

# Representing Complex Multi-Agent Organisations in UML<sup>\*</sup>

Joaquin Peña, Rafael Corchuelo and Miguel Toro

Dpto. de Lenguajes y Sistemas Informáticos  
Avda. de la Reina Mercedes, s/n. Sevilla 41.012 (Spain)  
E-mail: joaquinp@lsi.us.es, web page: www.tdg-seville.info

**Abstract.** Interaction has been proved one of the main sources of complexity in Multi-Agent Systems (MAS) and many researches are working on techniques to palliate it. Furthermore, Organization modelling techniques lies on representing the groups of agents which are related by some kind of interaction. Current UML approaches represent these relationships as a set of binary links usually represented as stereotyped UML associations not providing abstraction tools to manage the complexity derived from interactions. In this paper, we argue for multiparty links in order to increase the level of abstraction of organization models and thus, their ability to manage complexity.

**keywords:** Complex systems, organization modelling, multiparty interactions, agent protocol descriptions, UML.

## 1 Introduction

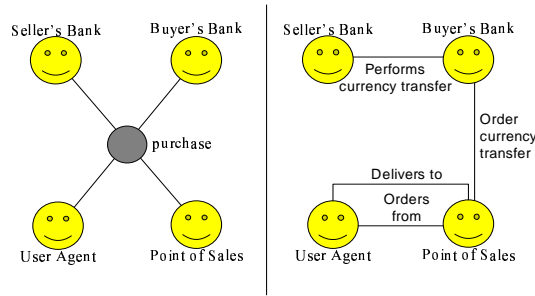
### 1.1 Organizations and Complexity

The organizational metaphor has been proved one of the most appropriate tools to engineer Multi-Agent Systems (hereafter MAS) being used as the abstraction which guides the analysis and design of MASs, e.g. [14]. Organization of Multi-Agent Systems is usually seen by many researchers as a collection of interacting roles [8], e.g. GAIA [14] or AUML social structures representations [11]. An organization shows the groups of agents formed in the system due to get benefits from one to another in a collaborative or competitive manner. As a matter of fact, it shows that an organization emerges when exists some kind of interaction between its participants (either through direct communication by means of speech acts or through the environment).

Most researchers agree on that MASs are a special kind of distributed systems (objects with their own threads of execution) with special features where a higher degree of complexity exists than in current Object-Based Software Systems [7]. This complexity of MASs is consequence of their features and mainly of their interacting nature: *Complexity is caused by the collective behaviour of many basic interacting agents.* James Odell [7].

---

<sup>\*</sup> The work reported in this article was partially supported by the Spanish Ministry of Science and Technology under grants TIC97-0593-C05-05



**Fig. 1.** Multiparty organization relationships *vs.* biparty relationships

Thus, one of the main goals of Agent Software Engineering (AOSE) is focused on dealing with complexity of MASs to which current software engineering techniques are not specially tailored [5,6]. In this sense, Odell *et al.* [1], Wooldridge and Jennings in [14], and others [11] has identified a set of techniques that cope with modelling such complex interaction/social structures.

## 1.2 The Need for More Abstract Organization Modelling Artifacts

Interaction modelling, and thus organization modelling, encompasses two aspects: i) *the structural aspect*: which models the relationships between artifacts in the system from the interaction point of view, e.g. as it is depicted in Figure 1, a seller's bank agents is linked with a buyer's bank agent by means of a relationship that represents that they must interact to perform a money transfer, and ii) *the behavioural aspect*: which models the order of apparition of these relationships over time, e.g. first the items to be purchased are chosen (orders items) to later purchase these items (orders money transfer), *etcetera*. Notice that other aspects such as environment modelling [11,14] and organization rules [14] are also crucial aspects which we do not cope with in this paper since they do not affect directly the complexity of models.

In order to be able to face large MASs a set of abstraction mechanisms must be proposed covering both aspects. By the best of our knowledge two techniques can be used:

On the one hand, the use of role modelling techniques provides a crucial tool for dealing with complexity since it allows us to "*divide and conquer*" seeing a complex organization as a set of separate sub-groups which can be studied separately.

