Hindawi Publishing Corporation
ISRN Software Engineering
Volume 2013, Article ID 803638, 53 pages
http://dx.doi.org/10.1155/2013/803638

Review Article

Towards the Consolidation of a Diagramming Suite for Agent-Oriented Modelling Languages

Brian Henderson-Sellers

University of Technology, Sydney, Broadway, NSW 2007, Australia

Correspondence should be addressed to Brian Henderson-Sellers; brian.henderson-sellers@uts.edu.au

Received 21 October 2012; Accepted 7 November 2012

Academic Editors: X. He, S. Sutton, and M. Viroli

Whilst several agent-oriented modelling languages have been developed by independent research groups, it is now appropriate to consider a consolidation of these various approaches. There are arguably three things that need consolidation and future standardization: individual symbols, the underpinning metamodel, and the diagram types. Here we address only the third issue by extending an earlier analysis that resulted in recommendations for various diagram types for the modelling of a multiagent system (MAS). Here, we take each of these previously recommended diagram types and see how each is realized in a wide variety (over 20) of current agent-oriented software engineering (AOSE) methodologies. We also take the opportunity to express, as exemplars, some of these diagram types using the recently published FAML notation.

1. Introduction

Any software development benefits from the use of a methodology. Part of such a methodological approach is a means to depict interim work products, typically documented using a graphical notation (a.k.a. concrete syntax). Symbols are used to represent single concepts, defined in an appropriate modelling language (ML), itself typically represented, at least in part, by a metamodel (e.g., [1]). These symbols can then be grouped in heterogeneous yet semantically related ways. A coherent model, thus depicted, is often said to be of a specific *diagram type*. In other words, a diagram type refers to a collection of classes in the metamodel, that is, it defines which metaclasses can be appropriately instantiated for this particular scope and focus.

For agent-oriented software engineering (AOSE), such modelling languages (and their notations and recommended diagram types) are in their infancy. A large number of AOSE methodological approaches exist, all with their own notational elements. As part of a community goal of standardizing agent-oriented modelling languages, collaborative notations have been proposed (e.g., [2]), as well as mergers at the conceptual level (e.g., [3, 4]), the latter of these being complemented more recently by a concrete syntax [5].

Notations need to have a high degree of usability, which can often be accomplished based on semiotic principles (e.g., [6–8]). Information is needed not only about individual agents and interagent communications, but also on the context of the environment in which they are situated. Current practice in many methodological approaches is to utilize standard object-oriented diagramming techniques, typically using UML [9–11] as a notation, whenever possible, although there are many concepts in AOSE not so representable. For example, Garcia et al. [12] comment on the need to include specific agenthood properties, including interaction/communication, autonomy, and adaptation with possible additional properties of learning, mobility, collaboration, and roles. A similar list, yet with a BDI (BDI = beliefs, desires, and intentions (e.g., [13, 14])) slant, is given by Sturm and Shehory [15, 16] as agent, belief, desire, intention, message, norm, organization, protocol, role, society, and task. Taveter and Wagner [17] identify the most important concepts as including agents, events, actions, communication, and message, underpinning these in terms of ontological theory (e.g., endurants and perdurants). Bertolini et al. [18] focus primarily on the Goal Diagram and the Actor Diagram in their presentation of TAOM4E—an Eclipse-based tool to support the Tropos methodology and based solidly on a metamodel.

Beydoun et al. [4] present a generic metamodel which itself contains four connected perspectives. In this case, the discrimination is between organization as compared to agent level and between design time and run time. However, they do not explicitly link these to diagram types, although there is in fact a weak relationship.

Diagram types are often divided into two loosely defined groups: static or structural diagrams and dynamic (a.k.a. behavioural) diagrams (e.g., [69, 70])—a grouping that will also be utilized here. The former depict aspects that might be termed architectural, typified by variants of an OO class diagram; the latter depict some forms of functionality and time-dependent actions.

Torres da Silva et al. [71] have presented MAS-ML as a metamodel-based modelling language for agent-oriented software engineering. As well as introducing new agent-focussed concepts, as discussed below, they also recommend a suite of diagram types—three static and two dynamic:

(i) Extended UML Class Diagram,

(ii) Organization Diagram,

(iii) Role Diagram,

(iv) Extended UML Sequence Diagram,

(v) Extended UML Activity Diagram.

In contrast to the approach taken in the ML proposed by Beydoun et al. [4] that focusses first on a viewpoint and later on the detailed concepts, Torres da Silva et al. [71] propose not viewpoints but specific diagram types, although they neglect to give a clear problem statement for which these diagram types are the proposed solution. In other words, whilst useful, they are at the diagram level rather than the viewpoint level as advocated in Henderson-Sellers [19]. We will therefore comment on each of these diagrams in the appropriate place in Sections 5 and 6.

In summary, we aim here to make a contribution towards future standardization of agent-oriented modelling languages—focussing here on diagram suites. Section 2 outlines the approach taken in determining an appropriate framework, which we then use to analyze over 20 contemporary agent-oriented methodologies in terms of the kinds of diagrams that they support and recommend. Section 3 discusses notational aspects, introducing the FAML notation [5] that we use in later examples in comparison with the notations used by these individual AOSE methodologies. Following an overview of diagram types in Section 4, in the next two sections, we describe in detail static diagram types (Section 5) and then dynamic diagram types (Section 6). In each of these two sections, we categorize diagrams using the several views derived in Section 2. Section 7 provides a final discussion and indicates some other related work not otherwise cited followed by a brief conclusion section (Section 8) including some ideas for future research. From this detailed comparison, we aim to draw out commonalities and variations in the suite of diagram types utilized across all extant agent-modelling languages as a precursor to future international standardization.

Table 1: Set of diagram types recommended in Henderson-Sellers [19] in the light of his analysis of AOSE methodologies. These diagram types should then be supplemented by textual based templates and descriptors as shown in Table 2.

Static diagram types	Dynamic diagram types
Environment description	Agent goal-based use case
Environmental connectivity	Use case map
External organization structure chart	Conversation a.k.a. interaction
Architecture	Protocol (a kind of conversation)
Agent society	Workflow
Agent role	Agent state
Role dependency	Task specification
Agent internals	Task state
Agent overview	
Goal decomposition	
Ontology	
Plan	
Capability	
Service	
Task decomposition	
Deployment	
UI design	

Table 2: Textual work products (static and dynamic).

Static textual diagram types	Dynamic textual diagram types
System requirements	Goal-based use case template
Role definition template	Contractual template
Agent descriptors	Event descriptors
CRC cards	Data descriptors
Plan descriptor	Plan descriptors
Capability descriptors	Task template
Service diagram	Protocol descriptors
Task template	Message descriptors
Percept descriptor	Action descriptor
	Process descriptor

2. Research Approach

As detailed in Henderson-Sellers [19], in order to analyze the various options for a suite of AOSE relevant diagrams, the first step was to identify static versus dynamic diagram types (Tables 1 and 2) and then to group these in terms of their relevance to a number of views or viewpoints as previously discussed in the AOSE literature (e.g., [20, 48]). Seven such views were identified (Table 3), and, for each, both static and dynamic diagram types were identified (Table 4). (Details of the several iterations needed to derive Table 4 are to be found by Henderson-Sellers [19] and are not replicated here.) Finally, the atomic elements identified for each of these diagram types are listed in Table 5. However, this list is not absolute in that different methodologies offer different interpretations and consequently use different atomic elements on

Table 3: Seven views recommended in the analysis of Henderson-Sellers [19]. Note that the original analysis was based on the AOSE literature which essentially eschews aspects of user interface. To these seven, an eighth one, UI, needs to be added (reprinted from [19], copyright 2010, with permission from IOS Press).

View name	Focus of view
Environment	External context, including system requirements
Architecture	High level structure of system independent of agent technology
Agent societies	Structure of agents into groups together with interactions and information exchange, typically within the group
Agent workflow	Workflows
Agent knowledge	Roles of individual agents, their responsibilities, and purpose
Agent services	Services offered, tasks to be undertaken, goals to be attained, and detailed capabilities. Applied to a small number of interacting agents
Deployment	Interface with run-time platform

Table 4: Two dimensional matrix for views versus static/dynamic aspects for various AOSE diagram types (modified and reprinted from [19], copyright 2010, with permission from IOS Press).

View	Static diagram types	Dynamic diagram types
Environment	Environment description; environmental connectivity; system requirements; use case	N/A
Architecture	Agent societies/organization	N/A
Agent societies	Agent society details; agent role	Conversation (including interaction and protocol); task
Agent workflow	N/A	Workflow
Agent knowledge	Goal; agent type; agent role; plan; ontology	Goal; agent state; capability
Agent services	Agent society details; agent type; goal; ontology	Goal; task; capability
Deployment	Allocation to run-time platform	N/A
UI	User interface design	States and transitions related to interface

any one named diagram—for example, Padgham et al. [2] note that in the Prometheus methodology an Agent Society model shows actions and percepts but would not use an Ontology diagram, whereas users of the PASSI methodology would use a separate Ontology diagram.

As the knowledge of AOSE increases, the diagram suite suggested in Table 4 and the details of Table 5 will almost certainly require further changes—this paper offers further comments based on further investigation of the extant literature.

An initial assessment [19] resulted in some suggested recommendations for each diagram type in Table 4. Here, we commence with those recommendations and evaluate how each particular diagram type is utilized in methodologies not previously discussed. With the recent advent of a proposed notational standard for FAML [5], we take the opportunity of including an evaluation of how the symbols in this modelling language (summarized in Figure 2) can be useful. In cases where problems are identifiable, this could lead to improvements to be proposed to the FAML notation itself.

3. Notations

Notations (a.k.a. concrete syntax) currently utilized for agent-oriented methodologies are typically individualistic. However, there are efforts under way to systematize these. Two proposed notations, AML [72, 73] and AUML [74, 75], are essentially extensions of an object-oriented modelling language—whether this is appropriate is discussed in, for example, Torres da Silva and de Lucena [76], Choren and Lucena [77], and Beydoun et al. [4].

In AML, UML class diagrams are used with subtypes of Ontology Diagrams, Society Diagrams, Behavior Decomposition Diagrams, and Mental Diagrams (with a further subtype of Goal-Based Requirements Diagram). Composite Structure Diagrams (from UML) can be either Entity or Service Diagrams in AML; UML sequence diagrams are used as Protocol Sequence Diagrams with a subtype of Service Protocol Sequence Diagram. Finally, UML communication diagrams are realized as Protocol Communication Diagrams, a subtype of which is the Service Protocol Communication Diagram.

These UML-based notations are not readily related to the seven views identified by Henderson-Sellers [19] (see Table 3), although they do discuss static versus dynamic aspects of each diagram (Table 1).

Secondly, a number of methodologies use as their main notation that of i^* [78] (later mapped in the agent-oriented context to UML by Mylopoulos et al. [79]). Designed for requirements engineering, i^*'s usage in AOSE has been primarily in the requirements and architectural design stages of Tropos (in later stages Tropos uses AUML/UML diagrams)

Table 5: Atomic elements and diagram types.

Diagram type	Atomic elements to be displayed
(1) *Static diagram types*	
Environment description	Entities represented by classes; relationships between the modelled entities
Environmental connectivity	Agents/MASs, internal and external resources, relationships across the MAS/environment interface
External organization structure chart	Organizational units in the real-life business
Architecture	Technology-independent large-scale structure
Agent society	Agents inside the MAS, how they associate with each other
Agent role	Links between the agents and the roles they play
Role dependency	Hierarchical structure of many roles
Agent internals	Constituent elements in an individual agent or role
Agent overview	High level view of an agent
Goal decomposition	Goals, subgoals
Ontology	The underpinning semantic structure
Plan	The (process) steps needed to effect a task and accomplish a goal
Capability	The ability or responsibility of an agent
Service	Functionality offered by the agent
Task decomposition	Tasks, subtasks
Deployment	Allocation of MAS elements to nodes of the run-time platform
UI design	TBD (the topic of proposed future research). (See brief discussion in Sections 5.7 and 6.5 on the relevant, non-AOSE UI literature)
(2) *Dynamic diagram types*	
Agent goal-based use case	Functionality offered by the MAS
Use case map	Threads across many agents to realize a use case
Conversation	Dynamic interaction details
Protocol	Rules associated with interactions
Workflow	Large-scale processes relating to problem solving (in the real world)
Agent state	Attribute values determining the current state of an agent
Task specification	Definitions of tasks needed to accomplish a specific goal
Task state	The current state of a task, in terms of how far through the task enactment

because that agent-oriented methodology uses requirements engineering concepts throughout the development process. However, more recently this notation has been more widely evaluated. For example, Lapouchnian and Lespérance [80] map between i^* and CASL (Cognitive Agents Specification Language [81]) representing agents' goals and knowledge as mental states; Franch [82] assesses the predictability of i^* models; Estrada et al. [83] undertake an empirical evaluation of i^* using industrial case studies and conclude that extensions and modifications are needed for i^* to address its lack of modularization.

Although most methodologists devise their own notation, there has been over the last few years a groundswell of opinion that notations (and metamodels) should be applicable to more than just a single methodological approach. In that spirit, Padgham et al. [2] suggest a notation based on a merger between the notations that are part of O-MaSE, Tropos, Prometheus, and PASSI. (Sources/citations for the various AOSE methodologies are found in Table 7). Although a huge step forward in the future creation of a widely acceptable standard AOML, Henderson-Sellers et al. [5] offered some areas for improvement, based on semiotic considerations. Using that experience (of Padgham et al. [2]), they then offered a notation that has a stronger semiotic basis whilst retaining ideas from Padgham et al. [2] when appropriate. This notation has elements that are conformant to the FAML metamodel of Beydoun et al. [4].

In their definition of a modelling language, which contains more detail than we seek at present, Beydoun et al. [4] split their metamodel diagrams into four parts, which correspond interestingly with the viewpoints discussed in Henderson-Sellers [19] and outlined above. Beydoun et al. [4] discriminate between internal versus external (to an agent) and design versus runtime perspectives. Their System-level diagram corresponds to the Organization view of Table 3 together with some aspects of the Knowledge view (specifically in terms of role modelling) and their Environment level diagram to the Environment view. Their agent-definition metamodel fragment depicts specifications for agent types, messages, and plans, *inter alia*, and would therefore seem to have a reasonable correlation with the Services view in Table 3, whereas the agent-level (runtime) portion of the

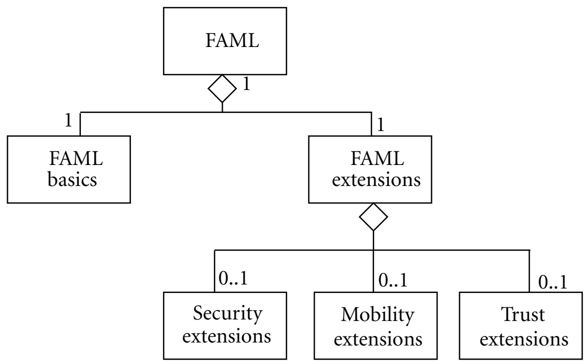

Figure 1: Organizational structure of FAML into basic elements and extensional elements.

Table 6: Initially proposed families and their members.

Family	Members	Shape	Colour (optional)	Source and/or influence for notation
Agents and roles[1]	Agent, role, group, position, organization	Circle atop mask or rectangle	Yellow	INGENIAS [20]
Tasks and plans	Action specification, FAML task, plan specification	Curvilinear	Green	ISO/IEC [21]
Events and resources	Event, resource	Triangular	Blue/green	
Goals	Hard goal, soft goal, belief	Complex curvilinear	Brown	
Ontology	Ontology, service, capability	Polygonal	Dark blue	
Use cases	Scenario, actor	Double oval, stick figure	None	Padgham et al. [2]
Messages	Conversation, message in, message out	Arrow heads	B/W	Padgham et al. [2]

[1] Strictly agent types and role types (design time concepts) rather than their run-time equivalents of individual agents and individual roles.

metamodel goes somewhat beyond the views of Table 3, since it describes metamodelling support for the runtime "Actions" of individual agents, moving on from plan descriptors, for example, to plan enactment. Run-time concepts can thus be linked to some of the dynamic diagram types discussed by other authors (and one of the two discriminators used in this survey). An important distinction is made between agent types (the equivalent to OO classes in a class diagram) and (runtime) agents, which are individuals (equivalent to objects in an OO environment) (see also [59, page 93]). This distinction was made after surveying the literature wherein agent types are often (mis)labelled agents.

The initial studies for the derivation of FAML's metamodel and notation were confined to what might be called "basics" (Figure 1), in that they did not take into account security, mobility, or trust. These are to be regarded as FAML Extensions, the detailed derivation of which is yet to be undertaken.

The set of symbols proposed for the FAML Basics (Figure 1) by Henderson-Sellers et al. [5] have since been slightly modified as a result of questions and discussions at the conference presentation. Figure 2 shows this final set, which we evaluate further in this paper. The principles behind the choice of symbol include ease of drawing, that "families" of symbols should have the same shape and colour (Table 6) and that colour should be an enhancer and not a determinant; that is, the shapes should be understandable in black and white as well as colour. These, and other principles, accord well with the semiotic discussion and principles of Constantine and Henderson-Sellers [6, 84] and Moody [7].

Symbols for agents and roles utilize the role "mask" and its variations. Process-style symbols are similar to those in ISO/IEC [21], topologically similar and green in colour. Events and resources, whilst being a little difficult to defend as a "family", have, nevertheless, similar shapes and colours. Goals, on the other hand, are linked to beliefs as part of the mental state of agents. They use a familiar representation using Yu's [78] i^* notation, as used in agent methodologies like Tropos and Secure Tropos [85]. When used, the fill colour is brown. Ontology, service, and capability are grouped together because both Service and Ontology are linked to Role in FAML. Finally, both scenarios and actors can be linked by their common usage in use case style diagrams. For these, we simply adopt the symbols proposed by Padgham et al. [2].

Agent interactions utilize various variants of an arrowhead (Figure 3). Two alternatives (for MessageIn and MessageOut) were also proposed, but discussants at the EMMSAD conference in June 2012 at which these ideas were first presented were undecided whether the symbols in Figure 3 or in Figure 4 were preferable. Here, we use those of Figure 3.

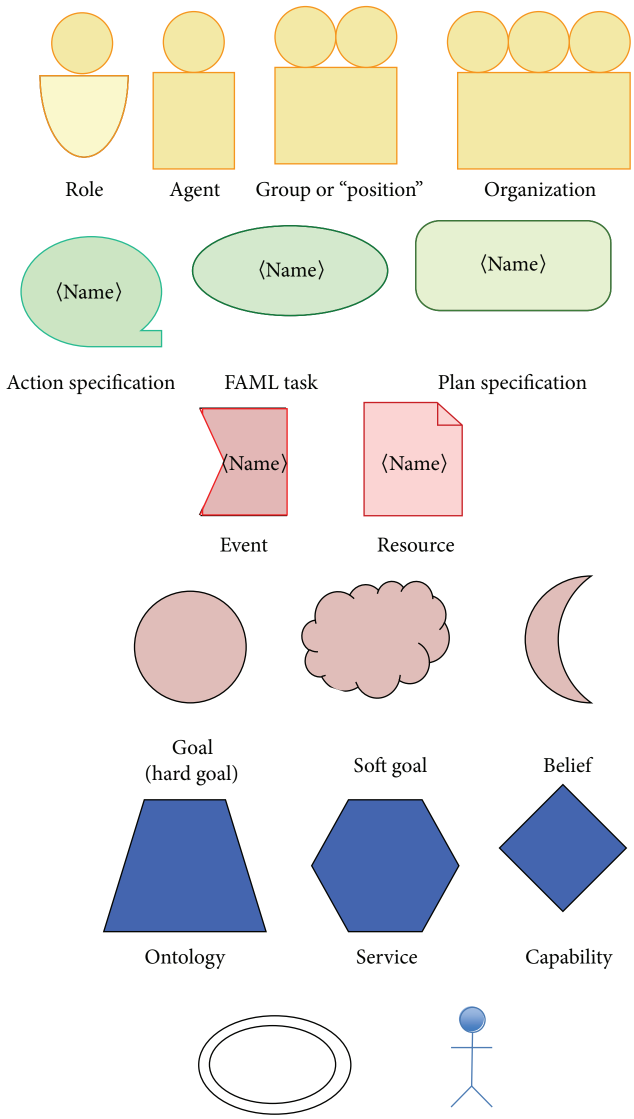

Figure 2: Symbols selected for FAML's notation.

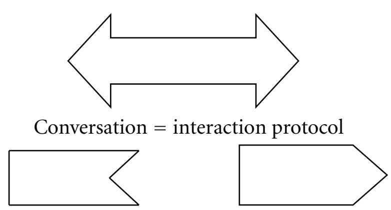

Figure 3: Communication symbols in FAML (after [5]).

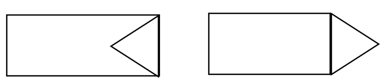

Figure 4: Some suggested alternative representations of agent communication.

4. Diagram Types Used in Current AOSE Methodologies

Henderson-Sellers [19] proposed a number of static and dynamic diagram types for the seven identified views, see Table 4. He then discussed a small selection of methodologies that supported each diagram type, the methodologies being selected from over 20 contemporary AOSE methodologies (Table 7)—excluding those dealing with mobility, for example, Hachicha et al. [86], security (Low et al. [87] discuss security diagrams, offering them as *extensions* to existing diagrams—as shown here in Figure 4), for example, Mouratidis [85], and Bresciani et al. [66], or with noncooperative and adaptive agents. (We, however, do include aspects of ADELFE relevant to cooperative agents), for example, Georgé et al. [88] and Steegmans et al. [89], which introduce additional specifically-focussed concepts, symbols, and diagram

Table 7: Prime references for the AOSE methodologies quoted here.

Methodology name	Main references
ADELFE	Picard and Gleizes [22]
Agent factory	Collier et al. [23, 24]
CAMLE	Shan and Zhu [25]
Cassiopeia	Collinot et al. [26] Collinot and Dragoul [27]
Elammari and Lalonde	Elammari and Lalonde [28]
Gaia	Wooldridge et al. [29] Zambonelli et al. [30, 31]
ROADMAP extensions to Gaia	Juan et al. [32] Sterling et al. [33]
INGENIAS	Pavón and Gómez-Sanz [34] Pavón et al. [20]
ISLANDER	Sierra et al. [35, 36]
MAS-CommonKADS	Iglesias and Garijo [37] Iglesias et al. [38, 39]
MaSE	Wood and DeLoach [40] DeLoach [41–43] DeLoach and Kumar [44]
O-MaSE	Garcia-Ojeda et al. [45] DeLoach and Garcia-Ojeda [46]
MESSAGE	Caire et al. [47, 48] Garijo et al. [49]
MOBMAS	Tran et al. [50] Tran and Low [51]
OperA	Dignum [52] Mensonides et al. [53]
PASSI	Burrafato and Cossentino [54] Cossentino [55, 56] Cossentino and Potts [57, 58]
Prometheus	Padgham and Winikoff [59, 60] Winikoff and Padgham [61] Khallouf and Winikoff [62]
RAP/AOR	Taveter and Wagner [17]
SODA	Omicini [63]
SONIA	Alonso et al. [64]
Tropos	Bresciani et al. [65, 66] Giorgini et al. [67]

types. Furthermore, Tran and Low [51] note that all are deficient in at least one of the three areas of agent internal design, agent interaction design, and MAS organization modelling. The numbers for each diagram type proposed in each of the methodologies of Table 7 are given in Table 8, although it should be noted that some diagram types could be classified under different headings.

In determining to which view (of Table 3) any specific methodological diagram type should be allotted, terminology definitions were sometimes found to be absent, ambiguous, or apparently contradictory. There are several sets of such terms including (i) organization and domain, (ii) interaction diagram and protocol diagram, (iii) goal and task, and (iv) "capability," "service," "responsibility," and "functionality".

Since some authors are using their own definitions, for example, in categorizing views/perspectives, the scoping we have established in Table 3 is sometimes not matched by particular methodological approaches. In particular, our anticipation that the Architecture view should be independent of technology chosen for the solution, as described, for instance, in Giorgini et al. [67], is not met (see further discussion in Section 5.2). In other methodologies, the different use of terms such as "model," "diagram," "view," and "viewpoint" is often unclear (e.g., [20, 29, 31, 39, 49, 90]). As another example, PASSI confounds work product terms with process terms by using model/diagram names to describe tasks.

Another challenge in developing a standard diagramming suite, useful for all AOSE methodologies, is that, while some published methodologies recommend a set of diagrams that occurs in every publication (e.g., [17, 44, 45]), other methodologies continue to evolve so that examination of any one methodology-specific paper often results in difficulty in our determining of what diagrams are recommended for that particular methodology at the present time, although some authors do make it clear what changes have been made (e.g., [62]). In other words, some methodologies contain a stable set of work products, whilst in others the recommended diagramming suite has not yet stabilized.

While Henderson-Sellers [19] attempted to be comprehensive, here we will emphasize those diagram types and diagram usages recommended therein, extending the discussion and incorporating new ideas on AOML notations [5]. When standard UML (OO) diagrams are recommended, we will not include a pictorial representation of what (we assume) will be a diagram well known to readers, being part of the International Standard 19501 [93].

We do not undertake a side-by-side methodology comparison, as is done, for example, in Tran and Low [94] or, more recently, in Dam and Winikoff [95]. Rather, we try to exemplify some of the differences in representational style for diagrams pertinent to each of the several views identified in Henderson-Sellers [19] and summarized below.

In the following two sections, we analyze diagram types currently used in a number of AOSE methodologies using the framework of Table 4. Section 5 discusses the various static diagram types and Section 6 the dynamic counterparts. For both sections, we adopt the seven views deduced in Henderson-Sellers [19] plus the added UI view (Table 3) and try to make additional suggestions, where appropriate, regarding appropriate notations for these identified diagram types.

The *Environment View* is used either to describe the interface between the MAS and the external entities in the problem domain and/or the externalities to the MAS (Figure 5). Indeed, domain modelling is seen by Müller [96], Parunak and Odell [97], and Dignum and Dignum [98] as being crucial.

Relevant diagram types may be solely focussed on the environment (a.k.a. domain), but there are many methodologies in which an organizational diagram type, as discussed

Table 8: Summary of the number of distinct usages of each diagram type per methodology.

	View										
	Environment		Architecture	Agent societies		Agent knowledge		Agent services		Deployment	User interface
	Static	Dynamic	Static	Static	Dynamic	Static	Dynamic	Static	Dynamic	Static	
ADELFE		1		1	1						
Agent factory		1		1	3						
CAMLE				1	1		2				
Cassiopeia				1							
Elammari and Lalonde		1	1	1 + 2 textual		1 textual					
Gaia	1		1	2 + 1 textual		2 textual					
ROADMAP extensions to Gaia		1		1 textual		1					
INGENIAS	1		2		2	1		1			
ISLANDER				2		1 + 1 textual	1	1	1		
MAS-CommonKADS		1	1	1	2	4 + 1 textual			1	2	
MaSE		1	1	2	2	1	2		1	1	
O-MaSE (extras)				2	1	3					
MESSAGE	4			5	1	2	1	2			
MOBMAS	4		1	2	2	7 + 1 textual	1			1	
OperA				4 + 5 textual	1	1 + 1 textual					
PASSI	1	1		2	3	2	1		1	1 + 1 textual	
Prometheus		2		2	2	5 + 2 textual			1		
RAP/AOR		2		5	2					1	
SODA	1			2		2 textual					
SONIA						3		1			
Tropos					1	5				1	

in Section 5.3, serves a second purpose: that of including not only the agent organization but also its interface with the environment, whilst retaining the (perhaps confusing) name of "organizational diagram." This is especially seen in methodological approaches such as MAS-CommonKADS and MESSAGE. For the organizational model of the former, it is clear that the organization model is intended to serve also beyond the agent organization and to interface with the environment, since the recommended notation for the organizational model (Figure 6) includes sensors and actuators (a.k.a. effectors) in the agent symbol (actually an agent type—see earlier discussion).

In other words, some of the diagram types discussed in Section 5.1 could well be equally allocated to Section 5.3 (and vice versa). (For a more detailed and more philosophical discussion of environment abstractions, see Viroli et al. [99]). Environment was also recently a major topic of conversation within the OMG as part of their emerging interests in agents [100].

For the *Architecture View*, we note that the term "architecture" can have many interpretations in the context of an MAS. Here, we use it to describe large-scale features that are independent of the technology used to undertake the implementation of the MAS. In different AOSE methodologies, the level of detail can vary—some diagrams include agents and their roles whilst others do not.

