H. Zhu, "SLABS: A Formal Specification Language for Agent-Based Systems," *Int'l J. Software Eng. and Knowledge Eng.,* vol. 11, no. 5, pp. 529-558, 2001.
[72] OMG "Agent Metamodel and Profile (AMP): Request for Proposal," 2008.
[73] B. Henderson-Sellers, "Evaluating the Feasibility of Method Engineering for the Creation of Agent-Oriented Methodologies," *Proc. Fourth Int'l Central and Eastern European Conf. Multi-Agent Systems (CEEMAS '05),* pp. 142-152, 2005.

Ghassan Beydoun received the degree in computer science and the PhD degree in knowledge systems from the University of New South Wales. He is a senior lecturer at the School of Information Systems and Technology at the University of Wollongong and an adjunct senior research fellow at the School of Information Systems, Management and Technology at the University of New South Wales. He has authored 12 international journal papers and 45 conference papers and is currently working on a project sponsored by an Australian Research Council Discovery Grant to investigate the best uses of ontologies in developing methodologies for Multiagent Systems (MASs) along with Graham Low and Brian Henderson-Sellers. His other research interests include MAS applications, ontologies, and knowledge acquisition.

Graham Low received the BE and PhD degrees from The University of Queensland. He is a professor of information systems in the School of Information Systems, Technology and Management at the University of New South Wales. His research program focuses on the implementation and adoption of new technologies. This can take the form of new/modified approaches/techniques for information systems development such as methodological approaches to agent-oriented information systems design; and management of the information systems design and implementation process.

Brian Henderson-Sellers received the doctor of science (DSc) degree from the University of London in 2001 for his research contributions in object-oriented methodologies. He is the director of the Centre for Object Technology Applications and Research and a professor of information systems at the University of Technology, Sydney (UTS). He is the author or editor of 31 books.

Haralambos Mouratidis is a principal lecturer in the School of Computing, IT and Engineering at the University of East London, where he is also the field leader for the Secure Systems and Software Development Field. His research interests are related to secure software engineering and agent-oriented software engineering.

Jorge J. Gomez-Sanz received the degree in software engineering and the PhD degree in computer science from the Universidad Complutense de Madrid (UCM), Spain. He is a professor at the UCM and participates in industrial technology transfer projects. His research focuses on developing multiagent systems following software engineering practices.

Juan Pavón has been an associate professor at the Universidad Complutense of Madrid since 1997, where he leads the Grasia Research Group, with focus on the application of multiagent systems technology.

Cesar Gonzalez-Perez is an assistant professor in the Heritage Laboratory (LaPa) at the Spanish National Research Council (CSIC), where he leads a research line on software engineering applied to cultural heritage issues. His research interests include conceptual modeling, metamodeling, and method engineering.