On the other hand, multiparty relationships provides means for more abstract organization structure models than biparty links such as associations. As it is depicted in Figure 1, some relationships may involve more than two agents, e.g. a purchase where a buyer's bank a seller's bank, a User Agent and a Point of Sales Agent participates. If a conceptually atomic relationship at some level of abstraction is represented by means of biparty relations we have to divide it into

a set of biparty relations mentally thus decreasing the level of abstraction. For example, since in Figure 1 the purchase can be represented abstractly by a single four-party relationship if we limit to biparty relationships we have to represent it by four finer grain relationships. In large MAS this implies decreasing the level of abstraction of models from the beginning which in consequence decrease our capacity of dealing with complex systems.

Many researchers has identified this need providing interaction abstractions in order to encapsulate the protocol performed between an arbitrary set of agents, e.g. AUML nested protocols, MESSAGE interactions or GAIA protocols. Unfortunately, by the best of our knowledge the abstractions proposed in the literature do not cover simultaneously both structural and behavioural aspects using UML.

In this paper, we present part of the notation of our methodology to model complex Multiagent Systems (<http://www.tdg-seville.info/joaquinp/MaCMAS>). We present an interaction abstraction based on previous work [1,2,12,13,14] to graphically represent both aspects in UML 2.0 [9]. The main advantage of our approach is that we provide the mechanisms needed to add a more abstract model of the organization than other proposals that use UML. Furthermore, our proposal does not disable others but it provides a mechanism of abstraction which can be used to produce abstract simpler models which can be refined to reach the level of details applied in others.

This paper is organised as follows: in Section 2 we present the related work; in Section 3 we present a multiparty abstraction called multi-Role Interaction to represent organization relationships; in Section 4 we present a case study; in Section 5 we discuss on the most appropriate UML notation for mRIs and we illustrate it with the case study; in Section 6 we present the UML models we propose. Finally, in Section 7, we summarise our main contributions.

## 2 Related Work

There exists two paths in the organization modelling literature [11]: i) the behaviouristic perspective, and ii) the mentalistic perspective. The former sees organization as a collection of roles whose features shows the external interface that agents playing a role offer to the group, thus focusing on the macro-level and requiring to understand organization (behaviour and structure) in a programmatic way. The later describes organizations in terms of mental notions such us desires, believes, facts, rules, *etcetera*, thus focusing on the agent micro-level and requiring to understand the organization as a branch of entities which in conjunction offers a joint behaviour not described explicitly.

The frontier between both perspectives is placed between such systems that can be understood programmatically and such whose complexity makes impossible to treat them from a behaviouristic perspective<sup>1</sup>. This paper focus on the behaviouristic approach providing mechanisms of abstraction that allows us to

---

<sup>1</sup> Notice that a sharp separation can not be established

move the frontier of behaviouristic approach to more complex systems using UML.

#### i) **Structural Aspect**

In AUML [11] structural relations between roles in an organization are represented as binary UML associations which force designers to decompose mentally multiparty relations. We think that UML association is not the most appropriate UML artifact to represent interaction relationships since they have been traditionally used to represent information relationships which may lead designers to see these relationships as type relations. Notice that n-ary associations may be used allowing representing relationships more abstractly. Unfortunately, they are not usually used in current approaches and they also present the same semantic drawbacks.

Although AUML recognises the need for multiparty interactions which they called nested protocols, these multiparty interactions between roles are not used to represent structural aspects in their organization models [11]. GAIA also recognises the need for multiparty interactions which they called *protocols*. Unfortunately, they do not provide an UML notation. In MESSAGE Organization models acquaintance relations to represent organization structure. Unfortunately, these relations are based on stereotyped associations ignoring other native UML constructions that represent the same concepts and which fit better semantically with interaction relationships.

MESSAGE also provides the concept of interaction which may be also used to represent organization relationships. Unfortunately, authors do not show the relation between acquaintance relations and interactions. They show neither the UML construction on which interactions are based. Finally, MESSAGE is based on UML 1.3 which did not represent roles properly [4].

#### ii) **Behavioral Aspect**

Using multiparty links to represent organization structure requires for tailored tools that allows us to represents the sequences of execution of such abstract joint tasks.