Both the Environment View and the Architecture View diagrams are restricted to static diagram types.

In the *Agent Societies View*, diagrams depict agent societies or organizations. (As noted earlier, the term organization can be used both as a synonym for society and to represent the environment); for example, Ferber and Gutknecht [102] provide more detail than that of an architecture diagram. They typically focus on agent interaction

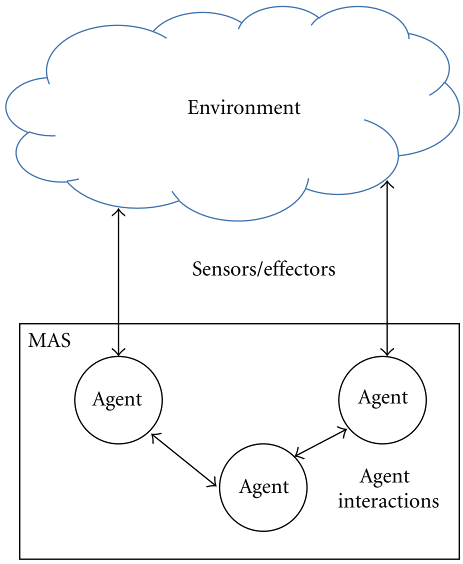

Figure 5: Agents in an MAS interact with their environment using sensors and effectors.

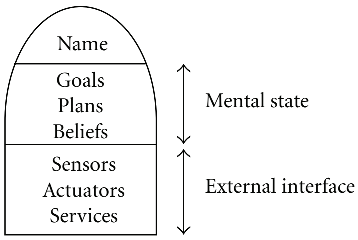

Figure 6: Organizational model notation for MAS-CommonKADS (based on [37], reprinted by permission of the publisher © IGI Global).

rather than system structure (e.g., [63, 103]). Indeed, the architectural diagrams identified in Section 5.2 for various AOSE methodologies can also be extended to depict agent society details.

Furthermore, "organizational patterns" (i.e., patterns applied to agent societies) are discussed in Zambonelli et al. [30] and Gonzalez-Palacios and Luck [104]. Typical examples include pipeline, single hierarchy, and multiple hierarchies.

Here, we seek to depict how agents interact in terms of such an interacting society of agents and/or roles, again dividing the discussion into static and dynamic aspects.

The *Agent Workflow View* relates solely to dynamic diagram types since a workflow reflects agent behaviour. This can involves concepts such as process, actions, and interagent messaging.

For the *Agent Knowledge View* we need to represent the internal structure and behaviour of individual agents. Concepts such as goals, beliefs, commitments, plans, capabilities, perceptions, protocols, events, sensors, actuators, and services are all considered by one or more authors. In Section 5.4 we focus particularly on goals, ontologies, and plans.

The *Agent Services View* can involve a number of different diagramming techniques (see Table 4) including goals, tasks, capabilities, and a domain ontology. Services can be described as encapsulated blocks of offered functionality [30, 32, 105]. In AOSE, a service may be described in terms of capabilities, where a capability is defined as "the ability of an actor of defining, choosing and executing a plan for the fulfillment of a goal, given certain world conditions and in presence of a specific event" [65], a definition similar to that used in Prometheus.

For the *Deployment View*, the allocation of software components to hardware nodes has traditionally been the focus; for AOSE a greater emphasis is placed on agent conversations.

Finally, the *UI View* is ill represented in current AOSE methodologies. Our discussion therefore makes suggestions from outside the agent-oriented methodology community.

In the following two sections, citations to specific methodologies will be by methodology name rather than author name(s)—these are found in Table 7—unless a specific paper needs a direct citation. We introduce methodology-specific examples of diagram types not discussed in Henderson-Sellers [19] and assess their match to the previous recommendations. We also describe a selection of these diagrams with the new FAML notation [5], merely as an illustration of the visualization resulting from the combination of a specific diagram type and this notation. We introduce an oversimplified running example in the Travel Agent domain. None of these diagrams are intended to be a complete depiction but rather should be regarded as merely illustrations of the diagramming style to which they refer.

5. Static Diagram Types

5.1. Environment View. For an MAS, the environment is relevant to two separate phases of the development lifecycle.

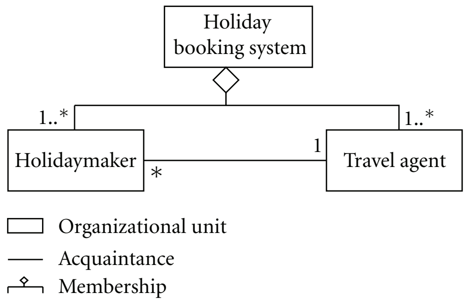

Figure 7: Organization context chart in MOBMAS.

Initially, requirements will relate to real-life problems, and the MAS will itself interact with this environment. This interaction will be evident in both the analysis and design phases. Secondly, environment issues are relevant in the deployment phase, when allocation of software code to a specific run-time platform node is necessitated. This second interface occasion is described in Section 5.6.

The recommended diagram type [19] for the environment description diagram, which models the external environment, is a UML-style class diagram with entities representing domain entities. For the environmental connectivity diagram, which shows the interfacial linkages between the environment and the top level agents in the MAS, particularly in terms of how agents are likely to access external resources such as databases, actors, and other MASs, a UML-style class diagram can also be useful. A third diagram type (more optional) is an *External organization structure chart:* a UML-style class diagram with entities = organizational unit, decomposition using the membership relation and acquaintance relationships between collaborating organizational units (see, e.g., Figure 7, which shows the use of this style of diagram in MOBMAS).

Environment description diagrams are also used in SODA and PASSI. INGENIAS offers an Environment Viewpoint diagram (Figure 8) depicting the external entities with which the MAS-to-be-constructed will interact.

A second style of diagram is often used to describe the functionality aspects relevant to the interaction between external stakeholders and the software system. This often relates to an early stage in the lifecycle, when requirements need to be identified and documented. Here, it is fairly common practice to use some sort of use case diagram, identical or very similar to that proposed in UML [10]. Henderson-Sellers [19] recommends that, to appropriately support the agent aspects more accurately, a goal-based use diagram that extends the "User-Environment-Responsibility (UER) case" diagram of Iglesias and Garijo [106] is useful for showing agent actors as well as human actors. An example of this is shown in Figure 9. To accompany this, a set of completed use case templates is necessary, such as that provided in Prometheus or by Taveter and Wagner [17], as originally proposed by Cockburn [107] (see example in Table 9). Here, the internal and external actors correspond directly to internal and external agents in AOR modelling.

Table 9: Example of a goal-based use case, here for the business process type "Process the request for a quote" (after [17], reprinted by permission of the publisher © IGI Global).

Use case 1	Process the request for a quote	
Goal of the primary actor	To receive from the seller the quote	
Goal of the focus actor	To provide the buyer with the quote	
Scope and level	Seller, primary task	
Success end condition	The buyer has received from the seller the quote	
Primary actor	Buyer	
Secondary actors		
Triggering event	A request for a quote by the buyer	
Description	Step	Action
	1	Check and register the availability of the product items included in the request for a quote
	2	Send the quote to the buyer.

As is the case with the use of use cases in object-oriented software development, the use case diagram only offers a high level viewpoint on requirements. Of more value [107] is the textual description of each use case. In the Prometheus approach, Padgham and Winikoff [59] note that, since agents have abilities beyond those of objects, it is necessary to provide a textual template significantly beyond those found in OO requirements engineering. Specifically, their textual template (called a "functionality descriptor") describes the system functionality in terms of name, description, percepts, actions, data used/produced, and a brief discussion of interactions with other functionality. While these functionality descriptors are said to be intermediate work products, a final work product that is cross-checked (Figure 10) with them is the use case scenarios (or "scenarios" for short). These are again textual—a typical scenario descriptor in Prometheus is given in Table 10. Each step described in the scenario is a small piece of functionality.

Other methodologies use UML use cases "as-is," for example, MaSE, ROADMAP, ADELFE, MAS-CommonKADS and PASSI where it is called a "domain descriptor diagram."

5.2. Architecture View. Henderson-Sellers [19] recommends a UML-style package diagram as the Organization-based architecture diagram (Figure 11) similar to that used in MESSAGE [49] (Figure 12), although this diagram often has a different name, using different basic shapes, for example, organization structure model in Gaia [29] (Figure 13), or as a jurisdictional diagram (as in Figure 14).

In INGENIAS, the generic elements of Figure 15 are used to depict an exemplar organizational viewpoint model in Figure 16 (notational key is given in Figure 17).

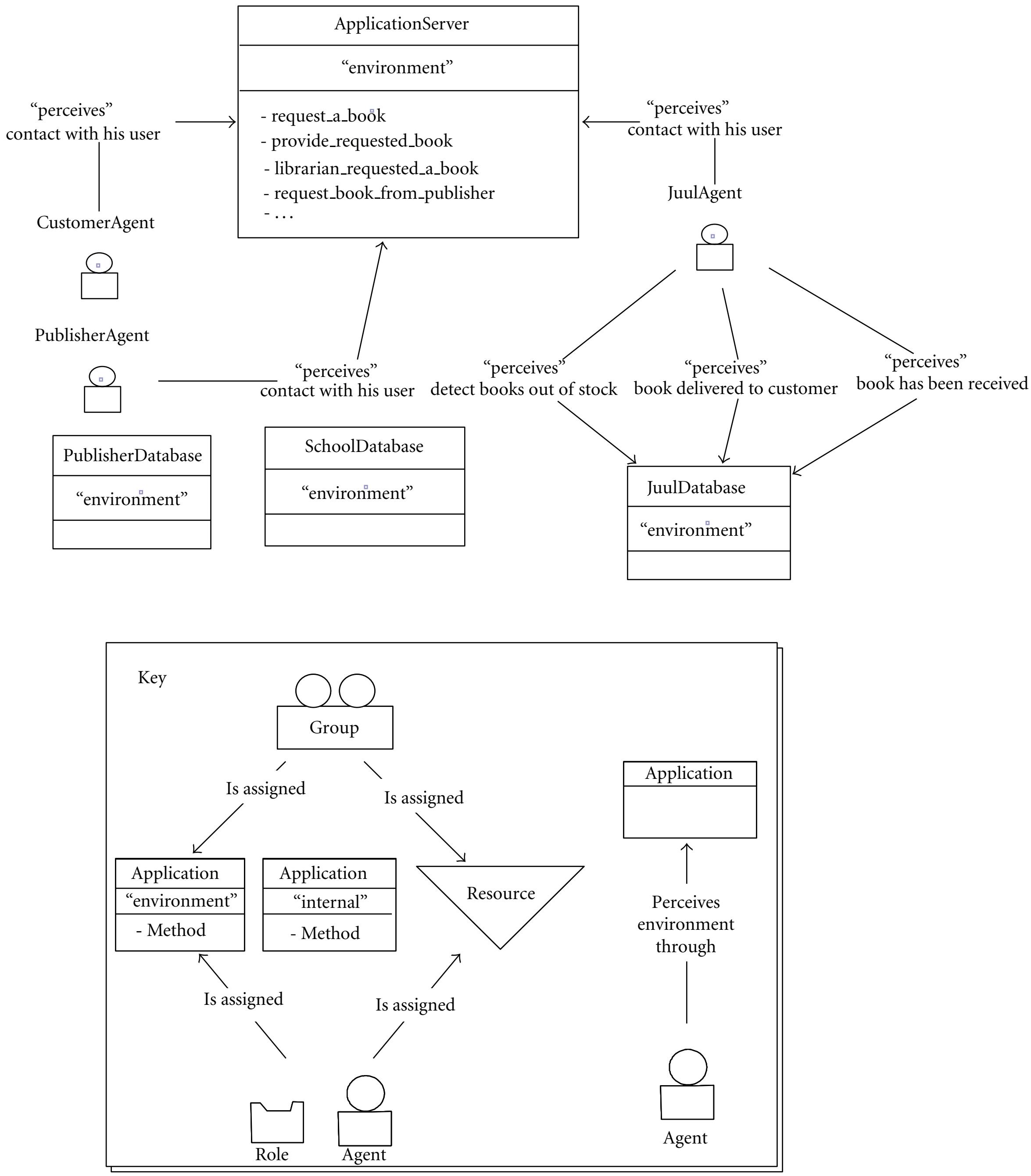

Figure 8: Example of an INGENIAS environment viewpoint diagram (after [20]), with key showing typical elements (after [20], reprinted by permission of the publisher © IGI Global).

5.3. Agent Societies View. A large number of AOSE methodologies have a strong focus on agent societies, especially SODA, ISLANDER, and OperA. Social structures were added to the earlier version of Gaia by Zambonelli et al. [30].

The style of an agent society diagram recommended in Henderson-Sellers [19] is that of a UML-style class diagram showing all agents and all interrelationships. Other information that may be chosen for display includes percepts, actions, capabilities, plans, data, and messages represented by entities rather than relationships. An example is seen in Figure 18, which uses Prometheus notation and depicts individual messages in the style of Padgham and Winikoff [60], for example, their Figure 5 (p. 123). An alternative depiction is given in Figure 19, which gathers messages into interaction protocols, following the style of Padgham and Winikoff [60], for example, their figure in page 93; part of which is also represented with FAML's notation in Figure 20.

Other diagram styles can be seen in, for instance, MAS-CommonKADS (Figure 21), which shows the mental state and internal attributes of an agent (e.g., goals, beliefs, and

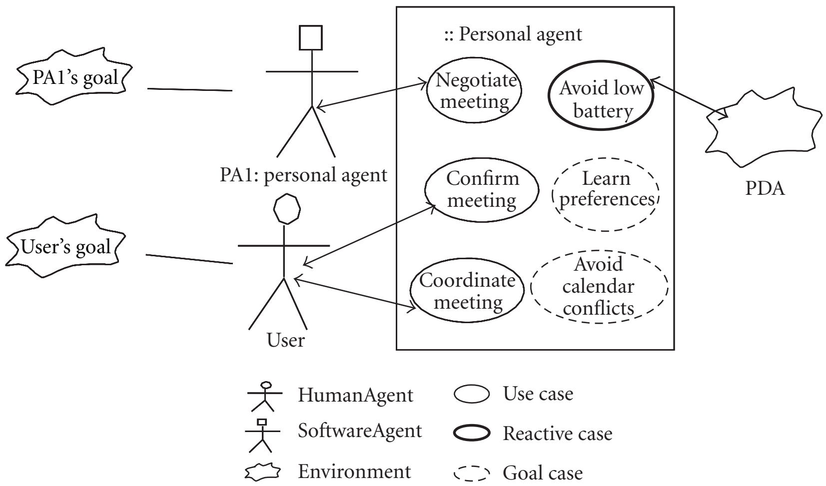

Figure 9: Recommended diagram for an agent goal-based use case diagram, (reprinted from [19], copyright 2010, with permission from IOS Press).

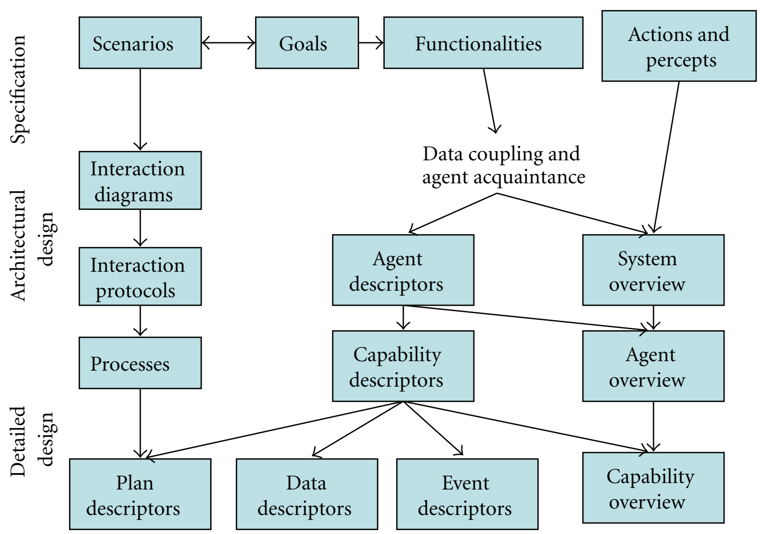

Figure 10: Phases and work products defined in Prometheus (after [60], reprinted by permission of the publisher © IGI Global).

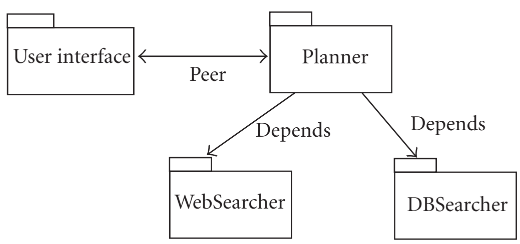

Figure 11: Recommended diagram style for an Architecture Diagram.

plans) (upper box) together with the external attributes of the agent (e.g., services, sensors, and effectors) (lower box) (Figure 6); MASE (Figure 22), in which the connections between classes denote conversations that are held between agent classes, and the second label in each agent class represents the role the agent plays in a conversation; MOBMAS (Figure 23), which shows acquaintances between agent classes and connections between these and any wrapped resources, a diagram that may also be enhanced to show protocols and associated ontologies; AOR (Figure 24), which

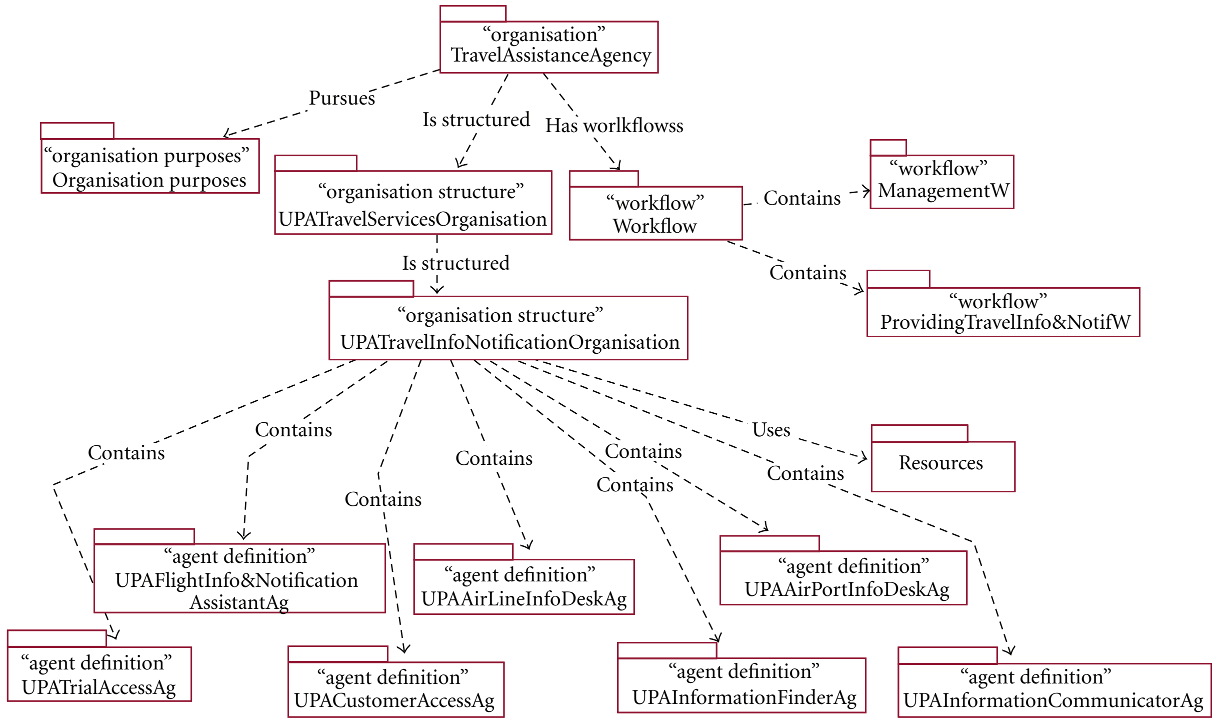

Figure 12: Organization-based architecture diagram of MESSAGE (after [49], reprinted by permission of the publisher © IGI Global).

Table 10: Example of a Prometheus scenario descriptor (after [60], reprinted by permission of the publisher © IGI Global).

Name: new meeting scheduled

Description: the user adds a new meeting

Trigger: new meeting requested by user

Steps:

No.	Type	Name	Functionality	Data
1	Percept	Request meeting	User interaction	
2	Goal	Propose time meeting user preferences	Meeting scheduler	MeetDB(R), Prefs(R)
3	Goal	Negotiate with other users	Negotiator	MeetDB(R), Prefs(R)
4	Goal	Update user's diary	Meeting manager	MeetDB(W)
5	Goal	Inform others of meeting	Contact notify	
6	Other	Wait for day of meeting		
7	Goal	Remind user of meeting	User notify	MeetDB(R), Prefs(R)
8	Goal	Remind others of meeting	Contact notify	ContactInfo(R)

Variations:
(i) Steps 2-3 may be repeated in order to obtain agreement.
(ii) If agreement on a meeting time is not reached then steps 4–8 are replaced with notifying the user that the meeting could not be scheduled.
(iii) The meeting can be rescheduled or cancelled during step 6 (waiting).

Key:
(i) MeetDB(R): meetings database read.
(ii) Prefs(R): user preferences read.
(iii) MeetDB(W): meetings database written.
(iv) ContactInfo(R): contact information read.

depicts agent types, their internal agents, and the relationships between them; PASSI (Figure 25); and ADELFE (Figure 26), which depicts the connectivity between cooperative agents, for which ADELFE was specifically designed (Figure 26).

Another approach to agent societies is to utilize a version of the UML collaboration diagram, although the omission of any sequencing of communications makes this use somewhat dubious. Typically, they depict static aspects of the agent society rather than being dynamic interaction diagrams (as

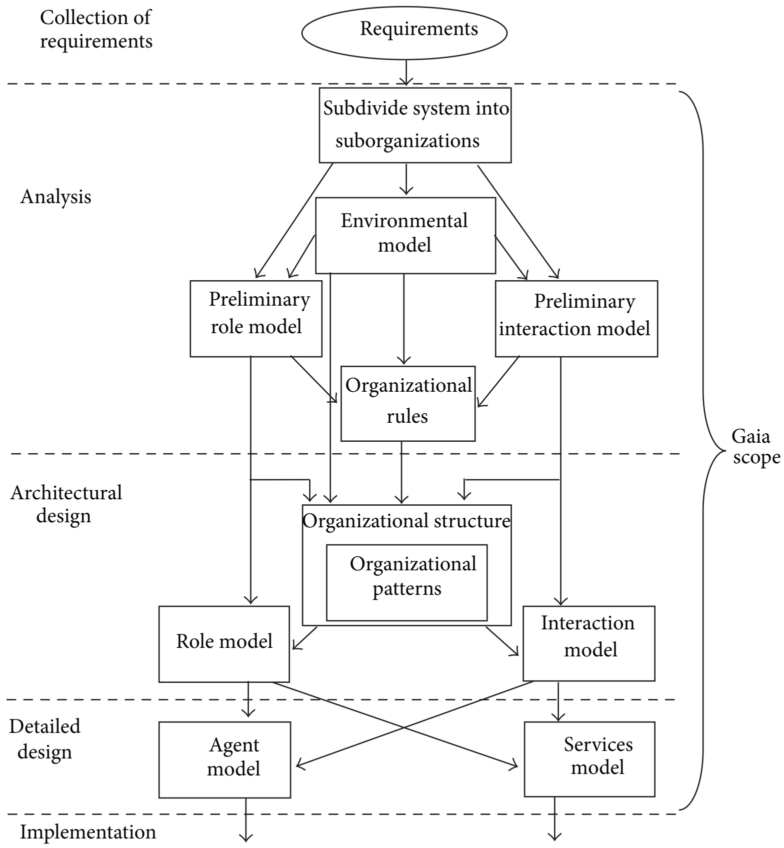

Figure 13: Phases and work products ("models") defined in Gaia (after [31], reprinted by permission of the publisher © IGI Global).

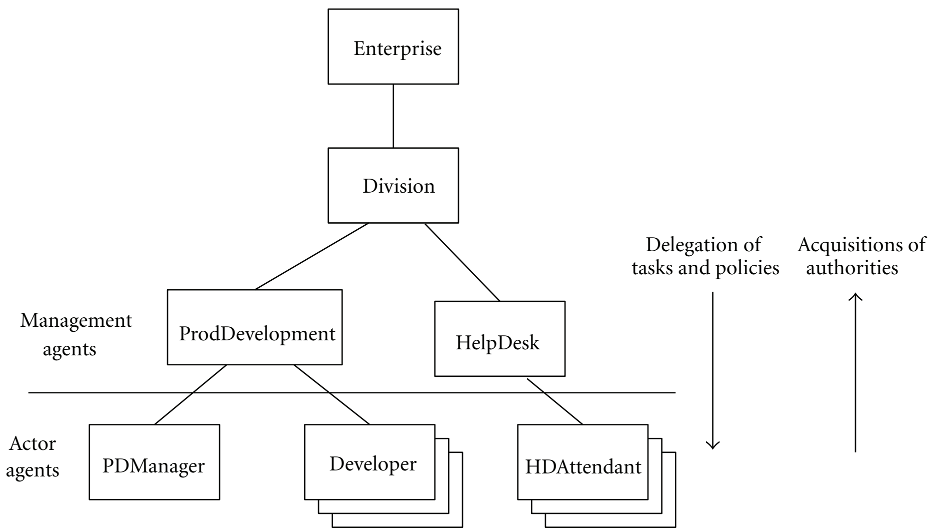

Figure 14: A jurisdictional model, showing management agents, actor agents, and subagents, part of the Agent Relationship Model (after [28]).

any true variant on a UML collaboration would be classified). Hence, they are summarized in this subsection.

The event flow diagram of MAS-CommonKADS [39], for example, represents events (the exchange of messages) (Figure 27) but does not depict any message sequencing. A somewhat similar diagram is to be found in OperA (Figure 28).

Visually different are the interaction-frame diagrams of RAP/AOR. This is used at both the class level (Figure 29) and the agent-instance level (Figure 30). In these diagrams, the

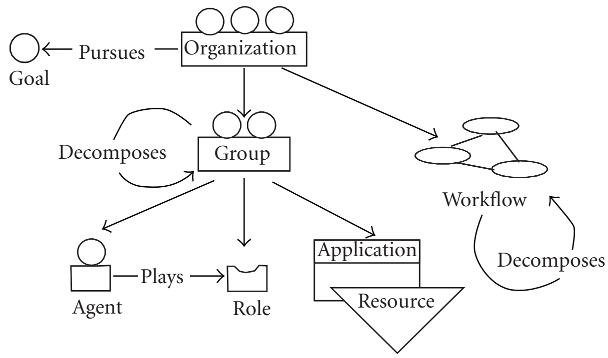

Figure 15: Typical elements in an INGENIAS organization viewpoint (structural description of an MAS organization) (after [20], reprinted by permission of the publisher © IGI Global).

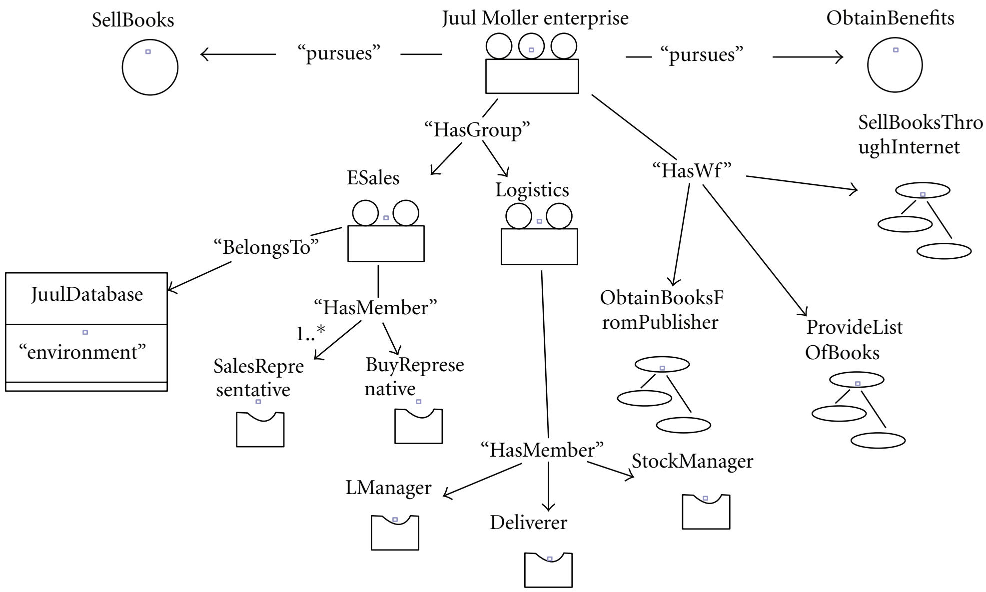

Figure 16: Example of an INGENIAS organization viewpoint (structural) description (after [20], reprinted by permission of the publisher © IGI Global).