GAIA represents such order by means of regular expressions assigned to each role in an organization based on FUSION notation [3]. For example, if a Role A participates in three joint tasks namely  $I_1$ ,  $I_2$  and  $I_3$  with roles B and C in all of them, the expression  $A = I_1I_2I_3$  shows that role A participates first in the protocol (joint task)  $I_1$ , to later participate in  $I_2$ , and to finalize with  $I_3$ . Unfortunately, they do not provide an UML-based graphical notations nor a way of representing the whole behaviour of an organization using a single model.

Finally, MESSAGE does not represent the sequences of interactions or acquaintance relations. Although MESSAGES workflows represent the sequence of tasks authors do not describe how interactions, acquaintance relations and workflow relate.

### 3 A First Class Abstraction to Model Organizations

An mRI is an *institutionalised pattern of interaction* that we propose as abstraction tool to represent multiparty relationships between an arbitrary number of roles. An mRI is the materialisation of a certain organization goal required in the system at the analysis stage, i.e. mRIs represents multiparty interactions that several agents playing the roles defined on it have to perform to achieve an organization goal.

The information represented by an mRI focuses on the nature of the joint process and not on how it is carried out.

They are the corner stone of our approach since organization is always described by means of this abstraction. Using mRI as the minimum modelling element, we do not have to take into account all the links required by a complex task (structural aspect) nor the messages that are exchanged to accomplish it (behavioural aspect) at stages where these details have not been identified clearly or are not even known. Obviously, when the level of detail of mRIs has been increased enough to clearly understand the system organization, biparty links for the structural aspect and messages descriptions for the behavioural aspect are the most adequate approach to define mRIs internally such as those proposed in AUML.

### 4 Case Study: The UN Security Council's Procedure to Issue Resolutions

The case study we use to illustrate our approach is a simplified version of the Modelling TC UN Security Council's Procedure to Issue Resolutions case study<sup>2</sup>. In [www.tdg-seville.info/joaquinp/MaCMAS/example](http://www.tdg-seville.info/joaquinp/MaCMAS/example) is available the complete model of the case study using our notation.

To pass a UN-SC resolution, the following procedure would be followed: 1) At least one member of UN-SC submits a proposal to the current *Chair*; 2) The *Chair* distributes the proposal to all members of UN-SC and set a date for a vote on the proposal; 3) At a given date that the Chair set, a vote from the members is made; 4) Each member of the security council can vote either FOR or AGAINST or SUSTAIN; 5) The proposal becomes a UN-SC resolution, if the majority of the members voted FOR, and no permanent member voted AGAINST; 6) The members vote one at a time; 7) The Chair calls the order to vote, and it is always the last one to vote; 8) The vote is open (in other words, when one votes, all the other members know the vote); 9) The proposing member(s) can withdraw the proposal before the vote starts and in that case no vote on the proposal will take place; 10) All representatives vote on the same day, one after another; 11) A vote is always finished in one day. The date of the vote is set by the chair.

---

<sup>2</sup> <http://www.auml.org/auml/documents/UN-Case-Study-030322.doc>

## 5 UML Notation

When an extension must be defined, OMG strongly recommends to base it on the most semantically near construction in order to avoid semantic mistakes or redundant language extensions [10, pag. 3-26]. In this sense, we have carefully studied UML to use the most appropriate modelling artifact to represent mRIs. In this work, we have based on the last version of UML: UML 2.0 [9]. The new features that it presents fits better with our purpose than previous versions where roles were not properly supported [4].

We have determined that from the dynamic modelling artifacts provided by UML 2.0, collaborations presents a quite similar semantic to mRIs. In [9] the OMG provides the following summary of collaborations <sup>3</sup>:

UML 2.0 [9, pag. 125]: *A behaviour of a collaboration will eventually be exhibited by a set of cooperating instances (specified by classifiers) that communicate with each other by sending signals or invoking operations. However, to understand the mechanisms used in a design, it may be important to describe only those aspects of these classifiers and their interactions that are involved in accomplishing a task or a related set of tasks, projected from these classifiers. Collaborations allow us to describe only the relevant aspects of the cooperation of a set of instances by identifying the specific roles that the instances will play. Interfaces allow the externally observable properties of an instance to be specified without determining the classifier that will eventually be used to specify this instance. Consequentially, the roles in a collaboration will often be typed by interfaces and will then prescribe properties that the participating instances must exhibit, but will not determine what class will specify the participating instances.*

As can be seen the semantics of collaborations fit properly with our purpose since they represent multiparty interactions using roles to abstractly describe the features of agents that will play them.