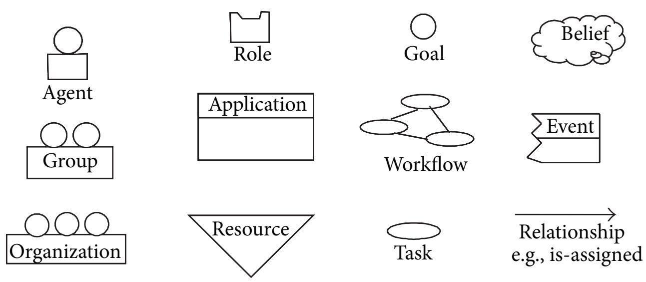

Figure 17: Notational key for INGENIAS diagrams.

solid arrows indicate a noncommunicative action event type, and the chain dashed arrows are message types.

Whilst not a methodology, MAS-ML [71] suggests, in this context, the use of two diagrams that have a UML style class diagram to them: (i) the organization diagram and (ii) an extended UML Class Diagram that shows the "structural aspects of classes, agents, organizations, environments, and the relationships between these entities," by introducing additional concepts (i.e., additional to those of UML Class Diagrams), including Environment Class,

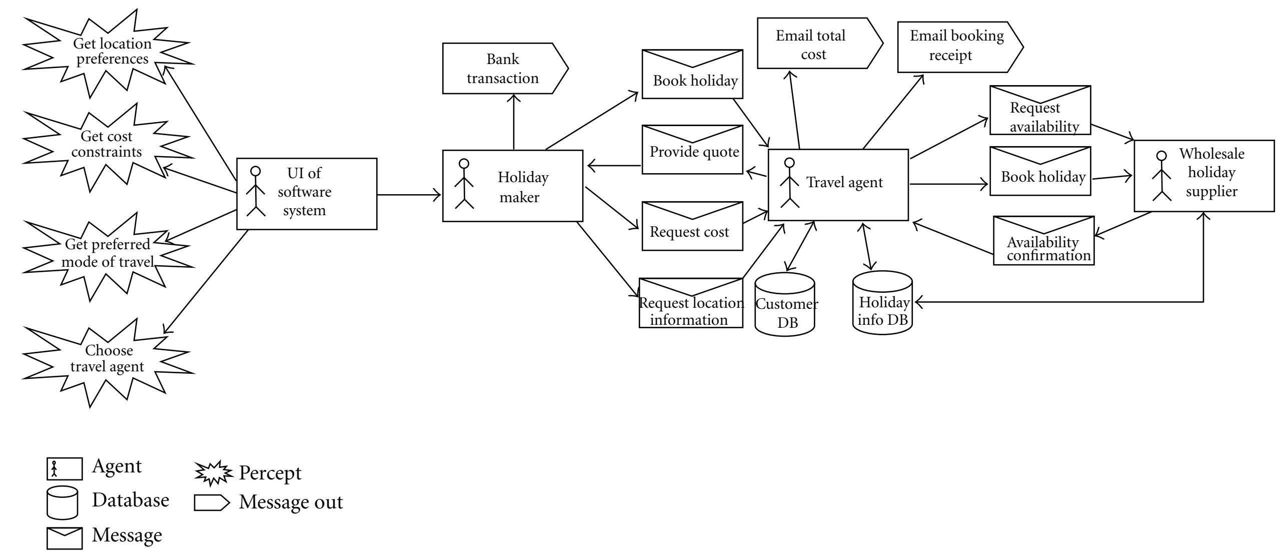

FIGURE 18: Prometheus style "system overview diagram" depicting messages.

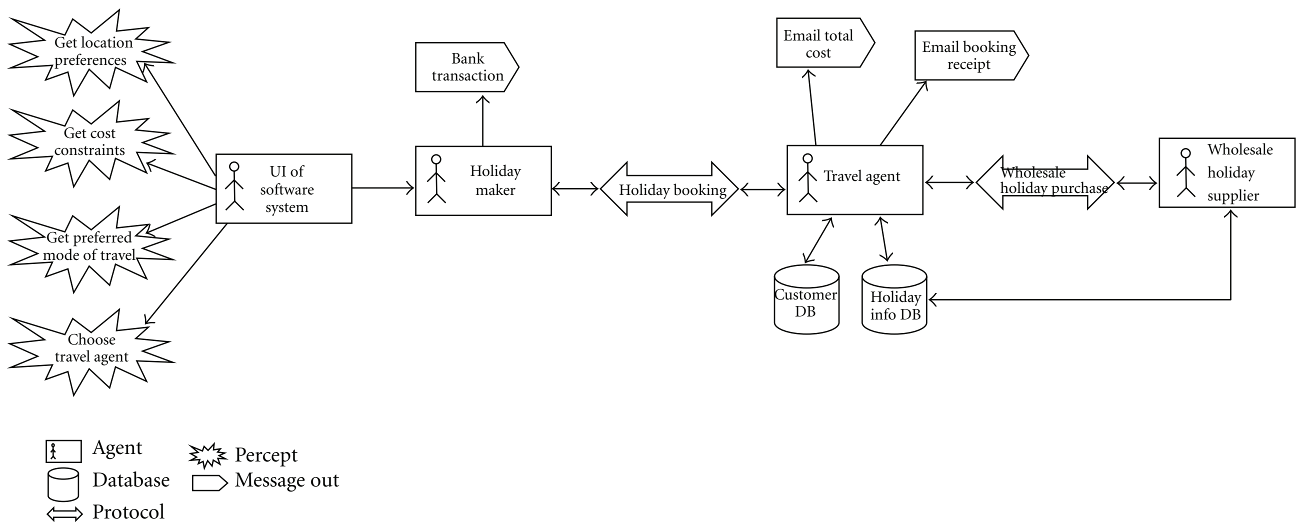

FIGURE 19: Prometheus style "system overview diagram" depicting protocols.

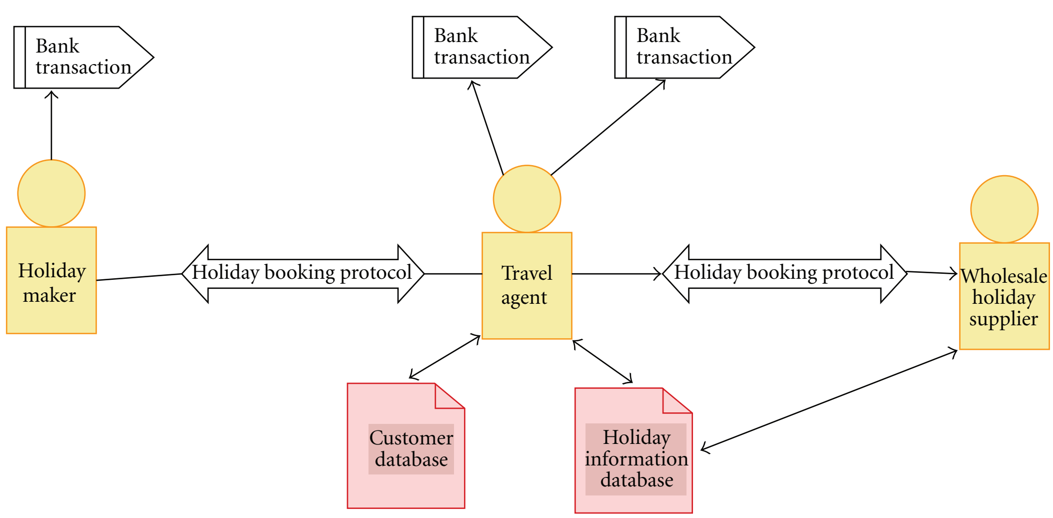

FIGURE 20: A portion of Figure 19 "translated" into FAML notation.

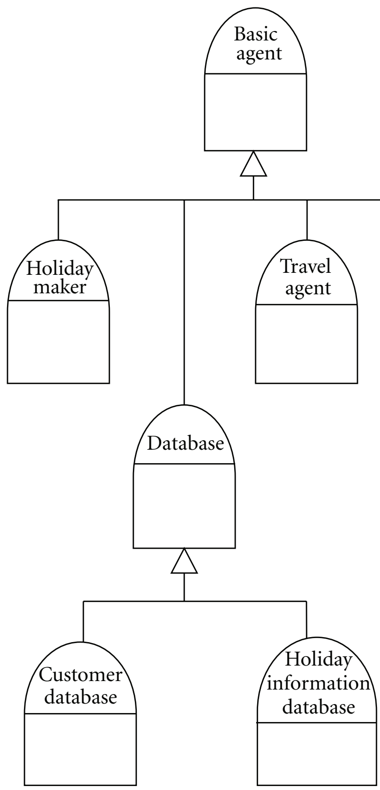

Figure 21: Example organizational model as recommended in MAS-CommonKADS. In the lower part of each symbol are shown the mental state and external interface as shown in Figure 6.

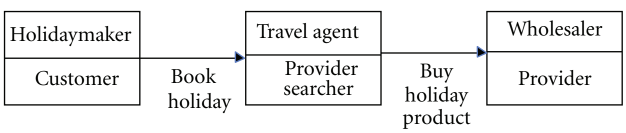

Figure 22: MaSE agent class diagram.

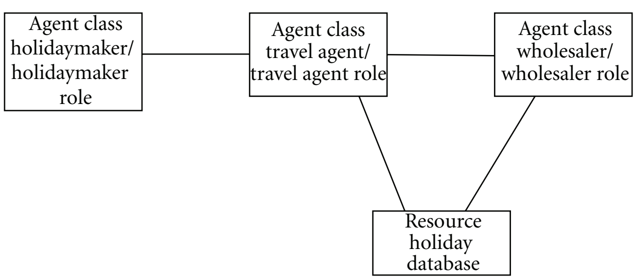

Figure 23: MOBMAS agent relationship diagram.

Organization Class, and Agent Class. The notation used (both for these two diagrams and their role diagram—see later discussion) is shown in Figure 31.

5.3.1. Roles. Although supported to some degree in object-oriented modelling languages, the much greater importance of roles in AOSE modelling [109–113] requires separate consideration, despite the common adoption of a UML-style class diagram to depict roles (Figure 32), for example, as used in Agent Factory, MOBMAS, PASSI, and O-MaSE. It should be noted that, in Figure 32, each role class is characterized by its associated so-called protocol identifiers, while each agent class is characterized by a list of protocol identifiers (any not specified in the associated role) and activities. Consequently, it is argued that a two-compartment symbol is needed for roles, and a three-compartment symbol for agents. In ROADMAP also, roles are explicitly linked to agents, as shown in Figure 33 [114]. Such role-focussed diagrams should be supplemented by a Role definition template (see, e.g., Table 11). The original ROADMAP template [115] was later simplified by Sterling et al. [33] as shown in Table 12.

A second role-focussed diagram is the role dependency diagram: a simple role decomposition diagram, using just names and unadorned lines. Figure 34 is one such example. This may be supplemented by a textual description of all the roles and their dependencies (Table 13).

Whilst not a methodology, MAS-ML [71] uses a UML-like Role Diagram using the notation given in Figure 31 to show "structural aspects of agents roles and object roles defined in the organisations" and their interrelationships. Concepts additional to those of UML Class Diagrams include Object Role Class to represent resources utilized by an Agent Role Class.

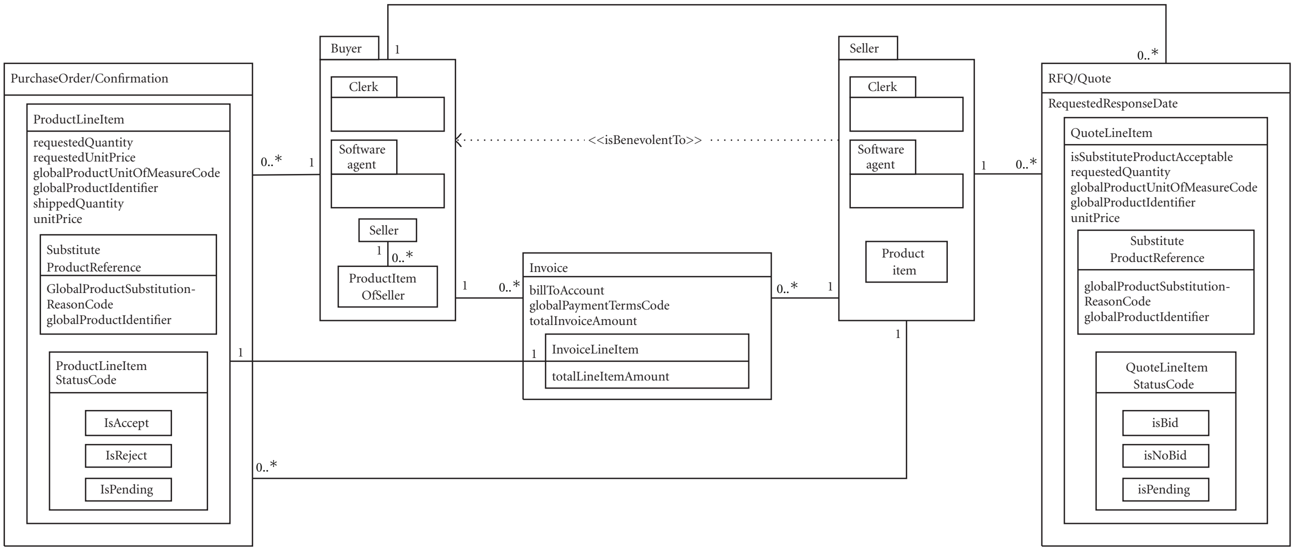

Figure 24: AOR agent diagram for the domain of B2B e-commerce (after [17], reprinted by permission of the publisher © IGI Global).

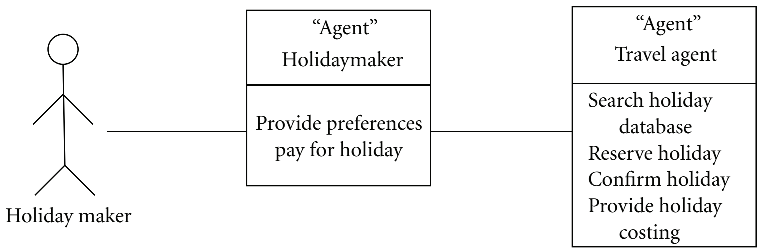

Figure 25: PASSI's Multiagent Structure Definition Diagram.

Table 11: Role definition as used in OperA (after [52]).

Role: PC member
Objectives:
Paper_reviewed(Paper, Report)
Subobjectives:
{read(P), report_written(P, Rep), review_received(org, P, Rep)
Rights:
access-confman-programme(*me*)
Norms:
OBLIGED understand(English)
IF DONE assigned(P, *me*, Deadline)
THEN OBLIGED paper_reviewed(P, Rep) BEFORE Deadline
IF DONE paper_assigned(P, *me*, _) AND is_a_direct_colleague(author(P))
THEN OBLIGED review_refused(P) BEFORE TOMORROW
Type:
External

Figures 35 and 36 show how the role dependency diagram of Figure 34 and the Agent-role dependency diagram of Figure 33 can be depicted using the FAML notation of Figure 4.

5.4. Agent Knowledge View. The static knowledge of individual agents is encapsulated in symbols for each agent type, typically by extending a basic icon, such as the UML rectangle or the MOSES tablet (as used, for instance, in Figure 21). Suggestions here include goals, beliefs, commitments, and plans added to the basic class symbol (Figure 37); however, several authors (e.g., [76]) argue that it is not yet clear whether the attributes and operations are valid features of an agent type. Since these are derived (via the metamodel) from the UML Classifier, their rejection would negate the generalization relationship between Agent and Classifier (as shown in Figure 36). (This is another illustration of the confounding in the literature between agent and agent type. Often what is referred to as an agent is an agent type, that is, the word is used to describe an entity that conforms to some subtype of Classifier in the metamodel. Although in our following analysis we will continue to use "agent" when quoting from the AOSE literature, it should be remembered that usually this should be replaced by "agent type").

Thus this suggests the need for an agent modelling language *not* based on UML (see revisions, proposed here, in Figure 38). Other proposals are to explicitly depict

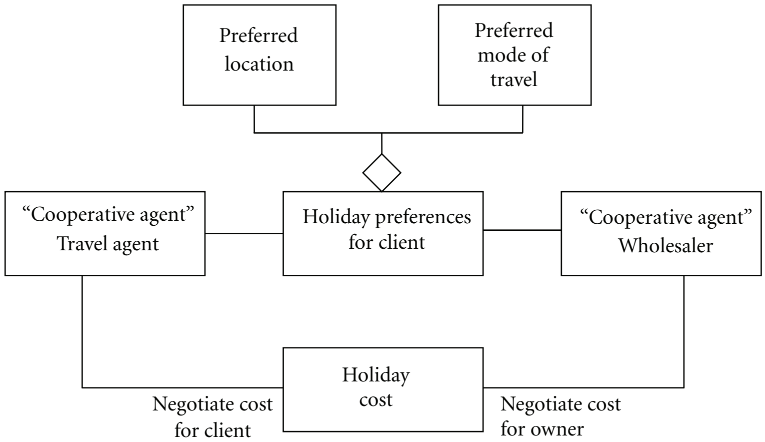

Figure 26: Typical ADELFE class diagram showing two cooperative agents and their interrelationships with other classes.

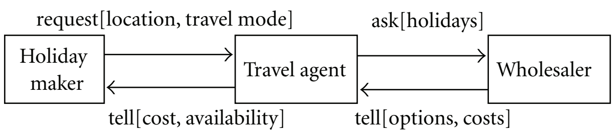

Figure 27: Event flow diagram using MAS-CommonKADS notation.

Table 12: Simplified version of the role template as used in more recent versions of ROADMAP—here for an Intruder Handler agent (reprinted from [33], copyright 2006, with permission from IOS Press).

Role name
Intruder handler
Description
Identifies and responds to the intruder detected
Responsibilities
Detect the presence of a person in the environment
Check the house schedule for strangers scheduled to be there
Take an image of the person
Compare the image against the database of known people
Contact the police and send the image to them
Check the house schedule for planned visitors
Send a message to stay away to each visitor expected that day
Inform the owner that the police are on their way and the visitors have been warned not to enter the house
Constraints
The owner and each person pointed out by him/her needs to provide in advance personal information (face) to be recognised
A subject to be detected needs to be seen within the camera's image area
The user must maintain the schedule
Visitors must be within the coverage area of mobile communication with their mobile access terminals switched on

Table 13: Social structure definition in OperA (after [52]).

Social structure definition
Roles:
A list of role definitions
Role dependencies:
A list of triples of two role names and the name of the relationship between them
Groups:
A list of sets of roles

capabilities, perceptions, protocols, and organizations (Figure 39); belief conceptualization and events (Figure 40); or sensors, actuators, and services (Figure 6); these often being supplemented by agent descriptors (see, e.g., Boxes 1 and 2). Textual templates for agent roles are recommended in Gaia [29]. Such a schema (one per role) gives information about the protocols, permissions, and responsibilities (liveness and safety) as well as an overall description for each agent role. It is also used by Suganthy and Chithralekha [118] and in SODA.

Knowledge is often expressed in terms of the "mental state" (Figure 41) of an agent, as described, for instance, in the Agent viewpoint of INGENIAS [20]—see example in Figure 42. This idea of "mental state" is also found in AOR (Figure 43) and in Silva et al. [119] who link the set of beliefs, goals, plans, and actions to the mental state of the agent.

Agent internals are represented in Prometheus in terms of an Agent Overview diagram (one for each agent). For the Travel Agent agent in Figure 19, Figure 44 shows its capabilities, percepts, and messages utilized. It is similar in style to the System Overview diagram of Figure 19 but at a finer granularity (Figure 45). Then each capability in the Agent Overview diagram can be expanded in a Capability Overview diagram (Figures 46 and 47).

In SONIA, the knowledge model is represented differently, by blocks of knowledge that group concepts and

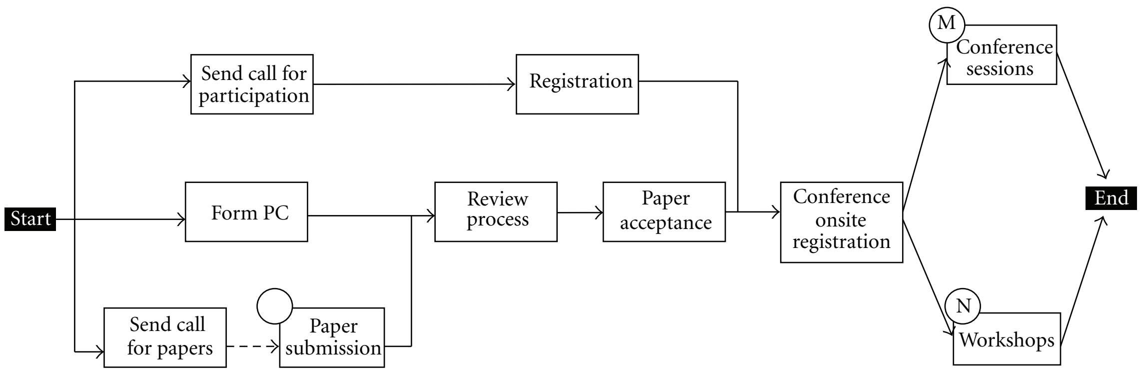

Figure 28: Interaction diagram from OperA for the "Conference" society (after [52]).

Name: Travel agent Description: Arranges holiday for holidaymaker client Cardinality: One per holidaymaker client Lifetime: Duration of interaction with Holidaymaker Initialization: Reads Customer database, receives messages from Holidaymaker Demise: Closes open database connections Functionalities included: Obtaining wholesale product, supplying packaged holiday to holidaymaker Uses data: Customer database, Holiday information database Produces data: Recommendation to holidaymaker client Goals: Respond to queries; obtain best deal for wholesaler; package wholesale products for client Percepts responded to: Logon by holidaymaker Actions: Advice of cost; Send receipt Protocols and interactions: Holiday booking protocol, Wholesale holiday purchase

Box 1: Example agent descriptor in Prometheus format.

Agent: Travel agent Role: Arranger of holidays for holidaymaker clients Location: Inside holiday booking agent society Description: This agent manages client requests and interfaces with wholesaler of holidays Objective: Get best deal for holidaymaker client subject to client preferences Exceptions: Missing preferences or fully booked holidays Input parameter: Client preferences Output parameter: Recommendations to client including costing Services: Recommendation to holidaymaker client; holiday booking with wholesaler Expertise: Respond to queries; obtain best deal for wholesaler; package wholesale products for client Communication: Holidaymaker agent; Wholesaler agent Coordination: Wholesaler agent

Box 2: Example of an agent definition template as proposed in MAS-CommonKADS—a format suggested by Peyravi and Taghyareh [68].

associations from the structural model. These knowledge blocks may be used internally or shared between agents.

5.4.1. Goals. Another aspect of agent knowledge is that of goals: agent goals as well as system goals. A goals is said to represent a state that is to be achieved, for example, Braubach et al. [120]—although other kinds of goals are possible, and goals may conflict with each other, for example, Van Riemsdijk et al. [121]. Goals are achieved by means of actions (a.k.a. tasks) (Figures 48 and 49), the combination of

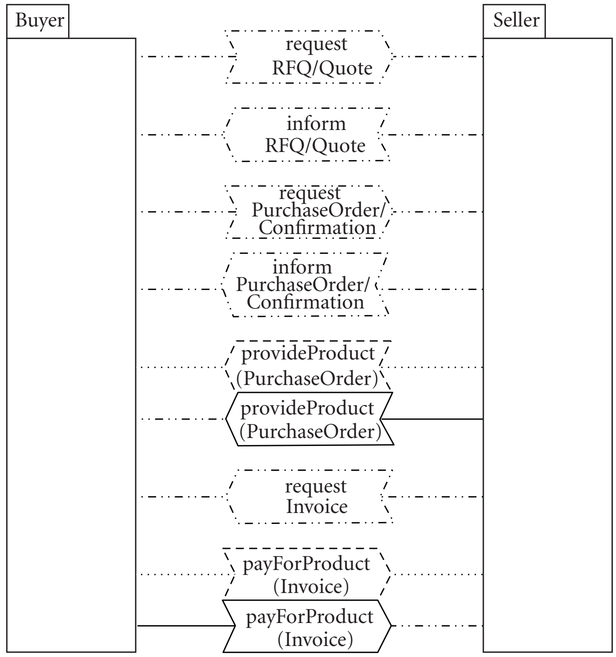

Figure 29: AOR interaction frame between agents of the types Buyer and Seller (after [17], reprinted by permission of the publisher © IGI Global).

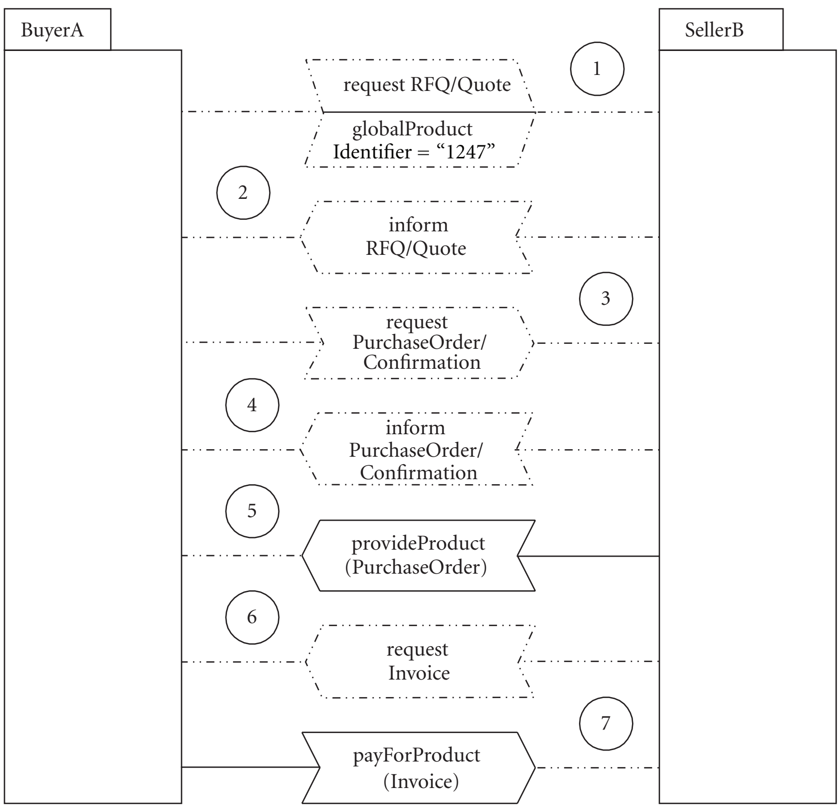

Figure 30: AOR interaction sequence between agent instances BuyerA and SellerB (after [17], reprinted by permission of the publisher © IGI Global).

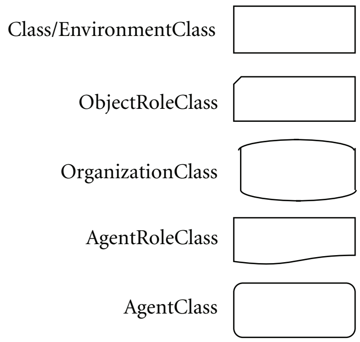

Figure 31: Notation used in MAS-ML.

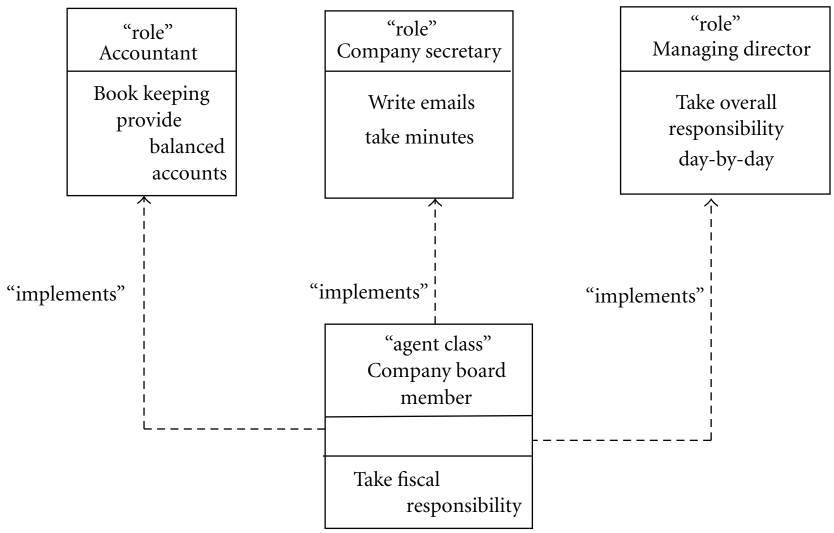

Figure 32: Example of Agent Factory's agent model.

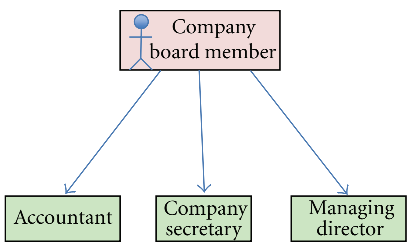

Figure 33: Agent-role coupling diagram of ROADMAP.

actions and goals forming the plan body. Tasks are discussed later in Section 6.4.

Goal-focussed diagrams are found in Prometheus (Figure 50), in MaSE, and in MESSAGE's "Level 0 Goal/Task Implication Diagram" (Figure 51). Prometheus also suggests a textual goal descriptor—with three lines only: name, description, and subgoals. (We note that in Figure 50, following Prometheus' guidelines, the names of the goals are almost task-like (i.e., verbs). Here, from Figure 48, we argue for state-like names for goals, as shown in Figure 52).

Relationships between goals are captured in MOBMAS's agent-goal diagram (Figure 53) and in the two goal-focussed diagrams of Tropos (Figures 54 and 55). The actor diagram (Figure 54) links actors to goals, whereas the goal

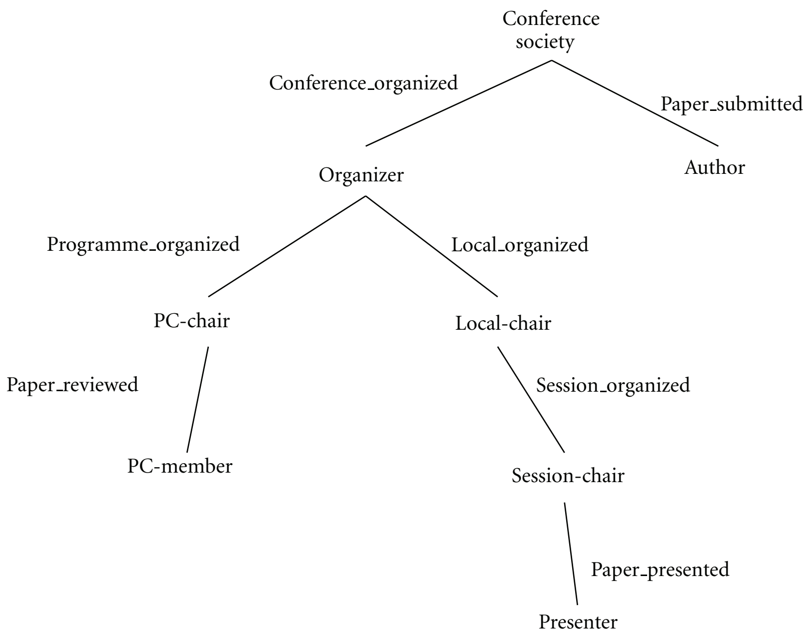

Figure 34: Role dependency diagram as used in OperA for the "Conference Society" (after [52]).

diagram (Figure 55) expands the internal details of the goal itself—here using the i^* notation of Yu [78] which permits discrimination between hard goals and soft goals. (It can be determined whether or not a hard goal has been satisfied; in contrast soft goals do not have well-defined achievement criteria and can only be "satisficed" (e.g., [65, page 207]). Hard goals are associated with capturing functional requirements and soft goals with nonfunctional requirements, e.g., Braubach et al. [120]). The goals, labelled Gx in Figure 53, are also captured in the third partition of the Agent class diagram (Figure 40).

A diagram type that appears to have some ambiguity in terms of its scoping (system versus agent) is found in MaSE, called the "goal hierarchy diagram" (Figure 56) (called Goal Model in OMaSE), and in MESSAGE, where it is called an agent goal decomposition diagram linking goals, tasks, actions, and facts. In both methodologies, it is a relatively simple tree structure where goals are represented as boxes and goal-subgoal relationships as directed arrows from parent to children (MASE) or children to parent (MESSAGE). (Clearly, such directional contradictions form an ideal target for standardization). It is interesting to observe that this would appear to be topologically isomorphic with Graham's [122] task decomposition diagram (using hierarchical task analysis). Indeed, based on our earlier discussion, Figure 56 is, more realistically, either a task (not a goal) diagram or else is a goal diagram with poorly named goals.

Goal hierarchy diagrams are also used in Hermes [123] but associated with agent interactions (Figure 57), and a goal-oriented interaction modelling technique is also introduced into Prometheus by Cheong and Winikoff [124] to replace the previous interaction protocol modelling techniques used in Prometheus.

A more elaborated version of this approach, based on the well-established AND/OR approach to goal modelling, is shown in Figure 58, as presented in OMaSE. The numbers indicate a precedence ordering of the goals, again suggesting tasks rather than goals, although goals are, of course, closely linked to tasks (see Section 6.4), as shown in the metamodel of Figure 48.

Notwithstanding, this pair of concepts ("task" and "goal") are frequently confused. In an etymological analysis of these terms, Henderson-Sellers et al. [101] recommend goals as future-desired states that, when committed to, require the enactment of a task (sometimes called "action") in order to achieve such a desired state (Figure 48). Thus the enactment of a task requires a duration. At the point of time at which this ends, the goal has been achieved (Figure 59). This means that goal names should be state names; that is, nouns, whereas task names should be more verb-like. Splitting up the achievement of a final goal into a set of intermediate or subgoals, as shown here, permits a differentiation (goal/subgoal) that could be seen as commensurate with the action/task differentiation of Figure 60; that is, a goal is achieved by an action, which can be broken down into more granular sections each of which depicts a subgoal being achieved by a task. However, in some methodologies these two terms (goal and task) are equated; that is, used as synonyms. This can often lead to names that are cognitive misdirectional, for example, in such methodologies, the names of goals are typically imperative

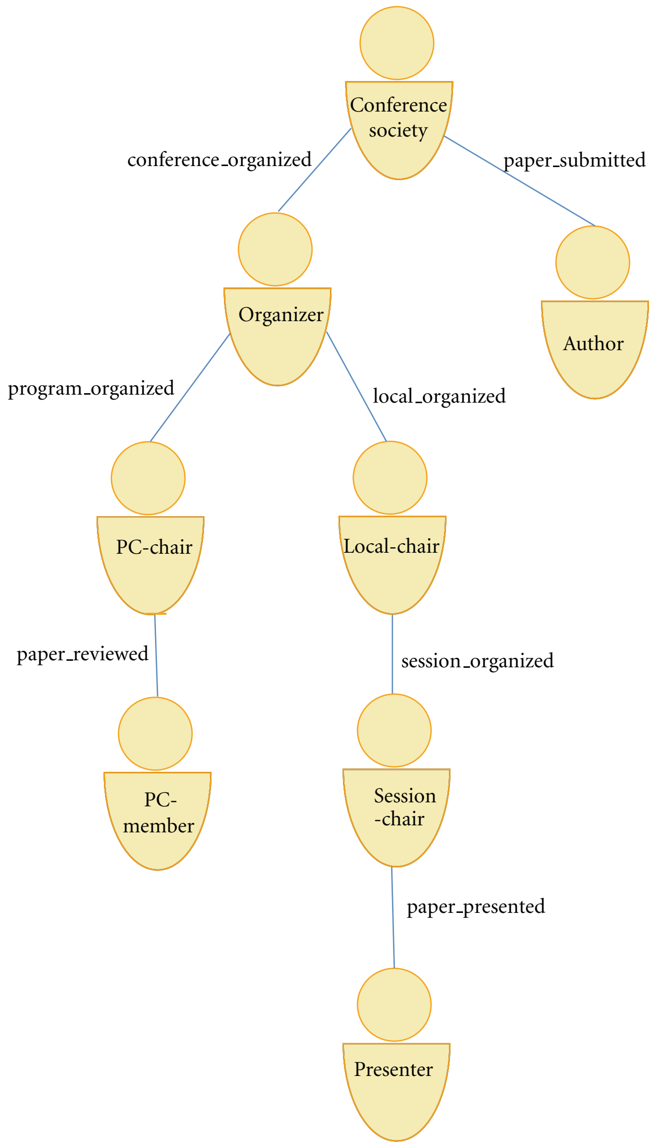

Figure 35: Role dependency diagram depicted using the FAML notation.

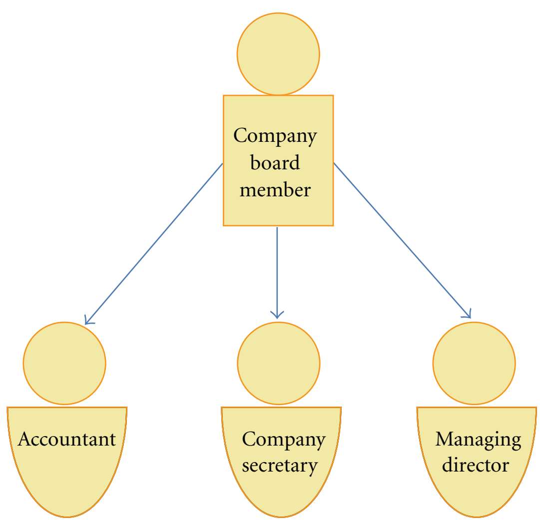

Figure 36: Agent-role coupling diagram depicted using the FAML notation.

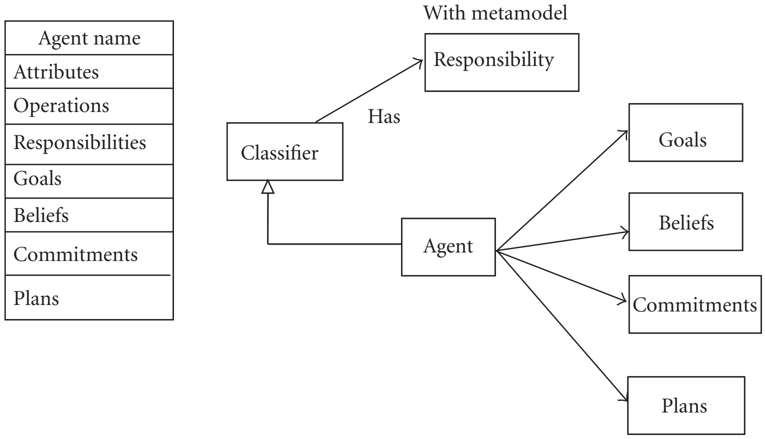

Figure 37: One proposal for extended UML notation for an individual agent type plus the underpinning metamodel fragment (after [91]).

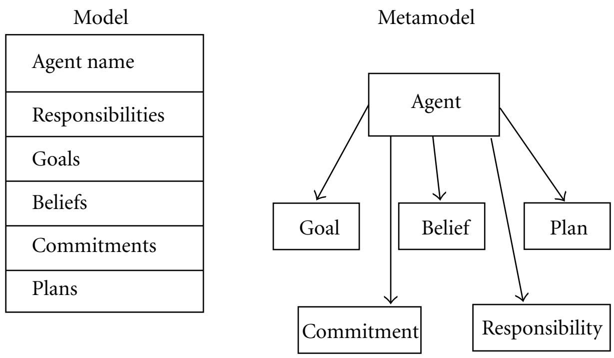

Figure 38: Revised proposal for an agent representation (model-scope symbol plus supporting metamodel fragment).

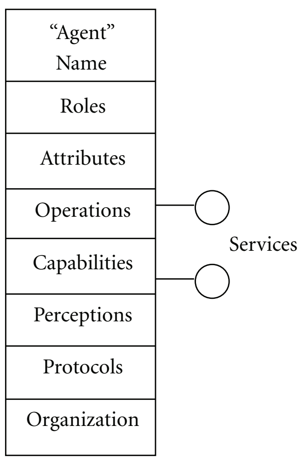

Figure 39: Agent symbol illustrating the kind of information proposed by Huget [92] to show agent attributes.

verb-like which, at first glance, suggests tasks rather than goals. This, therefore, needs to be borne in mind when reading or writing such diagrams.

5.4.2. Ontologies and Plans. Also in this group of recommendations (for static diagram types relevant to agent knowledge) are the ontology diagram and the plan diagram.

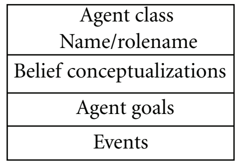

Figure 40: Agent class definition diagram of MOBMAS.

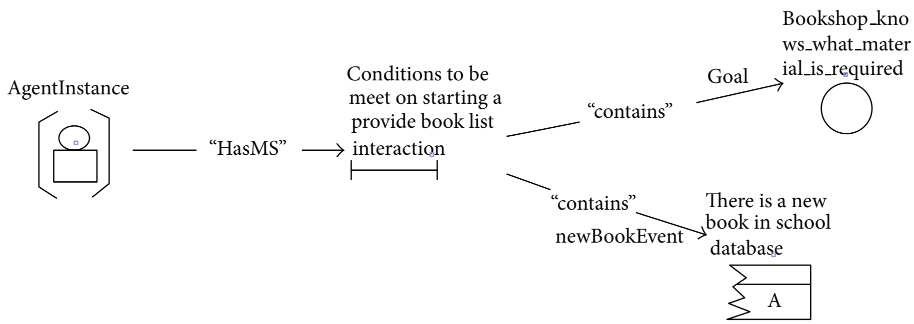

Figure 41: Example of an agent's mental state (which links mental state to facts, beliefs, and events and involves goals, tasks, and roles) as depicted by INGENIAS (after [20], reprinted by permission of the publisher © IGI Global).

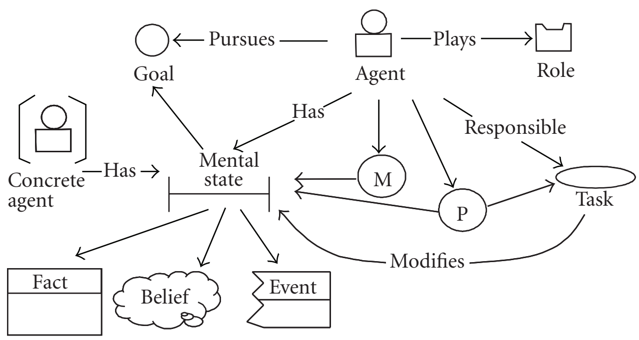

Figure 42: Typical elements in the agent viewpoint and the agent's mental state as depicted by INGENIAS (after [20], reprinted by permission of the publisher © IGI Global).

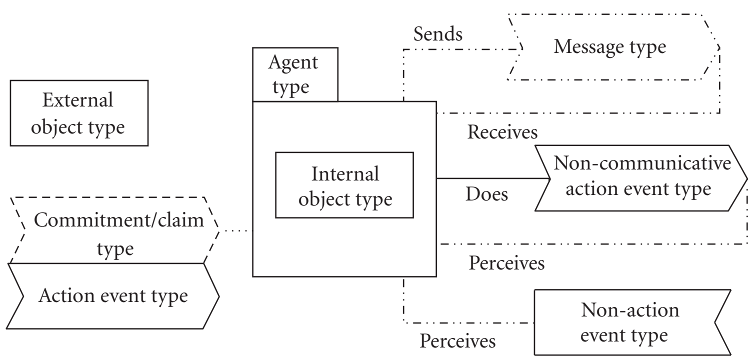

Figure 43: Core mental state structure modelling elements of external AOR diagrams (after [17], reprinted by permission of the publisher © IGI Global).

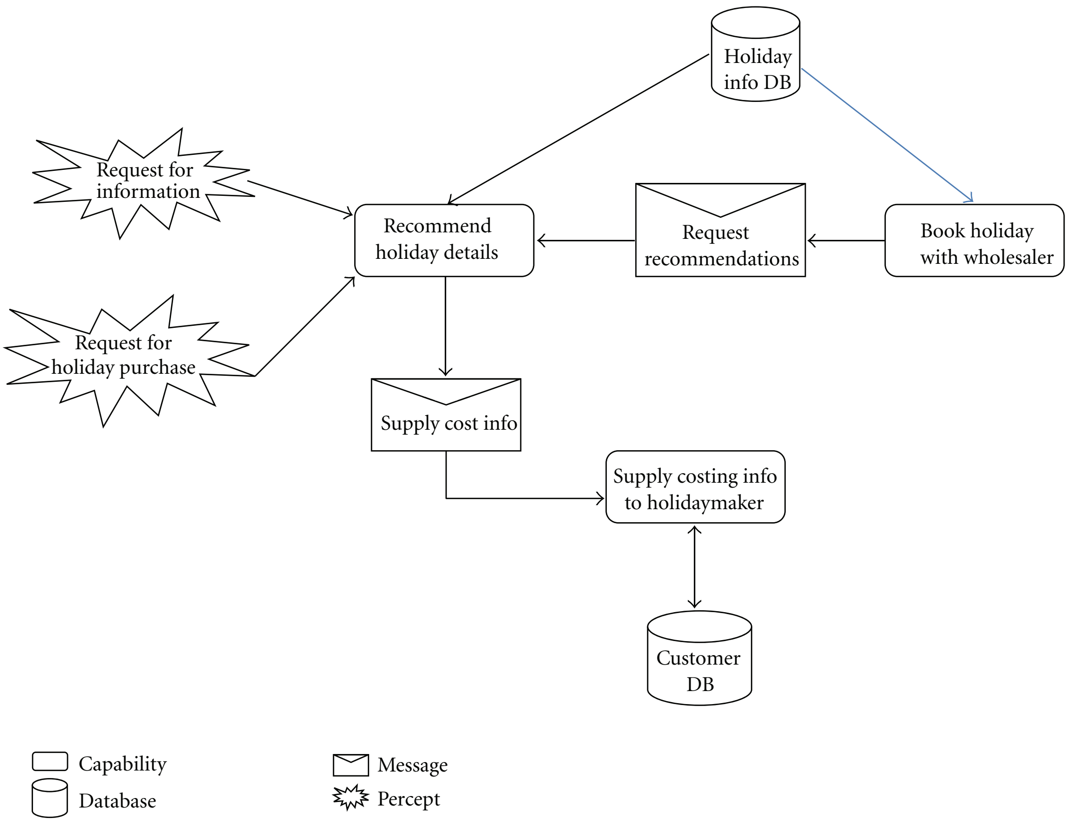

Figure 44: Example of Prometheus Agent Overview diagram showing some of the details of the Travel Agent agent and including percepts.

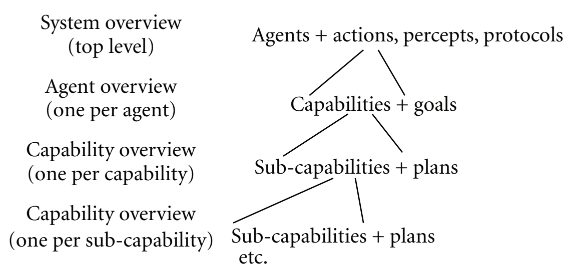

Figure 45: Increasing detail from system overview diagram to agent overview diagram to capability overview diagrams (as envisaged in the Prometheus methodology).

Ontologies are explicit in only a handful of methodological approaches: PASSI and MAS-CommonKADS, MOBMAS and AML [105], and, to a lesser extent, in OperA and ISLANDER. Ontologies, particularly domain ontologies as need here, represent knowledge that is effectively static. It is thus reasonable to depict that knowledge as a fairly standard UML class diagram (Figure 61).

Plans depict the internal details of how a task is to be performed and a goal attained. Plans are typically internal to a single agent and are linked to the tasks (or actions) needed to attain goals (Figure 48) or to the capabilities (Figures 44 and 46). Since the execution of plans may or may not be successful, alternative paths must be included (Figure 62). These alternatives may utilize AND/OR gates, which can be used either in context of an activity diagram or a state transition diagram—depending upon whether the developer wishes to have as his/her prime focus the process or the product aspect. In Tropos, the internal structure of a plan can be summarized as a single node on a Capability diagram (e.g., Figure 46). Plan diagrams may be based on the UML activity diagram as in Tropos or UML STD diagram as in O-MaSE. Mylopoulos et al. [79] show how the Tropos plan diagram can also be depicted using UML notations. Plan diagrams are also used in MOBMAS.

Plan name: Identify location and dates
Description: Ascertain possible holiday places and date
Trigger: Request from client
Context: Holiday not already booked
Data used and produced: Holiday brochures/database
Goal: Recommend time and place(s)
Failure: All likely holidays booked
Failure recovery: Change dates and/or place
Procedure: (1) Search data for location commensurate with client's desire
(2) Check dates holiday is possible
(3) Create list of possible place/date combinations

Box 3: Prometheus-style plan descriptor for the plan to identify holiday location and dates of Figure 46.

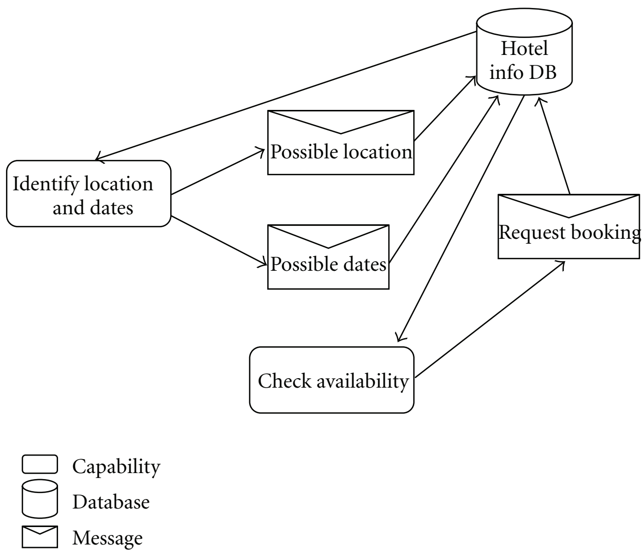

Figure 46: Example of Prometheus Capability Overview diagram for Recommend Holiday Details capability of Figure 44. This also shows a subcapability of identify location and dates, which, in turn, could be depicted graphically by another Capability Overview diagram.

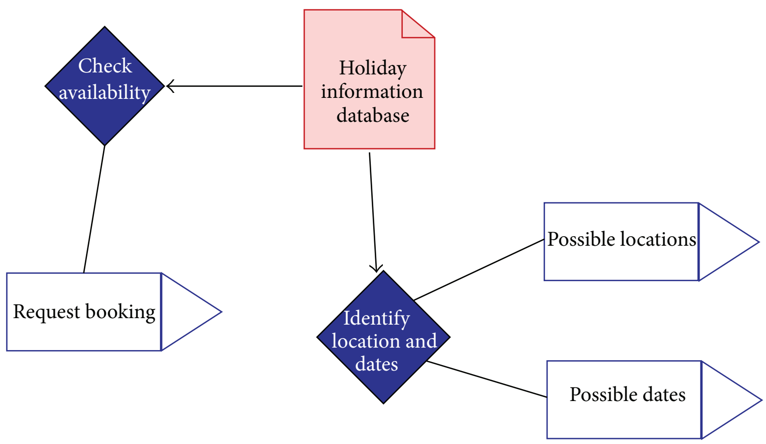

Figure 47: Translation of part of Figure 46 into FAML notation.

Plan diagrams, whether of the activity diagram style or the STD style, can be augmented by text in the form of a plan descriptor (Box 3), as used, for example, in MOBMAS, which defines the plan in terms of initial state, goals, strategies, actions, and events, and Prometheus, which defines a plan in terms of triggering events, messages, actions, and plan steps (a completed example of which is shown in Box 3).

Thangarajah et al. [125] note that there may be several acceptable plans for achieving a single goal such that there is an overlap. This led these authors to formulate mathematical

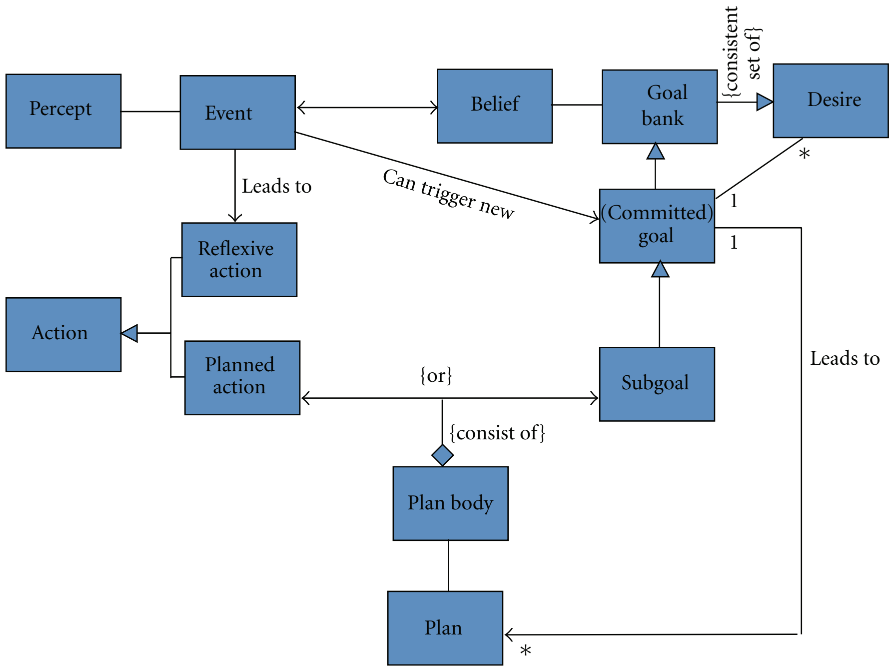

FIGURE 48: Metamodel fragment relevant to goals, tasks, and plans (after [101]).

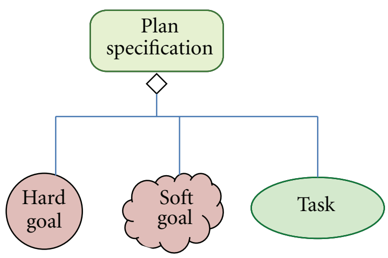

FIGURE 49: Generic model of plans, tasks, and goals conformant to a fragment of the metamodel of Figure 48.

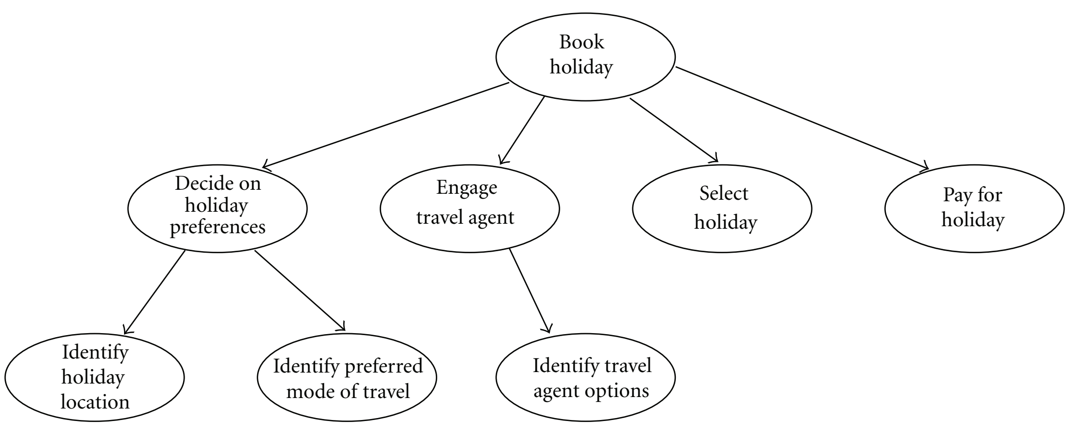

FIGURE 50: Example Prometheus Goal Overview Diagram.

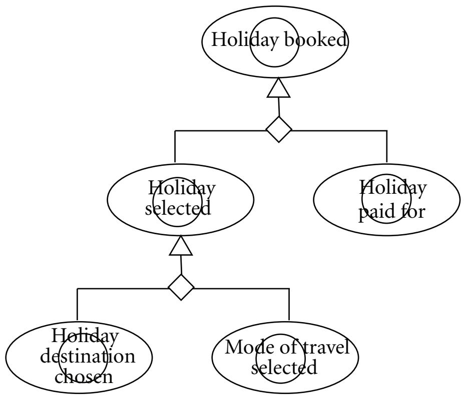

Figure 51: Level 0 Goal/Task Implication Diagram of MESSAGE.