UML provides two notations for collaborations: the *internal structure notation* and the *composite structure notation*. The former shows the internals of the collaboration representing roles and their communication paths using biparty links which is not our purpose (similar to [11]). The latter shows the collaboration using a collaboration icon: a dashed ellipse which the name of the collaboration inside without detailing how it is carried out. Roles of the collaboration are shown as associations (CollaborationRole), e.g in Figure 2 we represent the organization formed to vote a proposal where three roles participates: *Chair*, *Voter* and *Observer*. At the end of CollaborationRoles we place the interface required by each role showing the external features that an agent playing a certain role may expose to the organization [9, pag. 131]. UML interfaces may contain services and attributes thus we place there the knowledge processed by each role and the services offered.

<sup>3</sup> Notice that the collaboration notation and semantics has change from previous UML versions

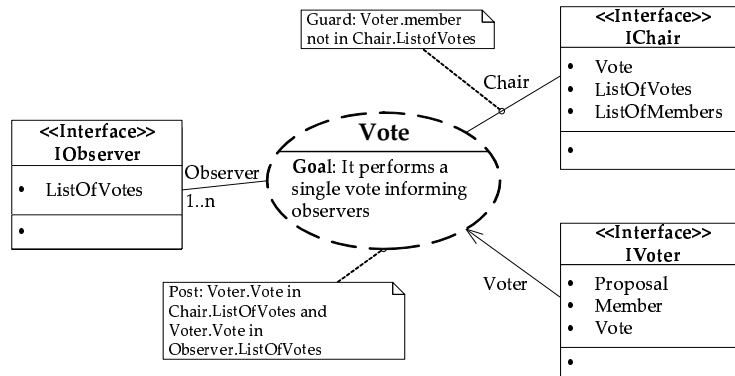


Fig. 2. mRI *Vote*

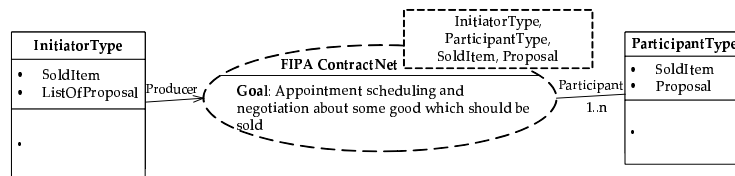


Fig. 3. Parametrised mRI of FIPA ContractNet Protocol

Although UML 2.0 collaborations do not define any attribute we have added the goal of the collaboration using a textual description. In Figure 2 this attribute can be observed inside the collaboration icon in a compartment with the name Goal. Furthermore, in order to represent the initiator of the mRI we represent them with an arrow from the interface to the collaboration, in our example the role *Voter*.

Each role that participates on an mRI can be decorated with a guard in order to indicate when it is interested on participating in it. Guards are graphically represented as textual notes linked with the association CollaborationRole. For example, the *Chair* will only participate in the mRI *Vote* if and only if the voter has not voted previously:  $Voter.Vote \notin Chair.ListOfVotes$ .

Finally, some interactions patterns can be generalised in order to reuse them. Parameterised mRIs are represented as parameterized UML collaborations adding a compartment to show the parameters of the mRI. It contains the concrete roles that participate in the mRI and the concrete knowledge that is managed. For example, in Figure 3, we show a parameterised mRI for the FIPA ContractNet Protocol where we can see that the Type of the *Producer* Role and *Consumer* role are open and also the knowledge that is exchanged. As this pattern should be well known to be reused, it can be attached with a FIPA parameterised pro-

to col description and a code framework that allows us to implement it almost directly.

## 6 UML Representation of Organizations

In our approach we represent an organization from two perspectives: i) the Organization structure model which shows statically all the relationships that may appear into an organization by means of UML collaborations and ii) the organization behaviour model which represent the order of apparitions of these links over time. In followings sections we detail both models.