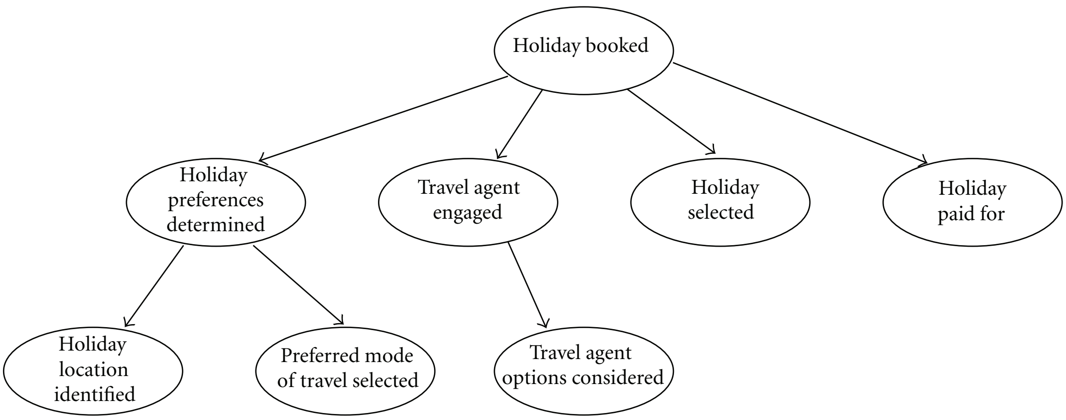

Figure 52: Example of Prometheus Goal Overview Diagram with state-like goal names.

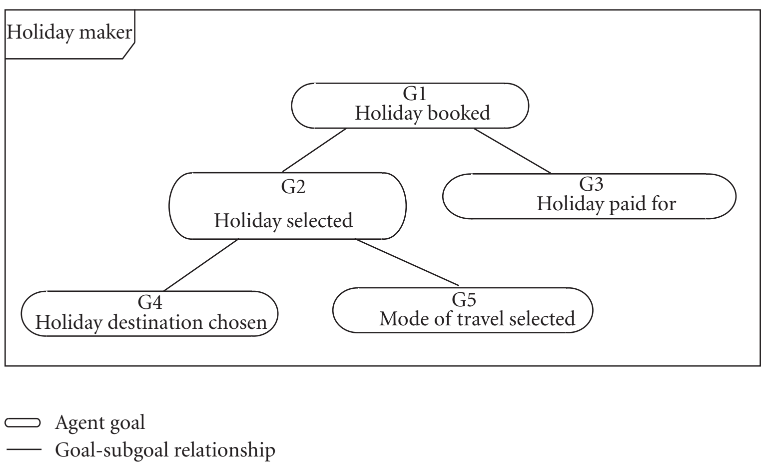

Figure 53: MOBMAS's agent-goal diagram.

expressions for this overlap and also the coverage. Overlap is readily represented using a Venn diagram; a typical goal-plan hierarchy is shown in Figure 63.

5.5. Agent Services View. UML-style class diagram is supplemented by UML-style activity diagrams to show details for each capability to expand a portion of the Capability

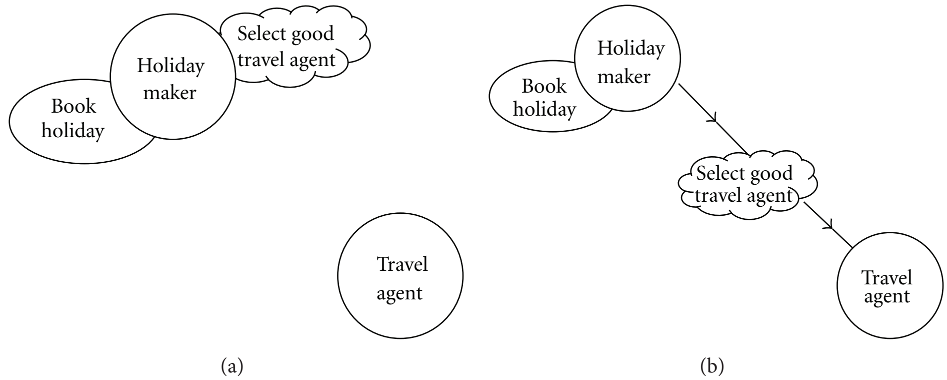

FIGURE 54: (a) Example actor diagram showing goals attached to actors. (b) Example of actor diagram showing an explicit depender (Holidaymaker), dependee (Travel Agent), and dependum (Select good travel agent).

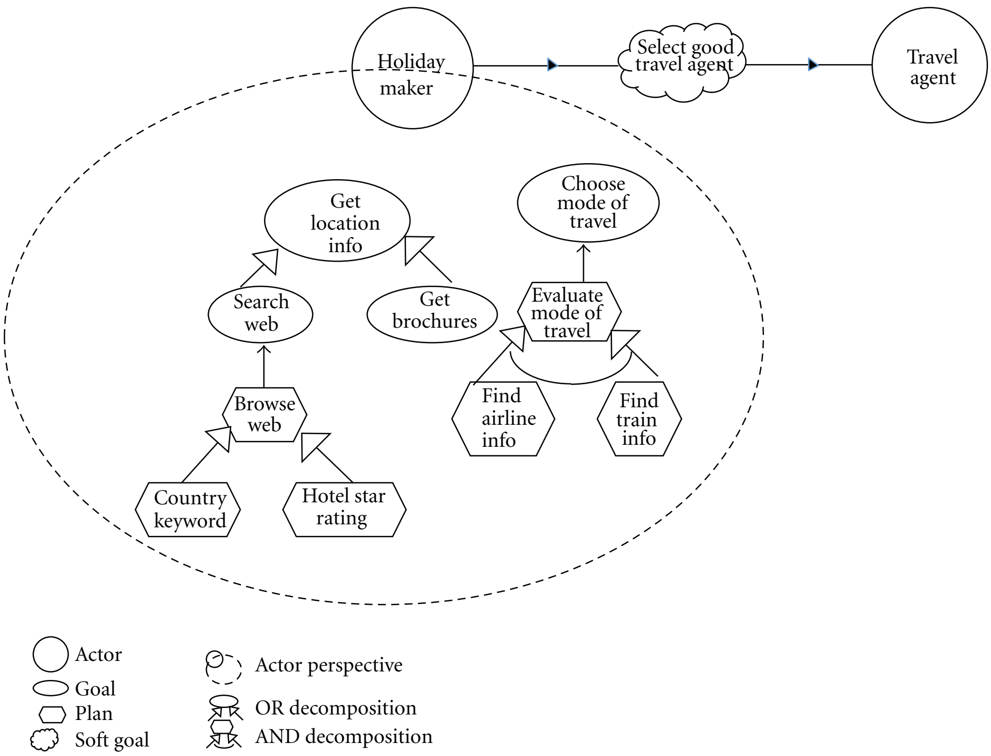

FIGURE 55: Example of a Tropos goal diagram.

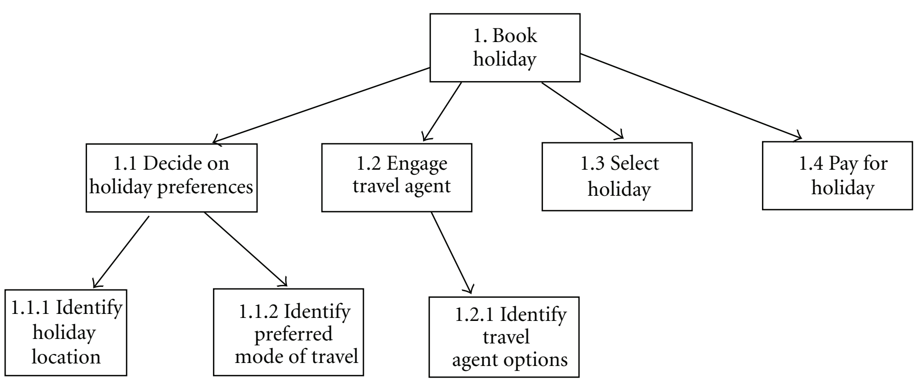

FIGURE 56: Goal hierarchy diagram (MaSE).

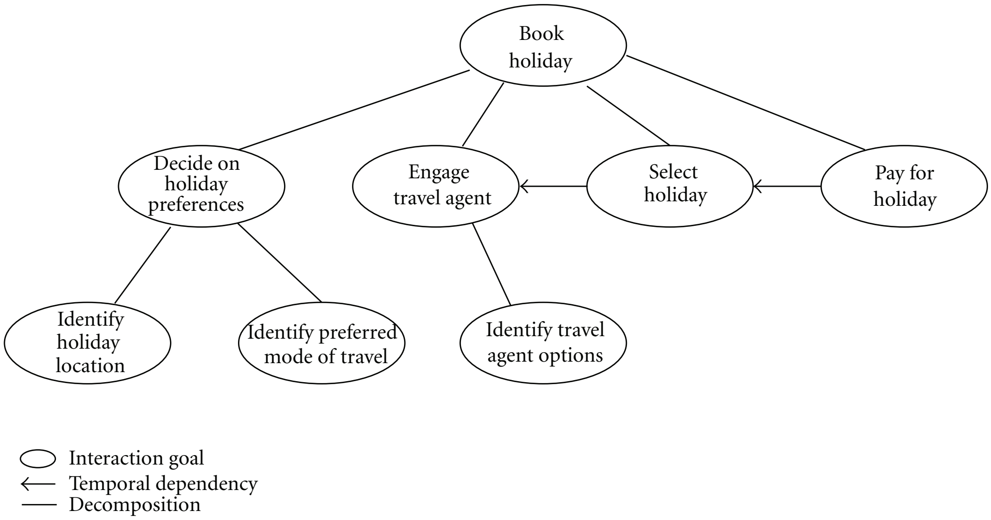

Figure 57: Interaction goal hierarchy of Hermes/Prometheus.

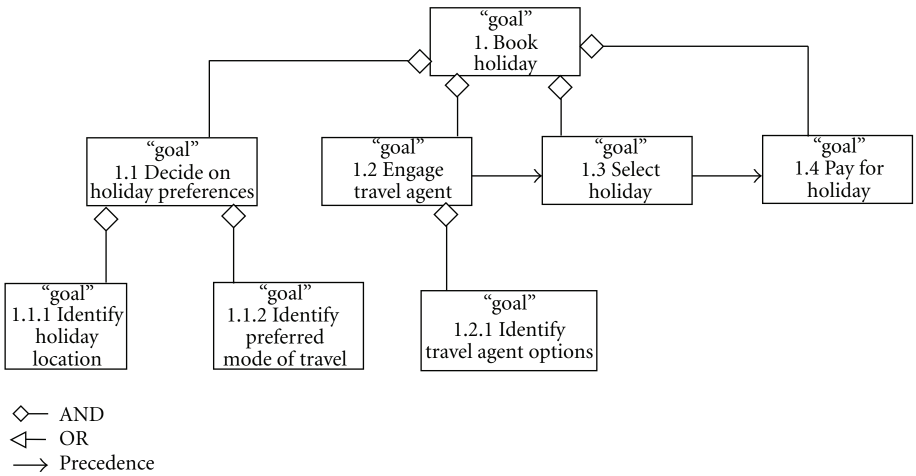

Figure 58: Refined GMoDS goal model using AND/OR decomposition technique (O-MaSE).

Overview diagram of Figure 46 into a more detailed Capability diagram—one diagram for each subcapability. An example, compatible with the Check availability capability of Figure 64, is given in Figure 65 (Prometheus notation). A textual template to accompany the diagram might also be useful to present in textual format details of goals, processes protocols, messages, percepts, actions, capabilities, plans, and data utilized in different ways. Figure 66 shows the alternative use of an Activity Diagram in this context, Capability Diagrams being used in Tropos (Figure 66 for a specific agent) wherein each node may be expanded into a Plan Diagram (see Section 5.4.2).

Services can alternatively be represented directly in either graphical or tabular form, the latter following, for example, the Gaia Service Model, which lists Services and their Inputs, Outputs, Preconditions and Postconditions, the former as depicted in the Level 1 analysis phase of MESSAGE [48, page 186], wherein a service is realized by a partially ordered set of tasks (see later discussion of Figure 81). Direct representation of service protocols is found also in AML (see later discussion of Figures 76 and 77) and in the Service Model of ISLANDER [35].

5.6. Deployment View. Henderson-Sellers [19] recommends using a fairly standard UML (or AUML) deployment diagram. MaSE uses a similar diagram (Figure 67) but notes that it differs from the standard use of the UML Deployment Diagram since

(i) the three-dimensional boxes represent agents in MASE, whereas they represent (hardware) nodes in UML,

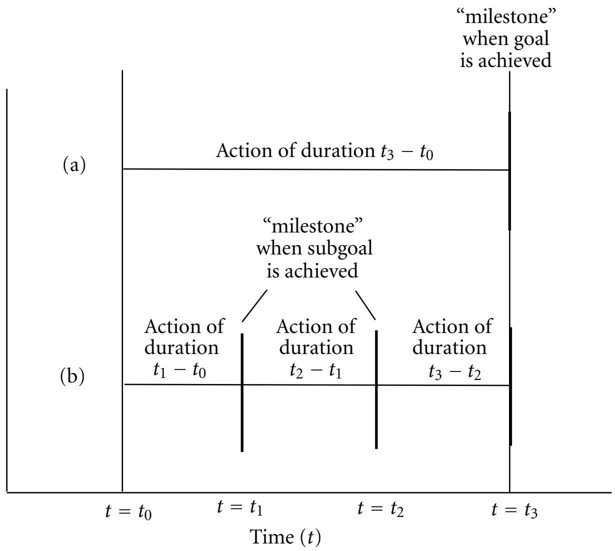

Figure 59: Milestones, subgoals, and goals: (a) a single action attains the goal or (b) several actions are needed, each achieving a subgoal (after [101]).

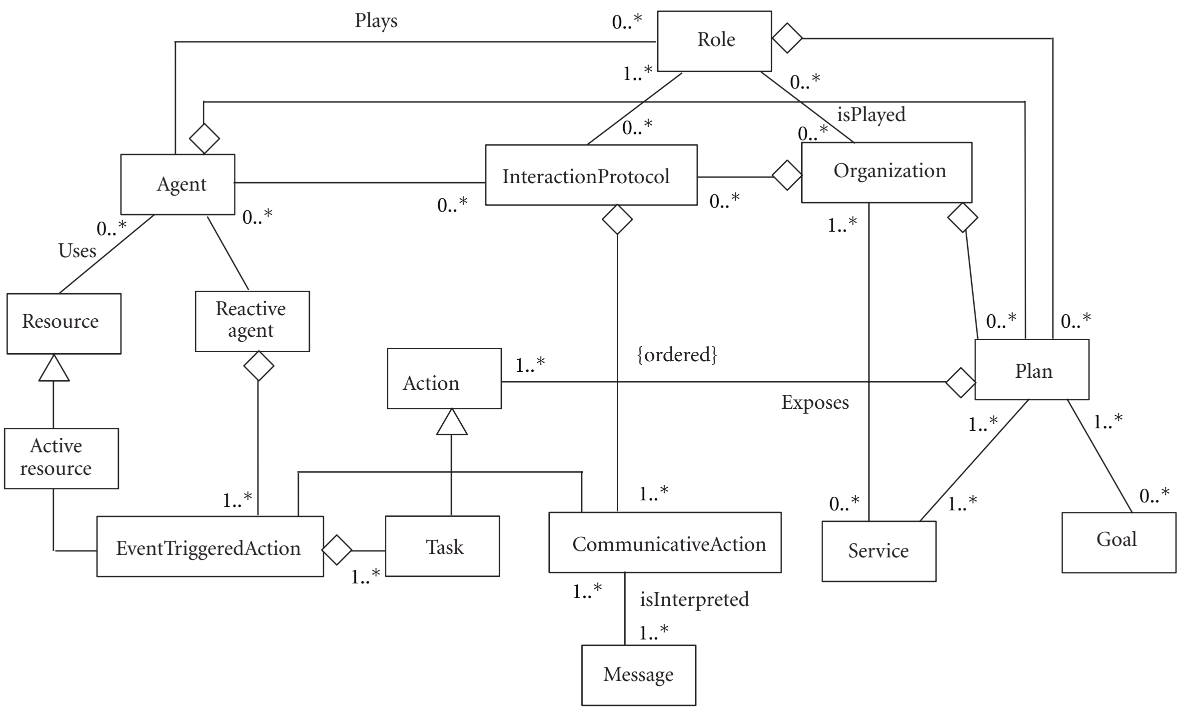

Figure 60: The action perspective of the metamodel of Azaiez et al. [108], (reprinted from [108], copyright 2007, with permission from IOS Press).

(ii) the connecting lines represent conversations between agents in MASE whereas in UML they represent physical connections between nodes,

(iii) MASE uses dashed-line box around agents to indicate that these agents are housed on the same physical platform.

Other methodologies adopting this UML style of deployment diagram include PASSI and RAP/AOR. Most other approaches neglect it. Whilst not employing such a diagram, Tropos does discuss implementation issues as related to its use of class diagrams (Figure 68). SODA only hints at implementation in terms of their environment model, preferring to

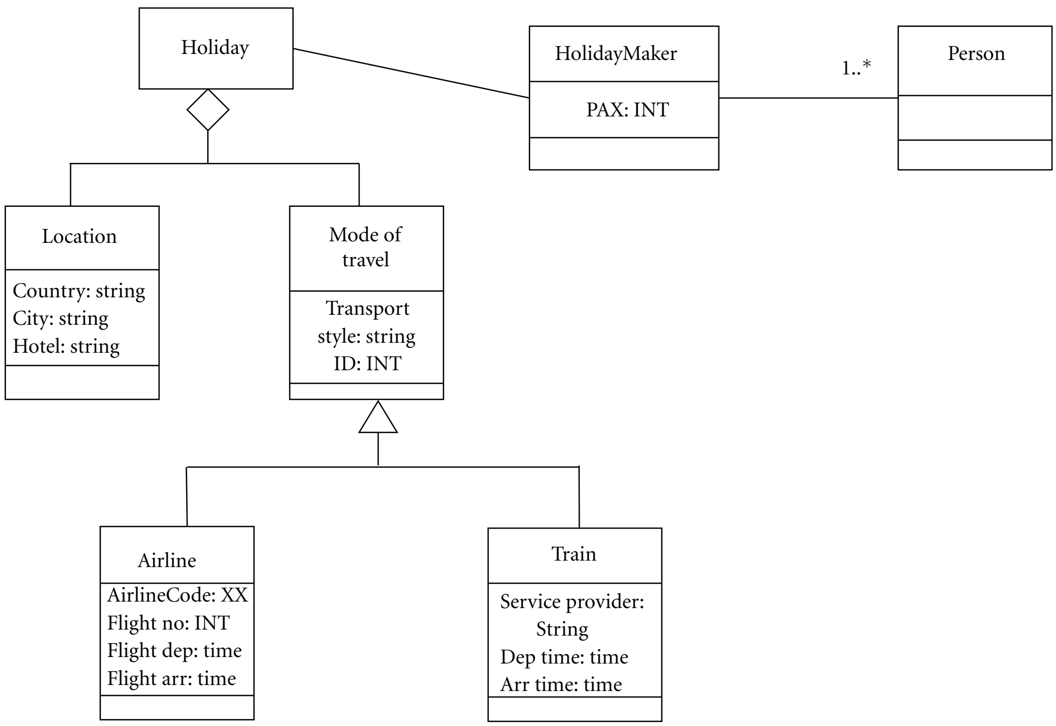

Figure 61: Ontology diagram as used in MOBMAS.

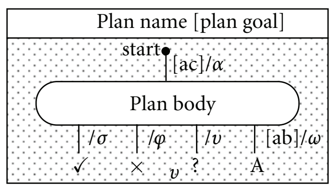

Figure 62: Plan diagram in which ac is the activation condition and α is the activation action. Stop states are labeled as success states "✓" (success action "σ"), fail states "×" (fail action "φ"), unknown states "?" (unknown action "υ"), or abort states "A" (abort condition "ab"; abort action "ω") (after [116], reprinted by permission of the publisher © IGI Global).

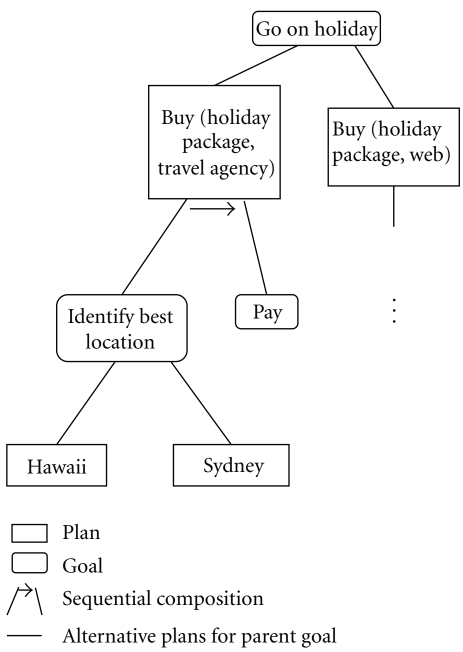

Figure 63: An example goal-plan hierarchy diagram using the notation proposed by Shapiro et al. [117].

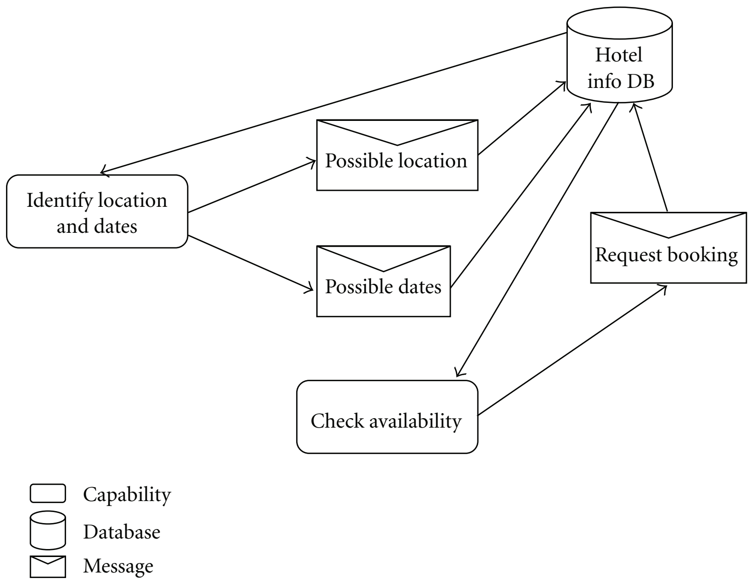

Figure 64: Duplicate of Prometheus Capability Overview diagram for Recommend Holiday Details capability of Figure 44 in comparison with Figure 65.

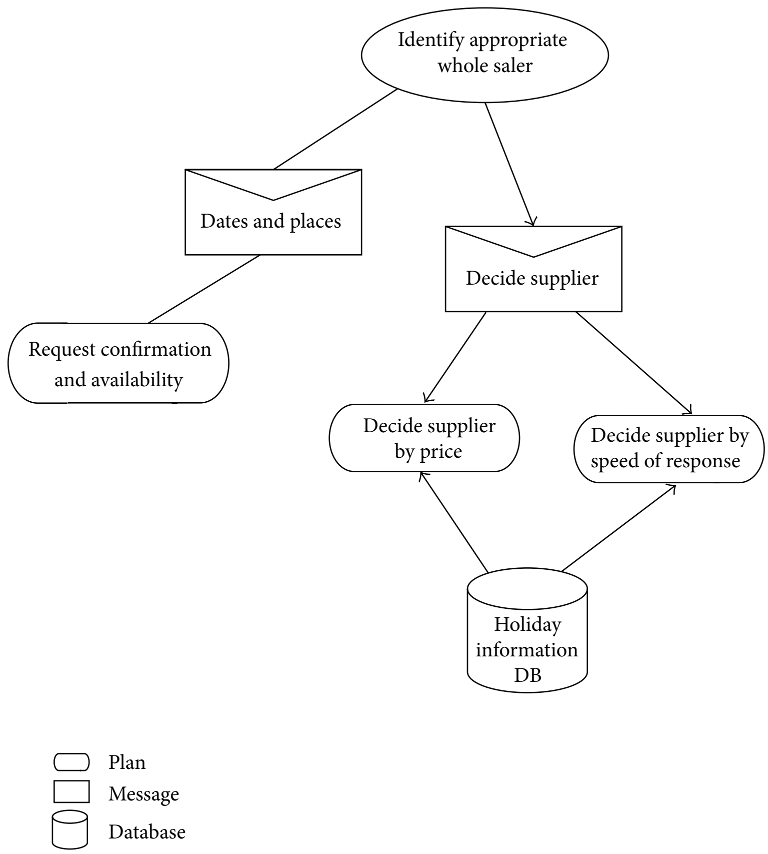

Figure 65: Example of Prometheus Capability diagram showing the plans for the Check availability capability of the Travel Agent agent shown in Figure 64.

defer such details to specific implementation methodologies. PASSI and O-MaSE seem to be the only methodologies that discuss implementation issues directly (i.e., at the code level).

5.7. UI View. Henderson-Sellers [19] noted the lack, in published AOSE methodologies, of any diagrams relating to the user interface. He therefore recommended adding (at least) a

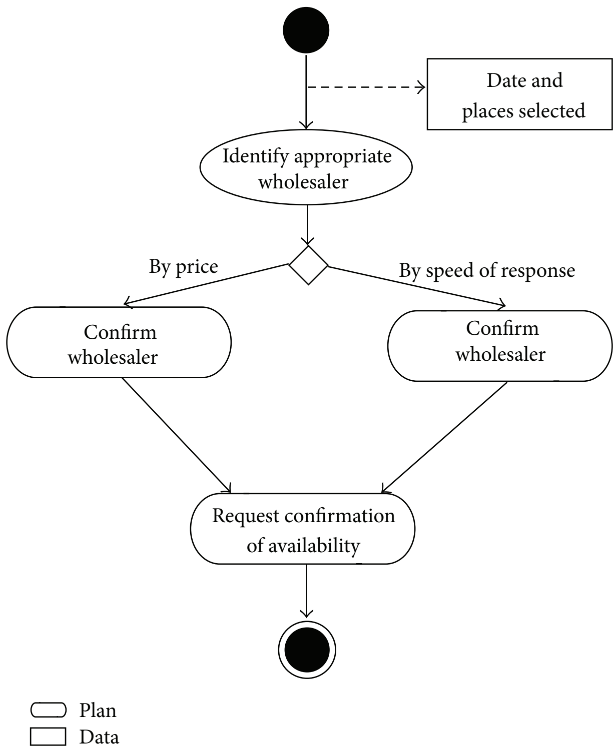

FIGURE 66: Example of a Tropos-style capability diagram for the capability of Check availability of Figure 64.

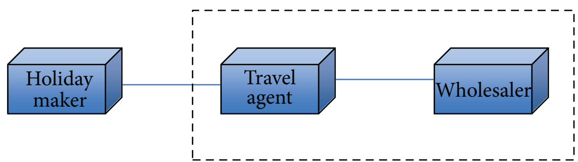

FIGURE 67: Example deployment diagram using MaSE notation.

UI design diagram, which could likely be represented using a semantic net (Figure 69). Henderson-Sellers [19] offers this as a placeholder; that is, a generic "UI design" diagram type pending future empirical work and utilization of the visual design theories of, for example, Ware [127] and the insights of Graham [122] and Constantine and Lockwood [128] (see also http://www.foruse.com/). This latter book also recommends user role maps and structure role models (where "role" refers to human roles in the software development process). These authors also provide heuristics on menu design, the use of iconic interfaces, and other more innovative interface design approaches. The topic was also explored in a non-AOSE context by Gonzalez-Perez [129].

Other suggestions in the literature, although sparse, include a placeholder for a UI prototype in ADELFE's (http://www.irit.fr/ADELFE/) Activities 8 and 9 (see also Jorquera et al. [130] who discuss UI prototyping as a work unit but do not offer a notation for the resultant work product).