Notice that our approach must be applied to complex systems at the early stages of modelling in order to clarify the organization of the systems thanks to abstract models we provide. AS a matter of fact, we do not cover how roles map into agents since this falls in the scope of finer grain techniques.

### 6.1 Organization Structure Model: Role Models

A role model represents an organization structure as a set of roles that relates by means of mRIs. They represent a partial view of the whole organization of the system where a certain organization goal of the MAS is represented orthogonally to the rest of them.

As we can see its definition is similar to mRI but the main difference is the level of detail that each concept represent. While an mRI represent a goal as a whole, a role model represents the goal as a set of mRIs, thus giving a detailed definition of the organization structure. Thus, when several mRIs are used to describe the same goal, each of them represents sub-goals of the general one.

There may exist a direct mapping between origination structures represented by means of a single mRI and role models which detail it. The refinement techniques presented in [12] constitutes a way for identifying the role model which represents an mRI internally.

Figure 4 shows the role model for the issue resolution organisational goal of our case of study where we can identify several mRIs: Accept/Reject proposal, Submit proposal, Vote, and withdraw proposal which model abstractly the whole case of study. Notice that, since in this role model each role is used by several mRIs several nested interfaces can be identified: one for each mRI linked to the main interface, e.g. the interface required by the role *Chair* for the *Vote* mRI it is nested in the interface *IChair*.

### 6.2 Organization Behavior Model: StateMachines and ProtocolStateMachines

The behavioural aspect of an organization, that it is to say, how the mRIs in a role model sequence, can be represented in two ways: a single dynamic view based on UML 2.0 State Machines [9, pag. 446] which represents the order of mRIs in the role model and a set UML 2.0 ProtocolStateMachines [9, pag. 422],

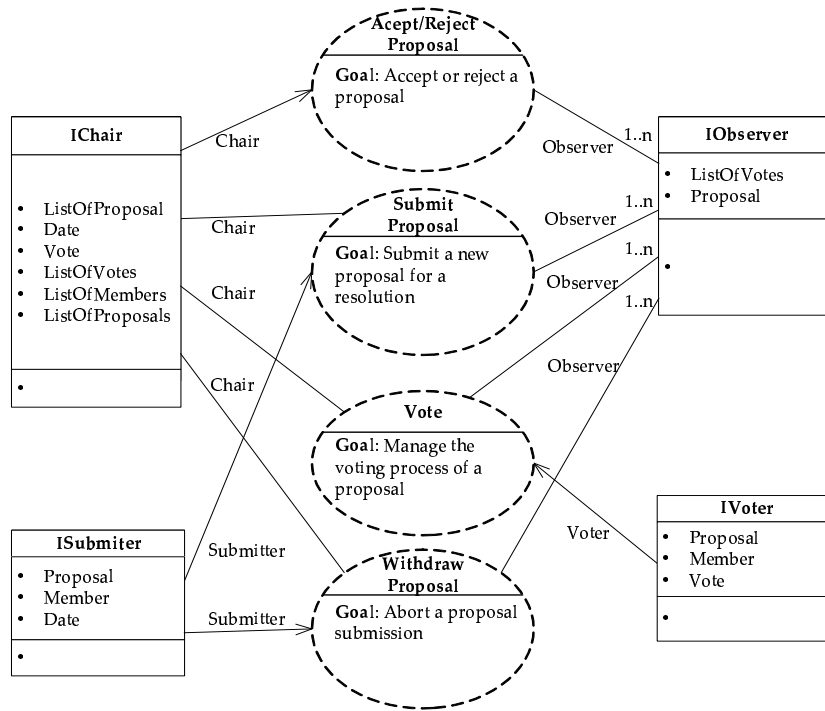
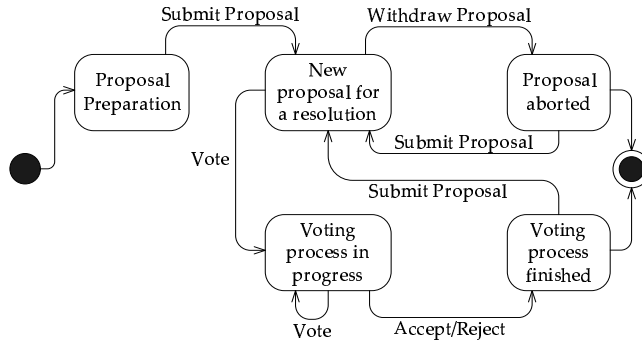