6. Dynamic Diagram Types

6.1. Agent Societies View. Agent behaviour is usually depicted in terms of agent-agent interactions, including message passing. A typical agent-oriented interaction diagram also shows the order of these messages needed to effect a single service. Consequently, a standard AUML [75] or AML [73] interaction diagram can be used as the basic interaction diagram (a.k.a. conversation diagram) (Figure 70). Further addition of more formal protocol information to the basic conversation diagram, again using, say, AUML or AML as the notation, would result in a protocol diagram. Prometheus optionally enhances these interaction diagrams with percepts and actions; that is, messages to and from an invisible timeline/agent. In addition to percepts, input from the environment and other events, including those self-generated, for example, by a clock, can be shown [131]. Events typically generate an action within the agent. The result is really

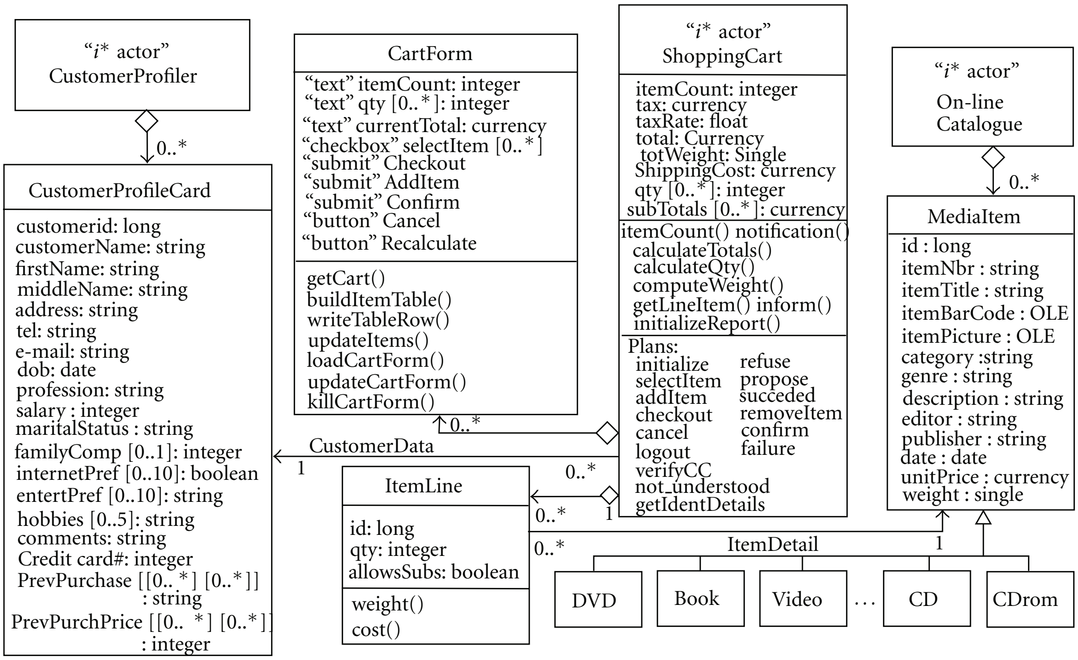

Figure 68: Partial class diagram for a Tropos implementation in their Store Front case study (after [67], reprinted by permission of the publisher © IGI Global).

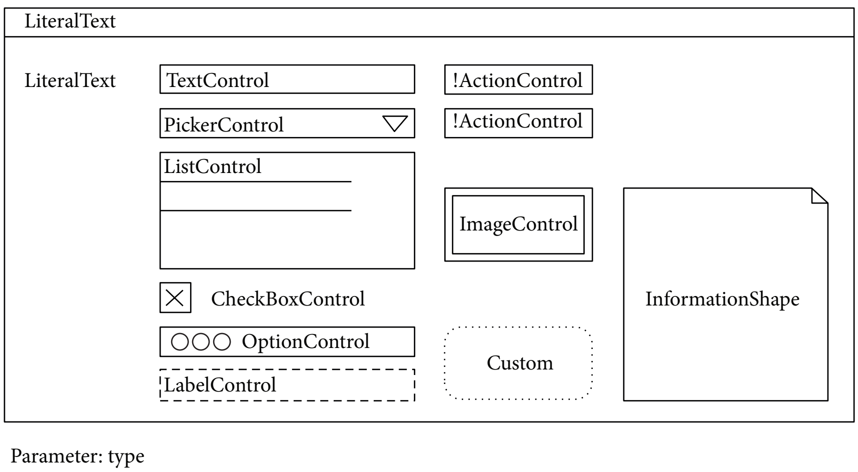

Figure 69: An OPEN/Metis user interface sketch (after [126]).

highly notation dependent but with AUML might look like Figure 71 and with AML it would look like Figure 72. This can then be supplemented by protocol descriptors and message descriptors. In Prometheus, Padgham and Winikoff [59] recommend fields for the protocol descriptor as Description, Scenarios, Agent names, a list of Messages in the protocol, and a final field to contain other miscellaneous information. For the Message descriptor, they recommend a natural language descriptor, the source and target of the message together with a statement on its purpose, and the information carried by the message.

Behavioural diagrams used in AOSE often adopt (or adapt) one of the two basic kinds of UML interaction diagram: sequence charts and collaboration diagrams (said to be semantically equivalent), the latter renamed as Communication diagram in UML Version 2.

Sequence-style diagrams are used in MaSE (e.g., [44]), PASSI, ADELFE, and MOBMAS as well as in OperA and

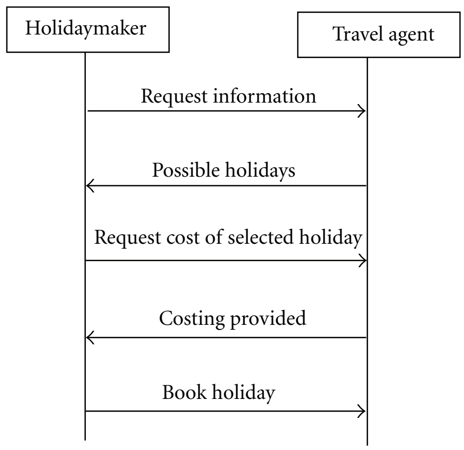

Figure 70: Interaction diagram between Holidaymaker agent and Travel_Agent agent.

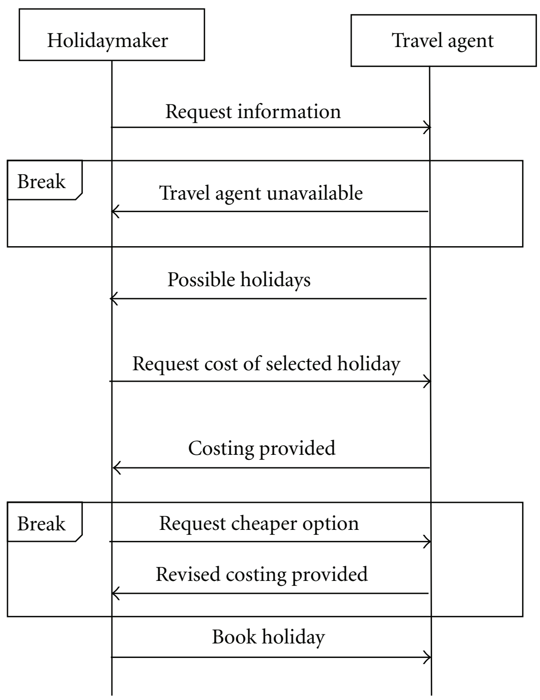

Figure 71: Example of Prometheus protocol diagram equivalent to Figure 70.

Agent Factory (Figure 73), where it is called a "protocol" and a "protocol model," respectively, and in MESSAGE (Figure 74) and Tropos (Figure 75). MOBMAS uses a slightly different version of an AUML sequence diagram in which ACL messages are replaced by tuples.

A specialized form of the AML Interaction Protocol Diagram (Figure 72) is the Service Protocol Diagram (Figure 76), used only within the context of the service specification.

Whilst not a methodology, MAS-ML [71] uses an extended UML Sequence Diagram to show interactions and their sequencing. These authors use this approach to depict the modelling of plans and actions, of protocols, and of role commitment.

Collaboration-style diagrams are used in Agent Factory, CAMLE, and AML (Figure 77) but seldom elsewhere wherein the sequence-style diagram is the preferred option.

6.2. Workflow View. Workflows reflect agent behaviour so that a standard workflow diagram would seem appropriate. UML-style activity diagrams are used in Prometheus to illustrate processes within an agent, including allusions to

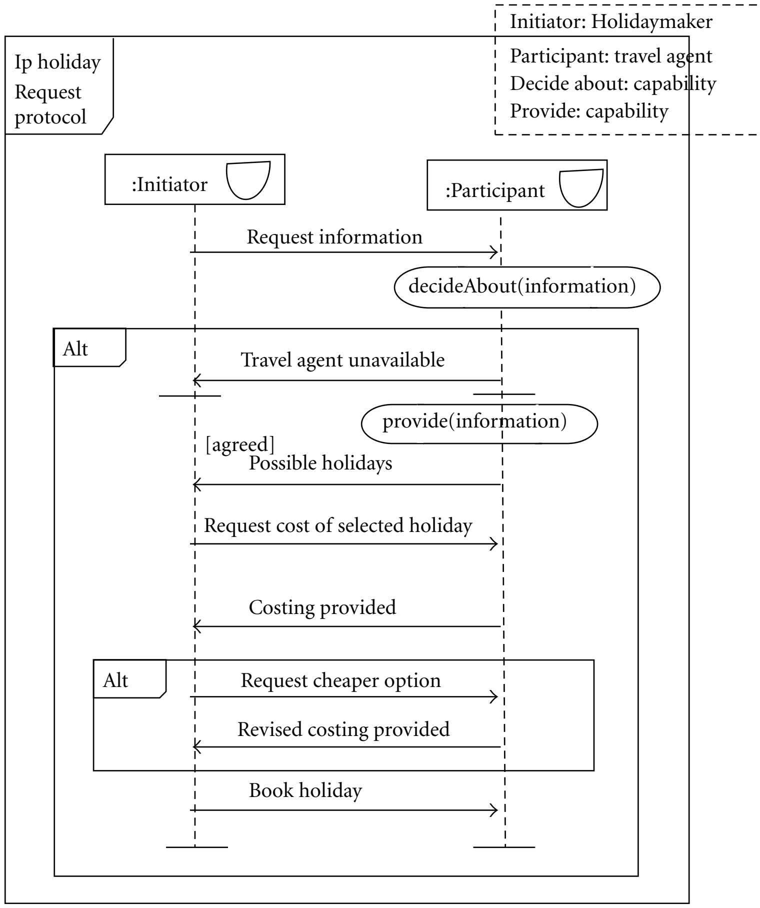

Figure 72: Example of AML Interaction Protocol as a sequence diagram.

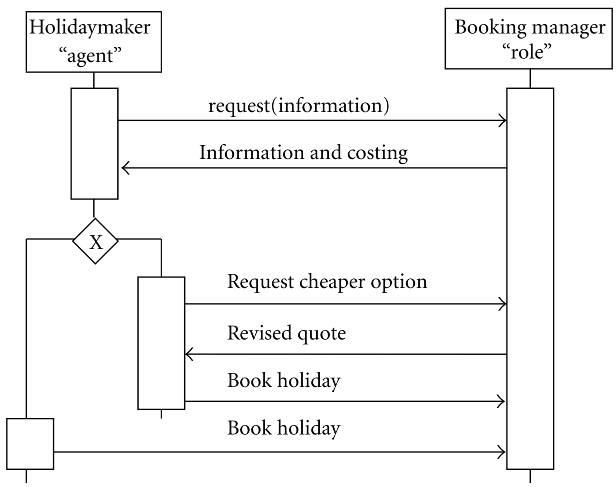

Figure 73: Example Protocol Model as used in MaSE, Agent Factory, OperA, PASSI, and ADELFE.

interactions with other agents. Figure 78 shows an example of a Prometheus Process Diagram. In this diagram, details of a process are shown together with interactions with other agents; these are depicted minimalistically by a single (envelope-shaped) icon. A Process Diagram can then be supplemented by a process descriptor listing activities, triggers, messages, and protocols. Whilst these diagrams show the higher level view, details can be presented textually

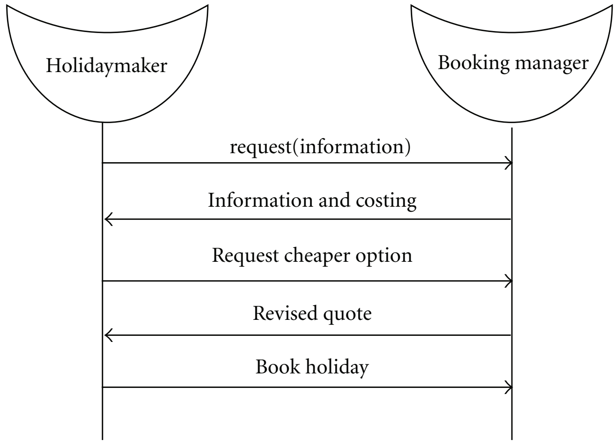

Figure 74: Example of an MESSAGE interaction protocol diagram.

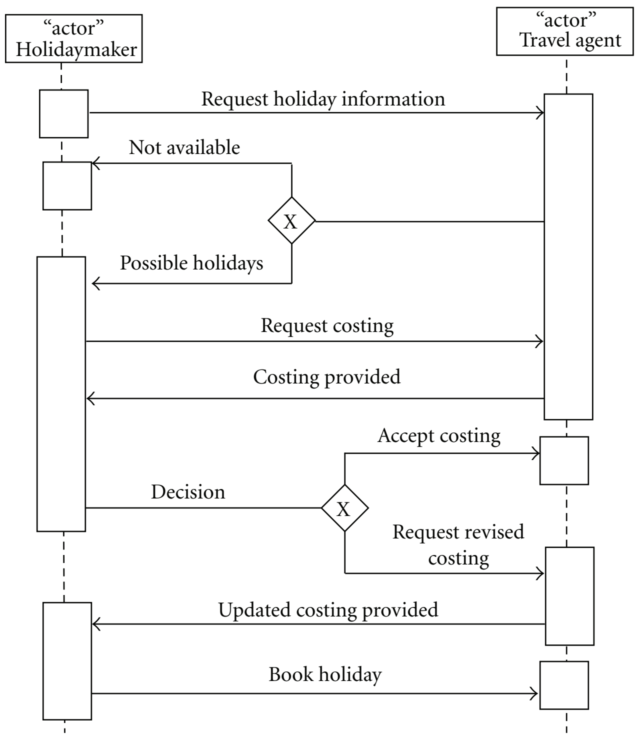

Figure 75: Example agent interaction protocol as used in Tropos.

[132]: for percepts, actions, and events. The percept descriptor lists the information gleaned by the agent in its interaction with the environment, whereas the action descriptor depicts the effect of the agent on the environment. An event descriptor defines an event in terms of its purpose, together with the data that the event carries. (For details of these templates, see [59, Chapter 7]).

UML-style activity diagrams are also used in Agent Factory, called there an “activity model.” These illuminate all the “activity scenarios,” wherein each swimlane represents the processing of a role involved in the scenario. In PASSI, a similar use is made of swimlanes to specify the methods of each agent or of each agent’s task.

Workflows are also represented explicitly in INGENIAS (Figure 79) using the notation shown in Figure 80. In this approach, a workflow is created from the tasks identified in the interaction specification diagram together with the goal/task view (see later discussion of Figure 84). Similarly,

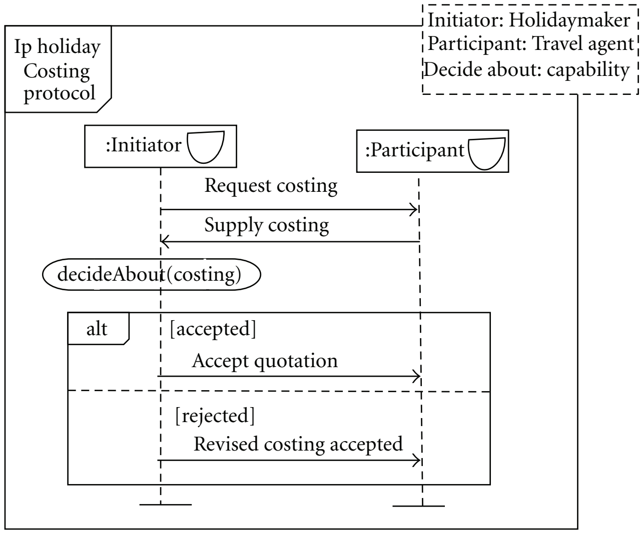

Figure 76: Example of AML Service Protocol as a sequence diagram.

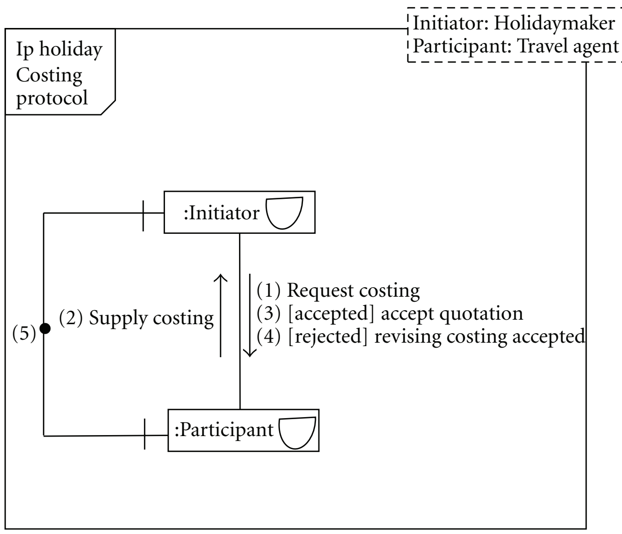

Figure 77: Example of AML Service Protocol as a communication diagram showing the same information as Figure 76.

MESSAGE depicts a workflow in terms of a partially ordered set of tasks that realize a service (Figure 81). AOR also uses workflow as the basis for its activity diagram (Figure 82).

MAS-ML [71] includes extensions proposed to the UML Activity Diagram in order to depict "a flow of execution through the sequencing of subordinate units called action." In this way, the authors can model plans and actions; goals; and messages, roles, organizations, and environment (for full details, see [133]).

6.3. Agent Knowledge View. The dynamic aspects of agent behaviour are addressed in several methodologies, both in terms of interagent behaviour and single agent behaviour. This is exemplified in MaSE's "communication class diagram" a.k.a. conversation diagram [43], based on the notation of a UML State Transition Diagram. It should be noted that this diagram type focusses on the states of an agent *during a particular conversation*. This means that for a conversation between two agents (initiator and responder) two state diagrams are required. Actions specified within a state represent processing required by the agent.

STD-style diagrams are recommended by PASSI, MESSAGE (Figure 83), and MOBMAS. Also with a slightly different visualization is the state machine-focussed Interaction Structure of ISLANDER, which shows dialogues called scenes. These then define protocols.

6.4. Agent Services View: Task Diagrams. Tasks represent the dynamic counterbalance to goals. Indeed, they are closely linked, as shown in the metamodel of Figure 48 and in INGENIAS (Figure 84). Indeed, in the SONIA methodology, a task model is one of the first to be developed, although

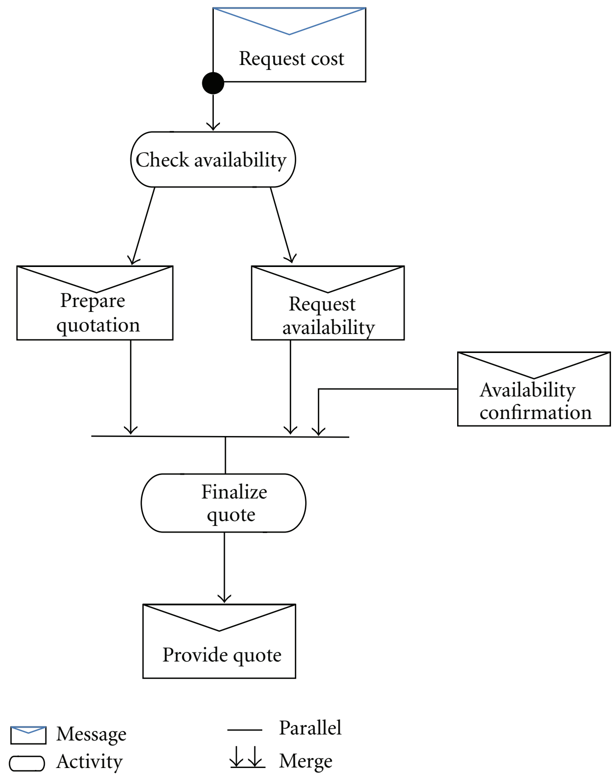

FIGURE 78: Example process diagram for the Request cost functionality.

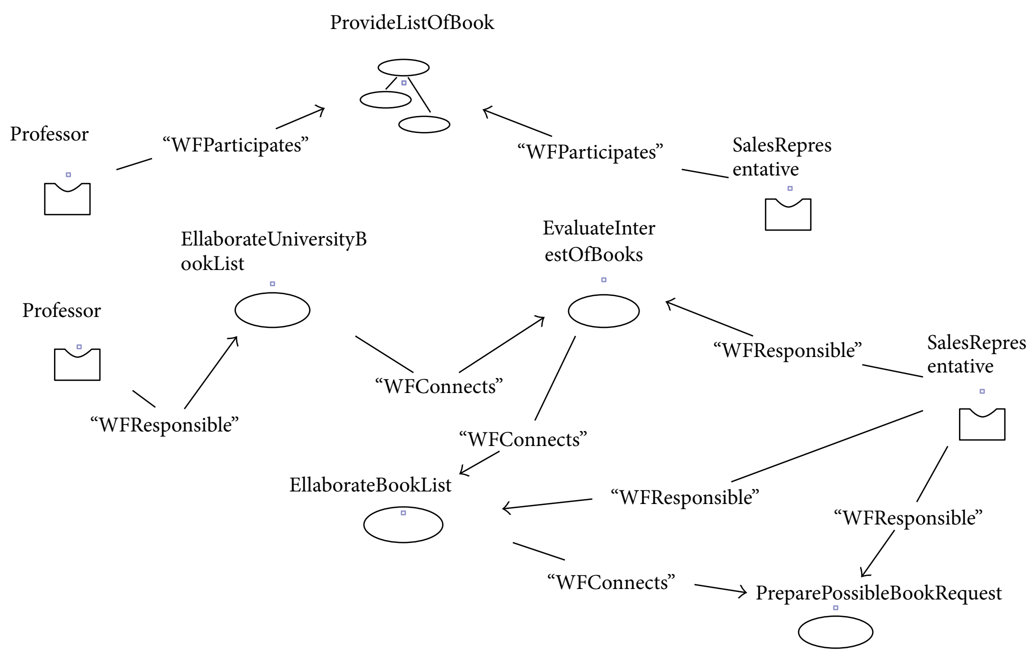

FIGURE 79: Example of workflow as used in INGENIAS (after [20], reprinted by permission of the publisher © IGI Global).

goals are not considered until later, in a Goal Model. Tasks are accomplished by the enactment of a Plan (see Section 5.4.2).

Several AOSE methodologies address task descriptions and task decomposition, for example, using a UML activity diagram with two swimlanes (Figure 85) or incorporating AND/OR task decomposition (Figure 86) to create a task hierarchy. Both diagram styles can be supplemented by Task templates. For the current state(s) of any task, a standard UML-style Statechart diagram can be used.

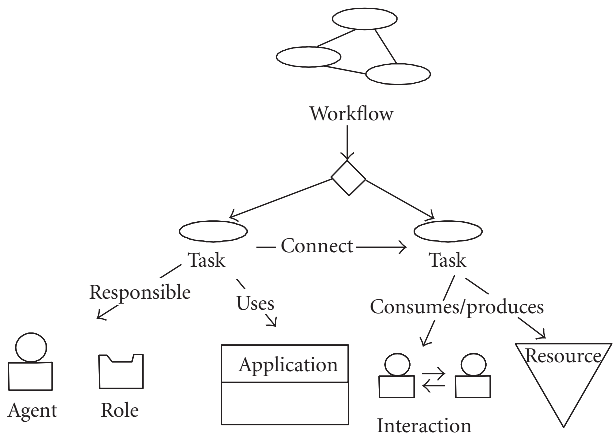

Figure 80: Typical elements in an INGENIAS's workflow (after [20], reprinted by permission of the publisher © IGI Global).

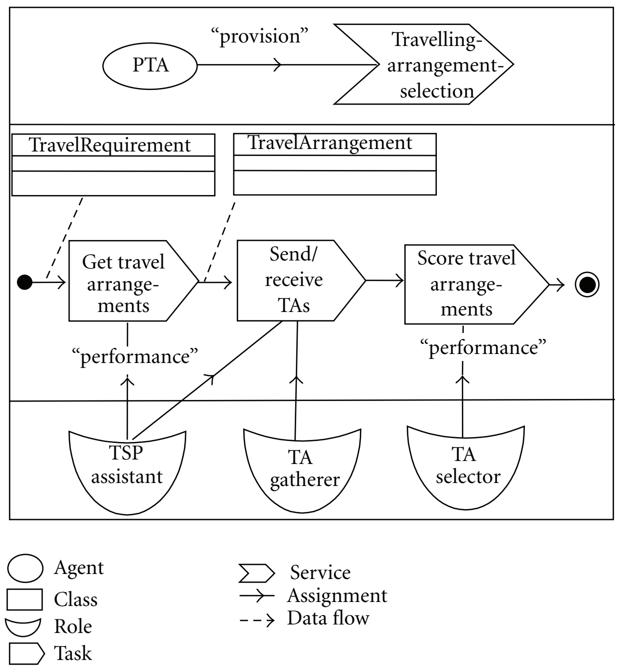

Figure 81: Task workflow from MESSAGE showing how a service is implemented by a series of tasks (after [49], reprinted by permission of the publisher © IGI Global).

A task model is also included in the diagram suite of MAS-CommonKADS. It consists of two elements for each task: a "task hierarchy diagram" and a "task textual template." Peyravi and Taghyareh [68] suggest the addition to MAS-CommonKADS of a standard activity diagram to represent the activity flow of a task together with a textual template for each individual task within the activity flow, together with a tabular rendering of task knowledge. Task models are also found in MaSE and in ISLANDER.

Interestingly, Fuentes-Fernández et al. [134] investigate the ideas of Activity Theory [135] as applied to AOSE. They propose mapping these activities to tasks and possibly to workflows or interactions. This is clearly a topic for further discussion beyond the standard methodological use of diagram types as outlined here.

6.5. UI View. In Section 5.7, we noted the possibility of introducing, as a static diagram type, a UI design diagram. There is also a need for a dynamic view on the UI. Gonzalez-Perez [126] suggests a service state diagram (Figure 87) to represent the various states of the UI and linking this directly to UI design diagram of Figure 69. Constantine

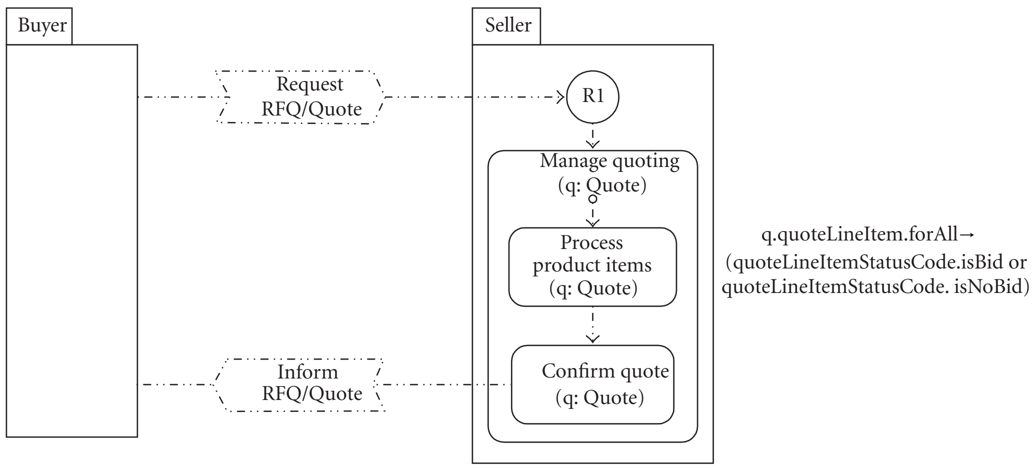

Figure 82: Incomplete AOR activity diagram for the quoting business process type from the perspective of the Seller agent (after [17], reprinted by permission of the publisher © IGI Global).

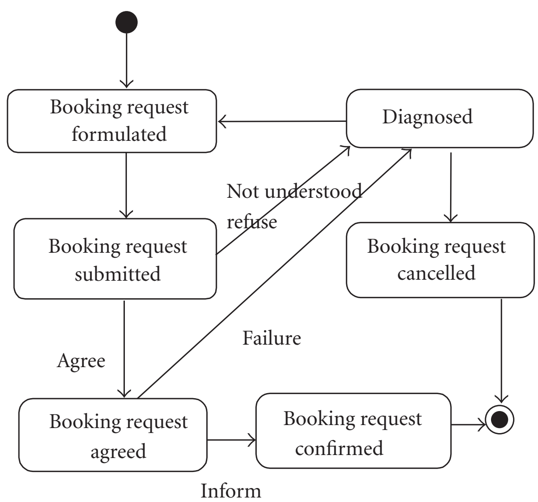

Figure 83: State chart in MESSAGE for a Booking Manager role (after [49], reprinted by permission of the publisher © IGI Global).

and Lockwood [128] also recommend the utilization of task modelling, essential use cases, and context navigation maps to describe the dynamic aspects of user interface design.

7. Discussion and Related Work

Recommendations for a standardized agent-oriented modelling language are still indeterminate with several approaches being investigated. These include use of many UML diagrams with little or no change (e.g., [20, 37, 49, 59, 136]) or bundled as a UML profile (e.g., [17, page 286]). Formal proposals to create an agent-oriented extension to UML include AUML [74, 137, 138] and AML [73]. They provide all the fine detail of the UML, also making some recommendations for diagram types in passing. Here, we have taken the results of the deliberations of Henderson-Sellers [19] regarding appropriate diagram types that could be recommended as part of a future standardized agent-oriented modelling language and have investigated a wider range of examples from the literature. In addition, we have introduced the FAML notation in order to see whether (i) the recommended diagram types can be visualized using this notation and (ii) there are any deficiencies in the FAML notation.

Our discussion here has focussed solely on the suite of diagram types, whilst recognizing that there are two other major elements of any modelling language: notation of individual atomic elements (e.g., [2, 100]) and the defining metamodel (e.g., [1]). In the latter case, there have been attempts to combine existing metamodels synergistically for (a) work products (e.g., [4, 71, 139]) and (b) method elements (e.g., [3, 140–142]).

In the A&A (agents and artifacts) approach [143], the three basic categories identified of agent, society, and environment align well with three of the views presented here (Tables 3 and 4). The aim of these authors is to be able to manipulate agent societies and the MAS environment as first-class entities. Their utilization of a multidisciplinary approach including speech act analysis [144] and activity theory [135]

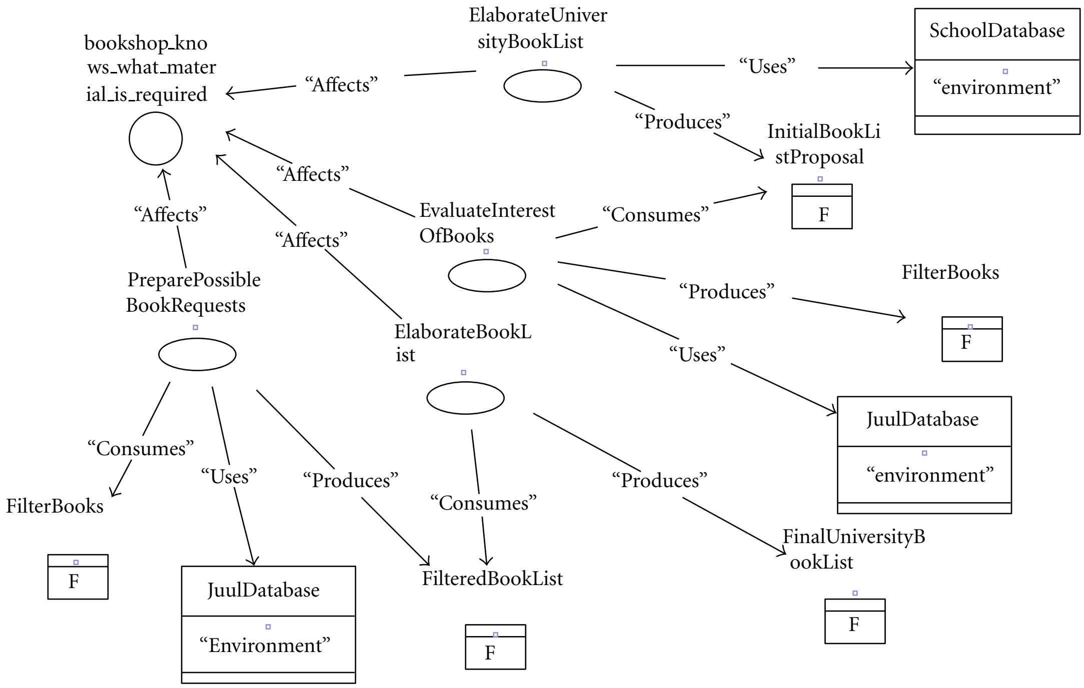

Figure 84: Goal-task relationships in INGENIAS (after [20]). For notation see Figure 17, it is reprinted by permission of the publisher © IGI Global.

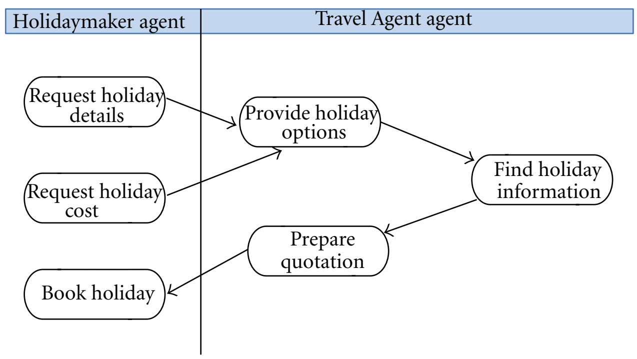

Figure 85: Example of PASSI task specification diagram.

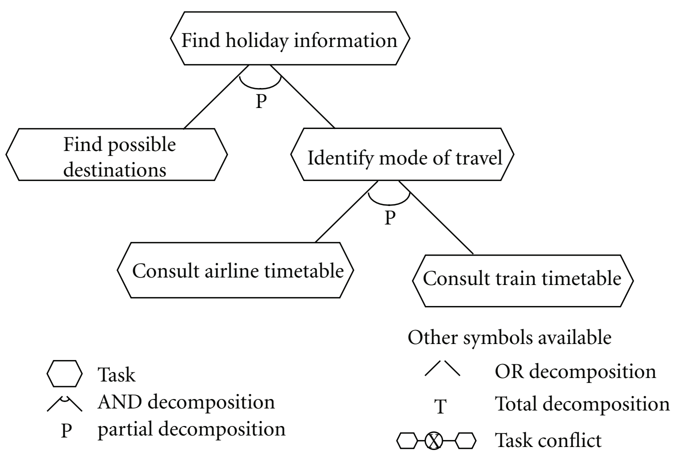

Figure 86: MOBMAS's use of a task decomposition diagram.

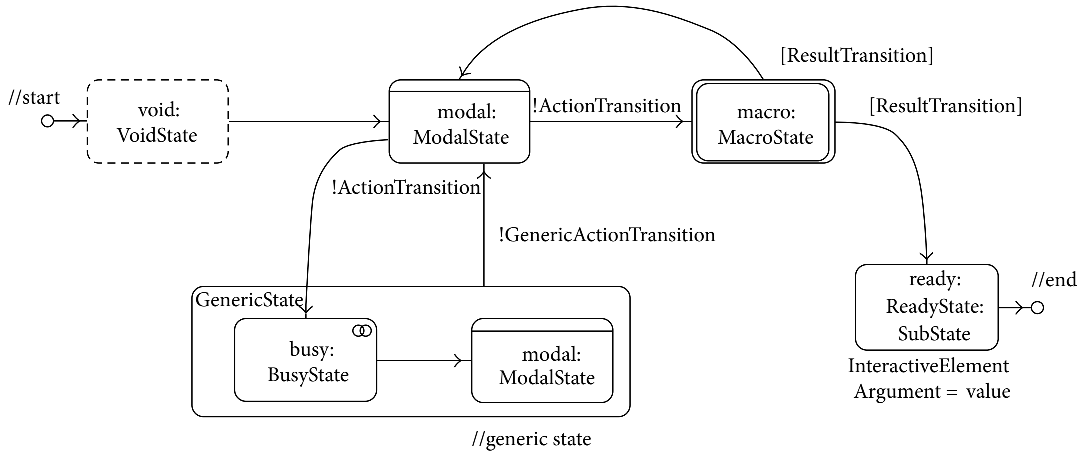

Figure 87: An OPEN/Metis service state diagram in which some of the annotations on arcs and nodes refer back to elements in the user interface sketch of Figure 69 (after [126]).

is a promising approach to gain a well-grounded conceptual foundation for agent-oriented modelling.

It should also be noted that all work products go through their own "lifecycle" in the sense that they are first created, and then modified towards maturity of a final state. This means that there will likely be several versions of each work product (e.g., [145, Appendix G]); that is, the notion of a "state" can be associated with each work product (e.g., [45, 146]).

As well as agent-oriented metamodelling treatises, some authors have sought to set their work in the context of model-driven engineering (MDE) or model-driven development (MDD), for example, Amor et al. [147]; Fischer et al. [148]; Taveter and Sterling [149] and the proposal by Benguria et al. [150] of Platform Independent Model (PIM) for Service-Oriented Architecture (SOA) named PIM4SOA. Others (e.g., Liu et al. [151]) have examined the possible utilization of agents for developing web services and in SOA (Service-Oriented Architecture). Internet development is also discussed by Zambonelli et al. [152].

There is also an emerging trend to adopt a method engineering mindset for agent-oriented software construction (e.g., [153]). In a recent paper on O-MaSE [46], the authors tabulate their recommended diagram types in the form of method fragments (Table 14).

In addition to these related works on notation and metamodelling, we were only able to find a small number of additional papers in the main topic area that are not already cited above. In particularly, although differently named in part, the six diagram types proffered by Juan et al. [154] in their "skeleton methodology" are commensurate with those discussed here. They are given as Use-case model, Environmental Interface Model, Agent Model, Service Model, Acquaintance Model, and Interaction Model. In addition, they proffer (i) from Prometheus: System Overview Diagram, Agent Overview Diagram, Capability Diagram, and Event, Data, Plan, and Capability Descriptors; and (ii) from ROADMAP: Environment Model, Knowledge Model, and Role Model.

Our use of FAML notation has only been indicative rather than the provision of any conclusive results. We have anticipated following the comprehensive evaluation method of the notation for ISO/IEC 24744 International Standard [21] as undertaken by Sousa et al. [8]. However, it turns out that there are significant differences between the mode of utilization of symbols in creating a process model (for which ISO/IEC 24744 was designed) and the way symbols are used in an AOML. In the former, not only are the symbols evaluated but also, perhaps more critically, the superposition of various combinations in terms of their usability vis-a-vis their shape and colour turns out to be the more important aspect. On the contrary, for an AOML, there is little (or zero) need to superpose symbols rather than to have a collection of them related to each other in any specific diagram type. This means that a symbol set for a AOSE diagramming suite need have little concern for juxtapositioning and superpositioning issues but simply be evaluated in terms of its semiotic value in terms of the degree to which each symbol successfully represents each AOSE concept. That means that our illustration of only a few diagram types, chosen to illustrate elements of the different "families" of Figure 2, indicates that further one-to-one translation of symbols between, say, Prometheus and FAML or between INGENIAS and FAML should be as successful. In terms of the goals of this current paper, the profferings of FAML diagram types are adequate, whilst leaving to future work (Section 8) comprehensive user and usability studies.

Thus, creating a standard, for which this paper is intended to be a potential precursor, needs careful mappings between the various methodology-linked diagram types by consideration of their associated semantics (i.e., not just names and notations). Once these similarities and any overlaps have been identified, it is likely that the number of diagram types needed for AOSE can be significantly reduced from the sum

Table 14: Diagram types recommended in O-MaSE depicted as method fragments (after [46]) and reprinted by permission of the publisher © Inderscience.

Activities	Tasks	Work products created or modified	Responsible method roles
Requirements gathering	Requirements specification	Requirements specification	Requirements engineer
Problem analysis	Model goals	Goal model	Goal modeller
	Refine goals		
	Model domain	Domain model	Domain modeller
Solution analysis	Model organization interfaces	Organization model	Organization modeller
	Model roles	Role model	Role modeller
	Define roles	Role description document	
	Define role goals	Role goal model	
Architecture design	Model agent classes	Agent class model	Agent class modeller
	Model protocols	Protocol model	Protocol modeller
	Model policies	Policy model	Policy modeller
Low level design	Model plans	Agent plan model	Plan modeller
	Model capabilities	Capabilities model	Capabilities modeller
	Model actions	Action model	Action modeller
Code generation	Generate code	Source code	Programmer

of all the diagram types across all AOSE methodologies—the aim of this current project.

8. Conclusions and Future Work

Using the list of over two dozen proposed diagram types in the AOSE literature, we have here extended the analysis of Henderson-Sellers [19] commencing with his recommendations and assessing to what extent these recommendations are seen in this wide range of AOSE methodologies. We have also taken the opportunity, as indicated in the Future Work Section of Henderson-Sellers [19], to express several of these recommended diagram types using the new FAML notation [5], itself conformant to the metamodel of Beydoun et al. [4], and derived in part from the suggestions of Padgham et al. [2] and DeLoach et al. [155], and taking into account the semiotic advice of Moody [7].

In summary, we have added further evaluations of the recommendations of Henderson-Sellers [19] by consideration of a wider range of diagram types in the AOSE methodologies listed in Table 7, the motivation being a small additional contribution to standards efforts current in organizations like FIPA, OMG, and ISO.

As noted in Henderson-Sellers et al. [5], further empirical research is required to evaluate the usability of these various diagram types using FAML's notation. We plan to undertake an experiment in which creative design students are asked to supply appropriate symbols for the FAML concepts as well as evaluating our current proposals (Figure 2). We also intend to conduct a comprehensive evaluation using a large group (20 plus) of experts followed by a usability study in a real world case study.

Acknowledgments

The author wishes to thanks Cesar Gonzalez-Perez for helpful comments on an earlier draft of this paper. This is contribution 12/06 of the Centre for Object Technology Applications and Research at the University of Technology, Sydney.

References

[1] A. Susi, A. Perini, J. Mylopoulos, and P. Giorgini, "The Tropos metamodel and its use," *Informatica*, vol. 29, no. 4, pp. 401–408, 2005.

[2] L. Padgham, M. Winikoff, S. Deloach, and M. Cossentino, "A unified graphical notation for AOSE?" in *Agent-Oriented Software Engineering IX: 9th International Workshop (AOSE '08) Estoril, Portugal, May 12-13, 2008 Revised Selected Papers*, M. Luck and J. J. Gomez-Sanz, Eds., vol. 5386 of *Lecture Notes in Computer Science*, pp. 116–130, Springer, Berlin, Germany, 2009.

[3] C. Bernon, M. Cossentino, M.-P. Gleizes, P. Turci, and F. Zambonelli, "A study of some multiagent meta-models," in *Proceedings of the 5th International Workshop on Agent-Oriented Software Engineering V (AOSE '04)*, J. Odell, P. Giorgini, and J. P. Müller, Eds., vol. 3382 of *Lecture Notes in Computer Science*, pp. 62–77, Springer, Berlin, Germany, 2004.

[4] G. Beydoun, G. Low, B. Henderson-Sellers et al., "FAML: a generic metamodel for MAS development," *IEEE Transactions on Software Engineering*, vol. 35, no. 6, pp. 841–863, 2009.

[5] B. Henderson-Sellers, G. C. Low, and C. Gonzalez-Perez, "Semiotic considerations for the design of an agent-oriented modelling language, enterprise, business-process and information systems modeling," in *Proceedings of the 13th International Conference and 17th International Conference (Emmsad '12)*, I. Bider, T. Halpin, J. Krogstie et al., Eds., vol. 113 of *Lecture*

Notes in Business Information Processing, pp. 422–434, Springer, Heidelberg, Germany, 2012.

[6] L. L. Constantine and B. Henderson-Sellers, "Notation matters: part 1—framing the issues," *Report on Object Analysis and Design*, vol. 2, no. 3, pp. 25–29, 1995.

[7] D. Moody, "The physics of notations: toward a scientific basis for constructing visual notations in software engineering," *IEEE Transactions on Software Engineering*, vol. 35, no. 6, pp. 756–779, 2009.

[8] K. Sousa, J. Vanderdonckt, B. Henderson-Sellers, and C. Gonzalez-Perez, "Evaluating a graphical notation for modelling software development methodologies," *Journal of Visual Languages and Computing*, vol. 23, no. 4, pp. 195–212, 2012.

[9] OMG, "Unified Modelling Language Specification," formal/01-09-68 through 80 (13 documents). Object Management Group, 2001.

[10] OMG, Unified Modeling Language: superstructure, Version 2. 0, formal/05-07-04, 709pp, 2005.

[11] OMG, OMG Unified Modeling Language (OMG UML), Superstructure, V2. 1. 2, formal/2007-11-02, 738pp, 2007.

[12] A. F. Garcia, C. J. P. De Lucena, and D. D. Cowan, "Agents in object-oriented software engineering," *Software*, vol. 34, no. 5, pp. 489–521, 2004.

[13] A. S. Rao and M. P. Georgeff, "BDI agents: from theory to practice," Technical Note 56, Australian Artificial Intelligence Institute, 1995.

[14] D. Kinny, M. Georgeff, and A. Rao, "A methodology and modelling technique for systems of BDI agents," in *Proceedings of the 17th European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW '96)*, pp. 56–71, Eindhoven, The Netherlands, 1996.

[15] A. Sturm and O. Shehory, "A framework for evaluating agent-oriented methodologies," in *Proceedings of the 5th International Bi-Conference on Agent-Oriented Information Systems*, P. Giorgini and M. Winikoff, Eds., pp. 60–67, 2003.

[16] A. Sturm and O. Shehory, "A comparative evaluation of agent-oriented methodologies," in *Methodologies and Software Engineering for Agent Systems. The Agent-Oriented Software Engineering Handbook*, F. Bergenti, M. P. Gleizes, and F. Zambonelli, Eds., chapter 7, pp. 127–149, Kluwer Academic Publishers, Boston, Mass, USA, 2004.

[17] K. Taveter and G. Wagner, "Towards radical agent-oriented software engineering processes based on AOR modelling," in *Agent-Oriented Methodologies*, B. Henderson-Sellers and P. Giorgini, Eds., chapter 10, pp. 277–316, Idea Group, 2005.

[18] D. Bertolini, A. Novikau, A. Susi, and A. Perini, "TAOM4E: an Eclipse ready tool for agent-oriented modeling," Issue on the Development Process, IRST Report, Trento, Italy, 2006.

[19] B. Henderson-Sellers, "Consolidating diagram types from several agent-oriented methodologies," *Frontiers in Artificial Intelligence and Applications*, vol. 217, pp. 293–345, 2010.

[20] J. Pavón, J. J. Gómez-Sanz, and R. Fuentes, "The INGENIAS methodology and tools," in *Agent-Oriented Methodologies*, B. Henderson-Sellers and P. Giorgini, Eds., chapter 9, pp. 236–276, Idea Group, 2005.

[21] ISO/IEC, "Software Engineering. Metamodel for Development Methodologies," ISO/IEC International Standard 24744, Annex A—Notation, ISO, Geneva, Switzerland, 2010.

[22] G. Picard and M. P. Gleizes, "The ADELFE methodology," in *Methodologies and Software Engineering for Agent Systems. the Agent-Oriented Software Engineering Handbook*, F. Bergenti, M. P. Gleizes, and F. Zambonelli, Eds., chapter 8, pp. 157–175, Kluwer Academic Publishers, Boston, Mass, USA, 2004.

[23] R. Collier, G. O'Hare, T. Lowen, and C. Rooney, "Beyond prototyping in the factory of agents," in *Multi-Agent Systems and Applications III*, V. Marik, J. Muller, and M. Pechoucek, Eds., vol. 2691 of *Lecture Notes in Computer Science*, pp. 383–393, Springer, New York, NY, USA, 2003.

[24] R. Collier, G. O'Hare, and C. Rooney, "A UML-based software engineering methodology for agent factory," in *Proceedings of the 16th International Conference on Software Engineering & Knowledge Engineering (SEKE '04)*, F. Maurer and G. Ruhe, Eds., pp. 25–30, Banff, Canada, June 2004.

[25] L. Shan and H. Zhu, "CAMLE: a caste-centeric agent-oriented modeling language and environment," in *Software Engineering for Multi-Agent Systems III*, R. Choren, A. Garcia, C. Lucena, and A. Romanovsky, Eds., vol. 3390 of *Lecture Notes in Computer Science*, pp. 144–161, Springer, Berlin, Germany, 2005.

[26] A. Collinot, A. Drogoul, and P. Benhamou, "Agent oriented design of a soccer robot team," in *Proceedings of the 2nd International Conference on Multi-Agent Systems (ICMAS '96)*, M. Tokoro, Ed., pp. 41–47, AAAI Press, Menlo Park, Calif, USA, 1996.

[27] A. Collinot and A. Drogoul, "Using the Cassiopeia method to design a robot soccer team," *Applied Artificial Intelligence*, vol. 12, no. 2-3, pp. 127–147, 1998.

[28] M. Elammari and W. Lalonde, "An agent-oriented methodology: high-level and intermediate models," in *Proceedings of the 1st International Workshop on Agent-Oriented Information Systems (AOIS '99)*, Heidelberg, Germany, June 1999.

[29] 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.

[30] F. Zambonelli, N. R. Jennings, and M. Wooldridge, "Developing multiagent systems: the Gaia methodology," *ACM Transactions on Software Engineering and Methodology*, vol. 12, no. 3, pp. 317–370, 2003.

[31] F. Zambonelli, N. Jennings, and M. Wooldridge, "Multi-agent systems as computational organizations: the Gaia methodology," in *Agent-Oriented Methodologies*, B. Henderson-Sellers and P. Giorgini, Eds., chapter 6, pp. 136–171, Idea Group, 2005.

[32] T. Juan, A. Pearce, and L. Sterling, "ROADMAP: extending the Gaia methodology for complex open systems," in *Proceedings of the 1st International Joint Conference on Autonomous Agents adn Multiagent Systems*, pp. 3–10, Bologna, Italy, July 2002.

[33] L. Sterling, K. Taveter, and The Daedalus Team, "Building agent-based appliances with complementary methodologies," in *Knowledge-Based Software Engineering*, E. Tyugu and T. Yamaguchi, Eds., pp. 223–232, IOS Press, 2006.

[34] J. Pavón and J. J. Gómez-Sanz, "Agent-oriented software engineering with INGENIAS," in *Proceedings of the 3rd International/Central and Eastern European Conference on Multi-Agent Systems (CEEMAS '03)*, V. Marik, J. Muller, and M. Pechoucek, Eds., vol. 2691 of *Lecture Notes in Artificial Intelligence*, pp. 394–403, Springer, Berlin, Germany, 2003.

[35] C. Sierra, J. Thangarajah, L. Padgham, and M. Winikoff, "Designing institutional multi-agent systems," in *Agent-Oriented Software Engineering VII*, L. Padgham and F. Zambonelli, Eds., vol. 4405 of *Lecture Notes in Computer Science*, pp. 84–103, Springer, Berlin, Germany, 2007.

[36] C. Sierra, J. A. Rodríguez-Aguilar, and J. L. Arcos, "Helios: harmonious electronic institution operational scheme," Tech. Rep. IIIA-TR-2009-10, IIIA, CSIC, Barcelona, Spain, 2009.

[37] C. A. Iglesias and M. Garijo, "The agent-oriented methodology MAS-CommonKADS," in *Agent-Oriented Methodologies*, B. Henderson-Sellers and P. Giorgini, Eds., chapter 3, pp. 46–78, Idea Group, Hershey, Pa, USA, 2005.

[38] C. A. Iglesias, M. Garijo, J. C. Gonzalez, and J. R. Velasco, "A methodological proposal for multiagent systems development extending CommonKADS," in *Proceedings of the 10th Knowledge Acquisition Workshop (KAW '96)*, SRDG Publications, Department of Computer Science, University of Calgary, Banff, Canada, 1996.

[39] C. A. Iglesias, M. Garijo, J. C. Gonzalez, and J. R. Velasco, "Analysis and design of multi-agent systems using MAS-CommonKADS," in *Intelligent Agents IV*, M. P. Singh, A. Rao, and M. J. Wooldridge, Eds., vol. 1365 of *Lecture Notes in Artificial Intelligence*, pp. 313–328, Springer, Berlin, Germany, 1998.

[40] M. F. Wood and S. A. DeLoach, "An overview of the multiagent systems engineering methodology," in *Proceedings of the 1st International Workshop on Agent-Oriented Software Engineering (AOSE '00)*, P. Ciancarini and M. J. Wooldridge, Eds., vol. 1957 of *Lecture Notes in Computer Science*, pp. 207–221, Springer, Berlin, Germany, 2000.

[41] S. A. DeLoach, "Multiagent systems engineering: a methodology and language for designing agent systems," in *Proceedings of the Agent-Oriented Information Systems (AOIS '99)*, Seattle, Wash, USA, May 1999.

[42] S. A. DeLoach, "Modeling organizational rules in the multi-agent systems engineering methodology," in *Proceedings of the Advances in Artificial Intelligence (AI '02)*, R. Cohen and B. Spencer, Eds., vol. 2338 of *Lecture Notes in Artificial Intelligence*, pp. 1–15, Springer, Berlin, Germany, 2002.

[43] S. A. DeLoach, "The MaSE methodology," in *Methodologies and Software Engineering for Agent Systems the Agent-Oriented Software Engineering Handbook*, F. Bergenti, M. P. Gleizes, and F. Zambonelli, Eds., chapter 6, pp. 107–125, Kluwer Academic Publishers, Boston, Mass, USA, 2004.

[44] S. A. DeLoach and M. Kumar, "Multiagent systems engineering: an overview and case study," in *Agent-Oriented Methodologies*, B. Henderson-Sellers and P. Giorgini, Eds., chapter 11, pp. 317–340, Idea Group, Hershey, Pa, USA, 2005.

[45] J. C. Garcia-Ojeda, S. A. DeLoach, Robby, W. H. Oyenan, and J. Valenzuela, "O-MaSE: a customizable approach to developing multiagent development processes," in *Proceedings of the 8th International Workshop on Agent Oriented Software Engineering VIII (AOSE '07)*, M. Luck, Ed., vol. 4951 of *Lecture Notes in Computer Science*, pp. 1–15, Springer, Berlin, Germany, 2007.

[46] S. A. DeLoach and J. C. Garcia-Ojeda, "O-MaSE: a customisable approach to designing and building complex, adaptive multi-agent systems," *International Journal of Agent-Oriented Software Engineering*, vol. 4, no. 3, pp. 244–280, 2010.

[47] G. Caire, W. Coulier, F. Garijo et al., "Agent oriented analysis using MESSAGE/UML," in *Proceedings of the 2nd International Workshop on Agent-Oriented Software Engineering II (AOSE '01)*, M. J. Wooldridge, G. Weiß, and P. Ciancarini, Eds., vol. 2222 of *Lecture Notes in Computer Science*, pp. 119–135, Springer, Berlin, Germany, 2001.

[48] G. Caire, W. Coulier, F. Garijo et al., "The MESSAGE methodology," in *Methodologies and Software Engineering for Agent Systems. The Agent-Oriented Software Engineering Handbook*, F. Bergenti, M.-P. Gleizes, and F. Zambonelli, Eds., chapter 9, pp. 177–194, Kluwer Academic Publishers, 2004.

[49] F. J. Garijo, J. J. Gómez-Sanz, and Ph. Massonet, "The MESSAGE methodology for agent-oriented analysis and design," in *Agent-Oriented Methodologies*, B. Henderson-Sellers and P. Giorgini, Eds., chapter 8, pp. 203–235, Idea Group, 2005.

[50] Q. N. N. Tran, G. Low, and G. Beydoun, "A methodological framework for ontology centric oriented software engineering," *Computer Systems Science and Engineering*, vol. 21, no. 2, pp. 117–132, 2006.

[51] Q. N. N. Tran and G. Low, "MOBMAS: a methodology for ontology-based multi-agent systems development," *Information and Software Technology*, vol. 50, no. 7-8, pp. 697–722, 2008.

[52] V. Dignum, *A model for organizational interaction based on agents, founded in logic [Ph.D. thesis]*, SIKS, Amsterdam, The Netherlands, 2004, SIKS Dissertation Series No. 2004-1.