Fig. 4. Role Model Issue Resolution

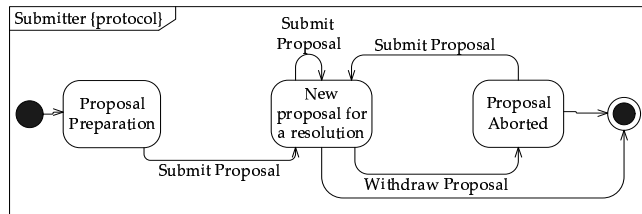
one for each role, which represents separately the behaviour of each role in the role model. In both cases, transitions are used to represent mRIs execution. UML defines them as follows:

**StateMachine:** *State machines can be used to express the behavior of part of a system. Behavior is modeled as a traversal of a graph of state nodes interconnected by one or more joined transition arcs that are triggered by the dispatching of series of events. During this traversal, the state machine executes a series of activities associated with various elements of the state machine.*

**ProtocolStateMachine:** *A protocol state machine is always defined in the context of a classifier. It specifies which operations of the classifier can be called in which state and under which condition, thus specifying the allowed call sequences on the classifier's operations. A protocol state machine presents the possible and permitted transitions on the instances of its context classifier, together with the operations which carry the transitions. In this manner, an instance lifecycle can be created for a classifier, by specifying the order in which the operations can be activated and the states through which an instance progresses during its existence.*



**Fig. 5.** State Machine of Role Model Issue Resolution

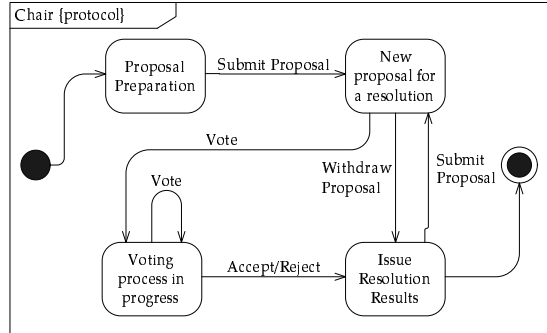


**Fig. 6.** Protocol State Machine of Role *Submitter*

As can be seen, StateMachines fit with the purpose of representing the behaviour of a part of the system represented in a role model, and ProtocolStateMachines are defined in the context of a classifier, which can be an interface, which fits also with our purpose of using them to represent the protocol executed by a single role.

In Figure 5 is represented the state machine that shows how to sequence the mRIs in the role model of Figure 4. Notice that each transition represents an mRI execution which requires all the roles involved to execute it. For example, the transition labelled with the mRI *Accept/Reject* requires that role *Chair* and *Observer* execute this mRI jointly. Notice that the guard of both roles for this mRI must hold to traverse the transition. In State Machines mRIs represent events that are produced/consumed by all the roles in it.

The another representation consist on providing a separate ProtocolState-Machine for each role in the Role Model which may be useful at design state to map several roles into the same agent. In Figures 7 and 6 are shown the protocol of some roles of the *Issue Resolution* role model. All of the role protocols execute its transitions coordinately [13]. Roughly speaking, when an mRI is executed by more than one role we must perform a transition in all of its protocols. Each of these transitions represents the part of the mRIs that each of them performs.



**Fig. 7.** Protocol State Machine of Role *Chair*

Whereby, to execute an mRI we must transit from one state to another in all the roles that participate in it.

For example, traversing the transitions *Submit Proposal* in the protocol of *Submitter*, Figure 6, implies to execute only this role part, but notice that as in the previous case, this transition can not be traversed unless the rest of roles of *Submit Proposal*, that is, the *Chair*, are in a state where this mRI is available and the guards of both hold. Thus, *Chair* must be in the state *Proposal Preparation* or *Issue Resolution Results*.

Finally, when several mRIs are available from a state, e.g. in Figure 5 from state *New Proposal for Resolution*, we can execute *Vote* or *Withdraw Proposal*, agents use guards to decide between them, e.g.  $CurrentDate() = Chair.Date$  for *Vote* and  $CurrentDate() < Chair.Date$  for *Withdraw Proposal*. If no guard is specified, the next mRI will be selected in a non deterministic way.

## 7 Summary

The main contribution of this paper is the identification of the most semantically appropriate UML artifacts to represent more abstract organisation models than current approaches. In this sense, we have provided three UML models to represent organizations: i) the organization structure model to represent relationships statically and ii) two equivalent organization behaviour models to show the order of apparition of these relations; one represents role behaviours isolated and another to represent organization behaviour as a whole both based on UML state machines.

Our approach must be applied to complex systems at the early stages of modelling in order to clarify the organization of the systems thanks to abstract models we provide. Then, other approaches based on finer grain descriptions can be applied.

## 8 Acknowledgments

This paper has been written as a result of the ideas discussed on the FIPA Modelling and Methodology TC with J. Odell, M. Consentino, R. Levy, D. Greenwood, G. Wagner, M. Huget, Hong Zhu, etcetera. We would like to thank all of them for their fruitful comments.

## References

1. B. Bauer, J. Müller, and J. Odell. Agent UML: A formalism for specifying multiagent interaction. *LNCS*, 1957:91–103, 2001.
2. G. Caire, F. Leal, P. Chainho, R. Evans, F. Garijo, J. Gomez, J. Pavon, P. Kearney, J. Stark, and P. Massonet. Agent oriented analysis using MESSAGE/UML. In *Proceedings of Agent-Oriented Software Engineering (AOSE'01)*, pages 101–108, Montreal, 2001.
3. D. Coleman *et al.* *Object Oriented Development: The Fusion Method*. Prentice-Hall, 1994.
4. R. Depke, G. Engels, and J. M. Küster. On the integration of roles in the uml. Technical Report 214, Dep. of Computer Science, University of Paderborn, August 2000.
5. N. Jennings. An agent-based approach for building complex software systems. *Communications of the ACM*, 44(4):35–41, 2001.
6. J. Odell. Agents (part 2): Complex systems, executive report. Technical Report Volume 3, Number 6, Cutter Consortium, Arlington, MA, 2000.
7. J. Odell. Agents and complex systems. *Journal of Object Technology*, 1(2):35–45, July-August 2002.
8. J. Odell, H. Parunak, and M. Fleischer. The role of roles in designing effective agent organizations. In A. Garcia and C. Lucena and F. Zambonelli and A. Omicini and J. Castro, editors, *Software Engineering for Large-Scale Multi-Agent Systems*, number 2603 in LNCS, pages 27–28, Berlin, 2003. Springer-Verlag.
9. Object Management Group (OMG). Unified modeling language: Superstructure. version 2.0. Final adopted specification ptc/03-08-02, OMG, August 2003. [www.omg.org](http://www.omg.org).
10. Object Management Group (OMG). Unified modeling language (UML), version 1.5. Technical report, OMG, March 2003. [www.omg.org](http://www.omg.org).
11. H. Van Dyke Parunak and James Odell. Representing social structures in UML. In Jörg P. Müller, Elisabeth Andre, Sandip Sen, and Claude Frasson, editors, *Proceedings of the Fifth International Conference on Autonomous Agents*, pages 100–101, Montreal, Canada, 2001. ACM Press.
12. J. Peña, R. Corchuelo, and J. L. Arjona. A top down approach for mas protocol descriptions. In *ACM Symposium on Applied Computing SAC'03*, pages 45–49, Melbourne, Florida, USA, 2003. ACM Press.
13. J. Peña, R. Corchuelo, and J. L. Arjona. Towards Interaction Protocol Operations for Large Multi-agent Systems. In M. Hinchey, J. Rash, W. Truszkowski, C. Rouff, and D. Gordon-Spears, editors, *Second International Workshop on Formal Approaches to Agent-Based Systems (FAABS 2002)*, volume 2699 of LNCS, pages 79–91, NASA, Greenbelt, MD, USA, 2002. Springer-Verlag.
14. F. Zambonelli, N. Jennings, and M. Wooldridge. Developing multiagent systems: the GAIA methodology. *ACM Transactions on Software Engineering and Methodology*, to be published 2003/2004.