[53] M. Mensonides, B. Huisman, and V. Dignum, "Towards agent-based scenario development for strategic decision support," in *Proceedings of the 8th International Bi-Conference Workshop on Agent-Oriented Information Systems IV (AOIS '06)*, P. Bresciani, A. Garcia, A. Ghose, B. Henderson-Sellers, M. Kolp, and H. Mouratidis, Eds., vol. 4898 of *Lecture Notes in Artificial Intelligence*, pp. 53–72, Springer, Berlin, Germany, 2006.

[54] P. Burrafato and M. Cossentino, "Designing a multi-agent solution for a bookstore with the PASSI methodology," in *Proceedings of the 4th International Bi-Conference Workshop on Agent-Oriented Information Systems (AOIS '02)*, P. Giorgini, Y. Lespérance, G. Wagner, and E. S. K. Yu, Eds., Toronto, Canada, May 2002, CEUR Workshop Proceedings 57, CEUR-WS.org.

[55] M. Cossentino, "Different perspectives in designing multi-agent systems," in *Proceedings of the Agent Technology and Software Engineering Workshop at (NODe '02)*, Erfurt, Germany, October 2002.

[56] M. Cossentino, "From requirements to code with the PASSI methodology," in *Agent-Oriented Methodologies*, B. Henderson-Sellers and P. Giorgini, Eds., pp. 79–106, Idea Group, 2005.

[57] M. Cossentino and C. Potts, "PASSI: a process for specifying and implementing multi-agent systems using UML," 2002, http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.4.3182.

[58] M. Cossentino and C. Potts, "A CASE tool supported methodology for the design of multi-agent systems," in *Proceedings of the International Conference on Software Engineering Research and Practice (SERP '02)*, Las Vegas, Nev, USA, June 2002.

[59] L. Padgham and M. Winikoff, *Developing Intelligent Agent Systems: A Practical Guide*, J. Wiley and Sons, Chichester, UK, 2004.

[60] L. Padgham and M. Winikoff, "Prometheus: a practical agent-oriented methodology," in *Agent-Oriented Methodologies*, B. Henderson-Sellers and P. Giorgini, Eds., chapter 5, pp. 107–135, Idea Group, 2005.

[61] M. Winikoff and L. Padgham, "The Prometheus methodology," in *Methodologies and Software Engineering for Agent Systems. The Agent-Oriented Software Engineering Handbook*, F. Bergenti, M. P. Gleizes, and F. Zambonelli, Eds., chapter 11, pp. 217–234, Kluwer Academic Publishers, Boston, Mass, USA, 2004.

[62] J. Khallouf and M. Winikoff, "The goal-oriented design of agent systems: a refinement of Prometheus and its evaluation," *International Journal of Agent-Oriented Software Engineering*, vol. 3, no. 1, pp. 88–112, 2009.

[63] A. Omicini, "SODA: societies and infrastructures in the analysis and design of agent-based systems," in *Proceedings of the Agent-Oriented Software Engineering (AOSE '00)*, P. Ciancarini and

M. J. Wooldridge, Eds., vol. 1957 of *Lecture Notes in Computer Science*, pp. 185–193, Springer, Berlin, Germany, 2000.

[64] F. Alonso, S. Frutos, L. Martínez, and F. J. Soriano, "The synthesis stage in the software agent development process," in *Proceedings of the 4th International Central and Eastern European Conference on Multi-Agent Systems and Applications (CEEMAS '05)*, M. Pěchouček, P. Petta, and L. Z. Varga, Eds., vol. 3690 of *Lecture Notes in Artificial Intelligence*, pp. 193–202, Springer, Berlin, Germany, 2005.

[65] P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, and J. Mylopoulos, "Tropos: an agent-oriented software development methodology," *Autonomous Agents and Multi-Agent Systems*, vol. 8, no. 3, pp. 203–236, 2004.

[66] P. Bresciani, P. Giorgini, H. Mouratidis, and G. Manson, "Multi-agent systems and security requirements analysis," in *Advances in Software Engineering for Multi-Agent Systems*, C. Lucena, A. Garcia, A. Romanovsky, J. Castro, and P. Alencar, Eds., vol. 2940 of *Lecture Notes in Computer Science*, pp. 35–48, Springer, Berlin, Germany, 2004.

[67] P. Giorgini, M. Kolp, J. Mylopoulos, and J. Castro, "Tropos: a requirements-driven methodology for agent-oriented software," in *Agent-Oriented Methodologies*, B. Henderson-Sellers and P. Giorgini, Eds., chapter 2, pp. 20–45, Idea Group, Hershey, Pa, USA, 2005.

[68] F. Peyravi and F. Taghyareh, "Applying mas-commonkads methodology in knowledge management problem in call centers," in *Proceedings of the IASTED International Conference on Software Engineering (SE '07)*, pp. 99–104, February 2007.

[69] G. Booch, J. Rumbaugh, and I. Jacobson, *The Unified Modeling Language User Guide*, Addison-Wesley, Reading, Mass, USA, 1999.

[70] L. Shan and H. Zhu, "Unifying the semantics of models and meta-models in the multi-layered UML meta-modelling hierarchy," *International Journal of Software and Informatics*, vol. 6, no. 2, pp. 163–200, 2012.

[71] V. Torres da Silva, R. Choren, and C. J. P. de Lucena, "MAS-ML: a multiagent system modelling language," *International Journal of Agent-Oriented Software Engineering*, vol. 2, no. 4, pp. 382–421, 2008.

[72] I. Trencansky and R. Cervenka, "Agent Modeling Language (AML): a comprehensive approach to modeling MAS," *Informatica*, vol. 29, no. 4, pp. 391–400, 2005.

[73] R. Cervenka and I. Trencansky, *AML. The Agent Modeling Language*, Birkhäuser, Basel, Switzerland, 2007.

[74] B. Bauer, J. P. Müller, and J. Odell, "Agent UML: a formalism for specifying multiagent software systems," *International Journal of Software Engineering and Knowledge Engineering*, vol. 11, no. 3, pp. 207–230, 2001.

[75] M. P. Huget and J. Odell, "Representing agent interaction protocols with agent UML," in *Proceedings of the 5th International Workshop on Agent-Oriented Software Engineering V (AOSE '04)*, J. Odell, P. Giorgini, and J. P. Müller, Eds., vol. 3382 of *Lecture Notes in Computer Science*, pp. 16–30, Springer, Berlin, Germany, 2004.

[76] V. Torres da silva and C. J. P. de Lucena, "Form a conceptual framework for agents and objects to a multi-agent system modeling language," *Autonomous Agents and Multi-Agent Systems*, vol. 9, no. 1-2, pp. 145–189, 2004.

[77] R. Choren and C. Lucena, "The ANote modelling language for agent-oriented specification," in *Proceedings of the Software Engineering for Multi-Agent Systems III (SELMAS '04)*, R. Choren, A. Garcia, C. Lucena, and A. Romanovsky, Eds., vol. 3390 of *Lecture Notes in Computer Science*, pp. 198–212, Springer, Berlin, Germany, 2004.

[78] E. S. K. Yu, *Modelling strategic relationships for process reengineering [Ph.D. thesis]*, University of Toronto, 1995.

[79] J. Mylopoulos, M. Kolp, and J. Castro, "UML for agent-oriented software development: the Tropos proposal," in *Proceedings of the 4th International Conference on the Unified Modeling Language (UML '01)*, M. Gogolla and C. Kobryn, Eds., vol. 2185 of *Lecture Notes in Computer Science*, pp. 422–441, Springer, Berlin, Germany, 2001.

[80] A. Lapouchnian and Y. Lespérance, "Modeling mental states in agent-oriented requirements engineering," in *Proceedings of the 18th International Conference on Advanced Information Systems Engineering (CAiSE '06)*, E. Dubois and K. Pohl, Eds., vol. 4001 of *Lecture Notes in Computer Science*, pp. 480–494, Springer, Berlin, Germany, 2006.

[81] S. Shapiro and Y. Lespérance, "Modeling multiagent systems with the Cognitive Agents Specification Language—a feature interaction resolution application," in *Proceedings of the 7th International Workshop (ATAL '00)*, vol. 1986 of *Lecture Notes in Artificial Intelligence*, pp. 244–259, Springer, Berlin, 2000.

[82] X. Franch, "On the quantitative analysis of agent-oriented models," in *Proceedings of the Advanced Information Systems Engineering (CAiSE '06)*, E. Dubois and K. Pohl, Eds., vol. 4001 of *Lecture Notes in Computer Science*, pp. 495–509, Springer, Berlin, Germany, 2006.

[83] H. Estrada, A. Martínez Rebollar, O. Pastor, and J. Mylopoulos, "An empirical evaluation of the i* framework in a model-based software generation environment," in *Proceedings of the Advanced Information Systems Engineering (CAiSE '06)*, E. Dubois and K. Pohl, Eds., vol. 4001 of *Lecture Notes in Computer Science*, pp. 513–527, Springer, Berlin, Germany, 2006.

[84] L. L. Constantine and B. Henderson-Sellers, "Notation matters: part 2—applying the principles," *Report on Object Analysis and Design*, vol. 2, no. 4, pp. 20–23, 1995.

[85] H. Mouratidis, *A security oriented approach in the development of multiagent systems: applied to the management of health and social care needs of older people in England [Ph.D. thesis]*, Department of Computer Science, University of Sheffield, 2004.

[86] H. Hachicha, A. Loukil, and K. Ghédira, "MA-UML: a conceptual approach for mobile agents' modelling," *International Journal of Agent-Oriented Software Engineering*, vol. 3, no. 2-3, pp. 277–305, 2009.

[87] G. Low, H. Mouratidis, and B. Henderson-Sellers, "Using a situational method engineering approach to identify reusable method fragments from the secure TROPOS methodology," *Journal of Object Technology*, vol. 9, no. 4, pp. 93–125, 2010.

[88] J. P. Georgé, B. Edmonds, and P. Glize, "Making self-organising adaptive multiagent systems work," in *Methodologies and Software Engineering for Agent Systems. The Agent-Oriented Software Engineering Handbook*, F. Bergenti, M. P. Gleizes, and F. Zambonelli, Eds., chapter 16, pp. 321–340, Kluwer Academic Publishers, Boston, Mass, USA, 2004.

[89] E. Steegmans, D. Weyns, T. Holvoet, and Y. Berbers, "A design process for adaptive behaviour of situated agents," in *Proceedings of the 5th International Conference on Agent-Oriented Software Engineering (AOSE '04)*, J. Odell, P. Giorgini, and J. P. Müller, Eds., vol. 3382 of *Lecture Notes in Computer Science*, pp. 109–125, Springer, Berlin, Germany, 2004.

[90] Y. Demazeau, *La Méthode VOYELLES, Dans Systèmes Multi-Agents. Des Théories Organisationnelles Aux Applications Industrielles*, Hermès, Oslo, Norway, 2001.

[91] B. Henderson-Sellers, Q.-N. Tran, and J. N. Debenham, "Incorporating elements from the Prometheus agent-oriented methodology in the OPEN Process Framework," in *Proceedings of the Agent-Oriented Information Systems II*, P. Bresciani, P. Giorgini, B. Henderson-Sellers, G. Low, and M. Winikoff, Eds., vol. 3508 of *Lecture Notes in Artificial Intelligence*, pp. 140–156, Springer, Berlin, Germany, 2005.

[92] M. P. Huget, "Agent UML class diagrams revisited," in *Proceedings of the Agent Technology Workshops*, R. Kowalczyk, J. Müller, H. Tianfield, and R. Unland, Eds., vol. 2592 of *Lecture Notes in Artificial Intelligence*, pp. 49–60, Springer, Berlin, Germany, 2003.

[93] ISO/IEC, "Unified Modeling Language (UML) Version 1. 4. 2," ISO/IEC 19501, International Organization for Standardization/International Electrotechnical Commission, Geneva, Switzerland, 2005.

[94] Q. N. N. Tran and G. Low, "Comparison of ten agent-oriented methodologies," in *Agent-Oriented Methodologies*, B. Henderson-Sellers and P. Giorgini, Eds., chapter 12, pp. 341–367, Idea Group, 2005.

[95] H. K. Dam and M. Winikoff, "Towards a next-generation AOSE methodology," *Science of Computer Programming*. In press.

[96] J. P. Müller, *The Design of Intelligent Agents*, Springer, Berlin, Germany, 1996.

[97] H. V. D. Parunak and J. J. Odell, "Representing social structures in UML," in *Proceedings of the Agent-Oriented Software Engineering (AOSE '01)*, M. Wooldridge, P. Ciancarini, and G. Weiss, Eds., vol. 2222 of *Lecture Notes in Computer Science*, pp. 1–16, Springer, Berlin, Germany, 2001.

[98] V. Dignum and F. Dignum, "Designing agent systems: state of the practice," *International Journal of Agent-Oriented Software Engineering*, vol. 4, no. 3, pp. 224–243, 2010.

[99] M. Viroli, T. Holvoet, A. Ricci, K. Schelfthout, and F. Zambonelli, "Infrastructures for the environment of multiagent systems," *Autonomous Agents and Multi-Agent Systems*, vol. 14, no. 1, pp. 49–60, 2007.

[100] OMG, Agent Metamodel and Profile (AMP). Request For Proposal, OMG Document: ad/2008-08-10, 2008.

[101] B. Henderson-Sellers, Q. N. N. Tran, and J. Debenham, "An etymological and metamodel-based evaluation of the terms "goals and tasks" in agent-oriented methodologies," *Journal of Object Technology*, vol. 4, no. 2, pp. 131–150, 2005.

[102] J. Ferber and O. Gutknecht, "A meta-model for the analysis and design of organizations in multi-agent systems," in *Proceedings of the 3rd International Conference on Multi Agent Systems (ICMAS '98)*, pp. 128–135, IEEE Computer Society, Los Alamitos, CA, USA, 1998.

[103] V. Dignum and H. Weigand, "Toward an organization-oriented design methodology for agent societies," in *Intelligent Agent Software Engineering*, V. Plekhanova, Ed., chapter 9, pp. 191–212, Idea Group Publishing, 2003.

[104] J. Gonzalez-Palacios and M. Luck, "A framework for patterns in Gaia: a case-study with organisations," in *Proceedings of the 5th International Workshop on Agent-Oriented Software Engineering (AOSE '04)*, J. Odell, P. Giorgini, and J. P. Müller, Eds., vol. 3382 of *Lecture Notes in Computer Science*, pp. 174–188, Springer, Berlin, Germany, 2004.

[105] R. Cervenka, I. Trencansky, M. Calisti, and D. Greenwood, "AML: agent modeling language toward industry-grade agent-based modelling," in *Proceedings of the 5th International Conference on Agent-Oriented Software Engineering (AOSE '04)*, J. Odell, P. Giorgini, and J. P. Müller, Eds., vol. 3382 of *Lecture Notes in Computer Science*, pp. 31–46, Springer, Berlin, Germany, 2004.

[106] C. A. Iglesias and M. Garijo, "UER technique: conceptualization for agent-oriented development," in *Proceedings of the 3rd World Multi Conference on Systemics, Cybernetics and Informatics (SCI '99) and 5th International Conference on Information Systems Analysis and Synthesis (ISAS '99)*, vol. 5, pp. 535–540, International Institute of Informatics and Systemics, 1999.

[107] A. Cockburn, *Writing Effective Use Cases*, Addison-Wesley, Boston, Mass, USA, 2001.

[108] S. Azaiez, M. P. Huget, and F. Oquendo, "An approach for multiagent metamodelling," *Multiagent and Grid Systems*, vol. 2, no. 4, pp. 435–454, 2007.

[109] E. A. Kendall, "Role modeling for agent system analysis, design, and implementation," *IEEE Concurrency*, vol. 8, no. 2, pp. 34–41, 2000.

[110] E. A. Kendall, "Agent software engineering with role modelling," in *Proceedings of the 1st International Workshop on Agent-Oriented Software Engineering*, vol. 1957 of *Lecture Notes in Computer Science*, pp. 163–170, Springer, Berlin, Germany, 2001.

[111] J. J. Odell, H. V. D. Parunak, and M. Fleischer, "The role of roles in designing effective agent organizations," in *Proceedings of the Software Engineering for Large-Scale Multi-Agent Systems (SELMAS '02)*, . Garcia, C. Lucena, F. Zambonelli, A. Omicini, and J. Castro, Eds., vol. 2603 of *Lecture Notes in Computer Science*, pp. 27–38, Springer, Berlin, Germany, 2002.

[112] J. J. Odell, H. V. D. Parunak, and M. Fleischer, "Modeling agent organizations using roles," *Software and Systems Modeling*, vol. 2, no. 2, pp. 76–81, 2003.

[113] C. B. Ward and B. Henderson-Sellers, "Utilizing dynamic roles for agents," *Journal of Object Technology*, vol. 8, no. 5, pp. 177–198, 2009.

[114] L. Sterling, "Agent-oriented modelling: declarative or procedural?" in *Proceedings of the Declarative Agent Languages and Technologies V (DALT '07)*, M. Baldoni, T. C. Son, M. B. van Riemsdijk, and M. Winikoff, Eds., vol. 4897 of *Lecture Notes in Artificial Intelligence*, pp. 1–17, Springer, Berlin, Germany, 2007.

[115] L. Cernuzzi, T. Juan, L. Sterling, and F. Zambonelli, "The Gaia methodology: basic concepts and extensions," in *Methodologies and Software Engineering for Agent Systems. The Agent-Oriented Software Engineering Handbook*, F. Bergenti, M.-P. Gleizes, and F. Zambonelli, Eds., chapter 4, pp. 69–88, Kluwer Academic Publishers, Boston, Mass, USA, 2004.

[116] J. Debenham and B. Henderson-Sellers, "Designing agent-based process systems—extending the OPEN Process Framework," in *Intelligent Agent Software Engineering*, V. Plekhanova, Ed., chapter 8, pp. 160–190, Idea Group Publishing, 2003.

[117] S. Shapiro, S. Sardina, J. Thangarajah, L. Cavedon, and L. Padgham, "Revising conflicting intention sets in BDI agents," in *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS '12)*, V. Conitzer, M. Winikoff, W. van der Hoek, and L. Padgham, Eds., pp. 1081–1088, International Foundation for Autonomous Agents and Multiagent Systems, Valencia, Spain, June 2012.

[118] A. Suganthy and T. Chithralekha, "Domain-specific architecture for software agents," *Journal of Object Technology*, vol. 7, no. 6, pp. 77–100, 2008.

[119] V. Silva, A. Garcia, A. Brandão, C. Chavez, C. Lucena, and P. Alencar, "Taming agents and objects in software engineering," in *Proceedings of the Software Engineering for Large-Scale Multi-Agent Systems: Research Issues and Practical Applications (SELMAS '02)*, A. Garcia, C. Lucena, F. Zambonelli, A. Omicini, and J. Castro, Eds., vol. 2603 of *Lecture Notes in Computer Science*, pp. 1–26, Springer, Berlin, Germany, 2002.

[120] L. Braubach, A. Pokahr, D. Moldt, and W. Lamersdorf, "Goal representation for BDI agent systems," in *Proceedings of the 2nd International Conference on Programming Multi-Agent Systems (ProMAS '04)*, R. H. Bordini, M. Dastani, J. Dix, and A. E. F. Seghrouchni, Eds., vol. 3346 of *Lecture Notes in Artificial Intelligence*, pp. 44–65, Springer, Berlin, Germany, 2004.

[121] M. B. Van Riemsdijk, M. Dastani, and J. J. C. Meyer, "Goals in conflict: semantic foundations of goals in agent programming," *Autonomous Agents and Multi-Agent Systems*, vol. 18, no. 3, pp. 471–500, 2009.

[122] I. Graham, *Migrating to Object Technology*, Addison-Wesley, Wokingham, UK, 1995.

[123] C. Cheong and M. Winikoff, "Hermes: designing goal-oriented agent interactions," in *Proceedings of the 6th International Workshop on Agent-Oriented Software Engineering (AOSE '05)*, J. P. Müller and F. Zambonelli, Eds., vol. 3950 of *Lecture Notes in Computer Science*, pp. 16–27, Springer, Berlin, Germany, 2005.

[124] C. Cheong and M. Winikoff, "Improving flexibility and robustness in agent interactions: extending Prometheus with Hermes," in *Proceedings of the Software Engioneering for Large-Scale Multi-agent Systems (SELMAS '05)*, A. Garcia, R. Choren, C. Lucena, P. Giorgini, T. Holvoet, and A. Romanovsky, Eds., vol. 3914 of *Lecture Notes in Computer Science*, pp. 189–206, Springer, Berlin, Germany, 2005.

[125] J. Thangarajah, S. Sardina, and L. Padgham, "Measuring plan coverage and overlap for agent reasoning," in *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS '12)*, V. Conitzer, M. Winikoff, W. van der Hoek, and L. Padgham, Eds., pp. 1049–1056, International Foundation for Autonomous Agents and Multiagent Systems, Valencia, Spain, June 2012.

[126] C. Gonzalez-Perez, "OPEN/Metis. The Integral Object-Oriented Software Development Framework," 2004, http://www.openmetis.com

[127] C. Ware, *Visual Thinking for Design*, Elsevier, Amsterdam, The Netherlands, 2008.

[128] L. L. Constantine and L. A. D. Lockwood, *Software for Use*, Addison-Wesley, Reading, Mass, USA, 1999.

[129] C. Gonzalez-Perez, "Filling the voids: from requirements to deployment with OPEN/Metis," in *Proceedings of the 5th International Conference on Software and Data Technologies (ICSOFT '10)*, Athens, Greece, July 2010.

[130] T. Jorquera, C. Maurel, F. Migeon, M. P. Gleizes, C. Bernon, and N. Bonjean, "ADELFE fragmentation," Rapport IRIT/RR-2009-26-FR, Université Paul Sabatier, Toulouse, France, 2009.

[131] M. Winikoff, L. Padgham, and J. Harland, "Simplifying the development of intelligent agents," in *Proceedings of the 14th Australian Joint Conference on Artificial Intelligence (AI '01)*, Adelaide, December 2001.

[132] L. Padgham and M. Winikoff, "Prometheus: a pragmatic methodology for engineering intelligent agents," in *Proceedings of the Workshop on Agent-Oriented Methodologies (OOPSLA '02)*, J. Debenham, B. Henderson-Sellers, N. Jennings, and J. J. Odell, Eds., Centre for Object Technology Applications and Research, Sydney, Australia, 2002.

[133] V. Torres da Silva, R. Choren, and C. J. P. de Lucena, "Modeling MAS properties with MAS-ML dynamic diagrams," in *Proceedings of the 8th International Bi-Conference Workshopon Agent-Oriented Information Systems IV (AOIS '06)*, M. Kolp, B. Henderson-Sellers, H. Mouratidis, A. Garcia, A. Ghose, and P. Bresciani, Eds., vol. 4898 of *Lecture Notes in Artificial Intelligence*, pp. 1–8, Springer, Berlin, Germany, 2006.

[134] R. Fuentes-Fernández, J. J. Gomez-Sanz, and J. Pavón, "Model integration in agent-oriented development," *International Journal of Agent-Oriented Software Engineering*, vol. 1, no. 1, pp. 2–27, 2007.

[135] L. S. Vygotsky, *Mind and Society*, Harvard University Press, Cambridge, Mass, USA, 1978.

[136] M. P. Gervais, "ODAC: an agent-oriented methodology based on ODP," *Autonomous Agents and Multi-Agent Systems*, vol. 7, no. 3, pp. 199–228, 2003.

[137] J. Odell, H. V. D. Parunak, and B. Bauer, "Extending UML for agents," in *Proceedings of the Agent-Oriented Information Systems Workshop at the 17th National Conference on Artificial Intelligence*, G. Wagner, Y. Lesperance, and E. Yu, Eds., pp. 3–17, Austin, Tex, USA, 2000.

[138] B. Bauer, "UML class diagrams revisited in the context of agent-based systems," in *Proceedings of the Agent-Oriented Software Engineering (AOSE '01)*, M. Wooldridge, P. Ciancarini, and G. Weiss, Eds., vol. 2222 of *Lecture Notes in Computer Science*, pp. 101–118, Springer, Berlin, Germany, 2001.

[139] G. Beydoun, C. Gonzalez-Perez, G. Low, and B. Henderson-Sellers, "Synthesis of a generic MAS metamodel," in *Proceedings of the 4th International Workshop on Software Engineering for Large-Scale Multi-Agent Systems (SELMAS '05)*, A. Garcia, R. Choren, C. Lucena, A. Romanovsky, T. Holvoet, and P. Giorgini, Eds., pp. 27–31, IEEE Computer Society Press, Los Alamitos, CA, USA, 2005.

[140] B. Henderson-Sellers and C. Gonzalez-Perez, "A comparison of four process metamodels and the creation of a new generic standard," *Information and Software Technology*, vol. 47, no. 1, pp. 49–65, 2005.

[141] ISO/IEC, "Software engineering. Metamodel for development methodologies," ISO/IEC International Standard 24744, ISO, Geneva, Switzerland, 2007.

[142] C. Gonzalez-Perez and B. Henderson-Sellers, *Metamodelling for Software Engineering*, J. Wiley & Sons, Chichester, UK, 2008.

[143] A. Omicini, A. Ricci, and M. Viroli, "Artifacts in the A&A metamodel for multi-agent systems," *Autonomous Agents and Multi-Agent Systems*, vol. 17, no. 3, pp. 432–456, 2008.

[144] J. Searle, *Speech Acts: An Essay in the Philosophy of Language*, Cambridge University Press, 1969.

[145] D. G. Firesmith and B. Henderson-Sellers, *The OPEN Process Framework. An Introduction*, Addison-Wesley, 2002.

[146] D. G. Firesmith and B. Henderson-Sellers, "Improvements to the OPEN process metamodel," *Journal of Object-Oriented Programming*, vol. 12, no. 7, pp. 30–35, 1999.

[147] M. Amor, L. Fuentes, and A. Vallecillo, "Bridging the gap between agent-oriented design and implementation using MDA," in *Proceedings of the 5th International Workshop on Agent-Oriented Software Engineering V (AOSE '04)*, J. Odell, P. Giorgini, and J. P. Müller, Eds., vol. 3382 of *Lecture Notes in Computer Science*, pp. 93–108, Springer, Berlin, Germany, 2004.

[148] K. Fischer, C. Hahn, and C. Madrigal-Mora, "Agent-oriented software engineering: a model-driven approach," *International Journal of Agent-Oriented Software Engineering*, vol. 1, no. 3-4, pp. 334–369, 2007.

[149] K. Taveter and L. Sterling, "An expressway from agent-oriented models to prototypes," in *Proceedings of the Agent-Oriented Software Engineering VIII (AOSE '07)*, M. Luck and L. Padgham, Eds., vol. 4951 of *Lecture Notes in Computer Science*, pp. 147–163, Springer, Berlin, Germany, 2007.

[150] G. Benguria, X. Larrucea, B. Elvesæter, T. Neple, A. Beardsmore, and M. Friess, "A platform independent model for service oriented architectures," in *Enterprise Interoperability New Challenges and Approaches*, G. Doumeingts, J. Müller, G. Morel, and B. Vallespir, Eds., pp. 23–32, Springer, London, UK, 2006.

[151] L. Liu, Q. Liu, and C. H. Chi, "Towards a service requirements modelling ontology based on agent knowledge and intentions," *International Journal of Agent-Oriented Software Engineering*, vol. 2, no. 3, pp. 324–349, 2008.

[152] F. Zambonelli, N. Jennings, A. Omicini, and M. Wooldridge, "Agent-oriented software engineering for internet applications," in *Coordination of Internet Agents: Models, Technologies, and Applications*, A. Omicini, F. Zambonelli, M. Klusch, and R. Tolkdorf, Eds., pp. 326–346, Springer, Heidelberg, Germany, 2001.

[153] B. Henderson-Sellers, "Creating a comprehensive agent-oriented methodology—using method engineering and the OPEN metamodel," in *Agent-Oriented Methodologies*, B. Henderson-Sellers and P. Giorgini, Eds., chapter 13, pp. 368–397, Idea Group, 2005.

[154] T. Juan, L. Sterling, and M. Winikoff, "Assembling agent oriented software engineering methodologies from features," in *Agent-Oriented Software Engineering III*, F. Giunchiglia, J. Odell, and G. Weiss, Eds., vol. 2585 of *Lecture Notes in Computer Science*, pp. 198–209, Springer, Berlin, Germany, 2003.

[155] S. A. DeLoach, L. Padgham, A. Perini, A. Susi, and J. Thangarajah, "Using three AOSE toolkits to develop a sample design," *International Journal of Agent-Oriented Software Engineering*, vol. 3, no. 4, pp. 416–476, 2009.