Towards Interaction Protocol Operations for Large Multi-agent Systems

Joaquín Peña, Rafael Corchuelo, and José Luis Arjona

Dpto. de Lenguajes y Sistemas Informáticos
Avda. de la Reina Mercedes, s/n. Sevilla 41.012 (Spain)
joaquinp@lsi.us.es, www.lsi.us.es/~tdg

Abstract. It is widely accepted that role-based modelling is quite adequate in the context of multi-agent systems (MAS) modelling techniques. Unfortunately, very little work has been reported on how to describe the relationships between several role models. Furthermore, many authors agree on that protocols need to be encapsulated into high-level abstractions. The synthesis of role models is an operation presented in the OORAM methodology that allows us to build new role models from others in order to represent the interrelations they have. To the best of our knowledge this operation has to be performed manually at protocol level and works with protocols expressed by means of messages. In this paper, we present two algorithms to extract the protocol of a role from the protocol of a role model and vice versa that automate the synthesis or role models at the protocol level. Furthermore, in order to deal with protocol descriptions in a top down approach both operations work with protocols expressed by means of an abstraction call multi-role interaction (mRI).

1 Introduction

When a large system is modelled, complexity becomes a critical factor that has to be dealt with properly. In order to tackle complexity G. Booch recommended several powerful tools such as: Decomposition, Abstraction, and Hierarchy [5]. In addition, these tools were also presented as appropriate for Agent-Oriented Software Engineering (AOSE) for complex MAS, and were adapted to this field in [18] as follows:

- Decomposition: It is based on the principle *divide and conquer*. Its main advantage is that it helps to limit the designers scope to a portion of the problem.
- Abstraction: It is based on defining simplified models of the system that emphasises some details and avoid others. It is interesting since it limits the designer scope of interest and the attention can be focused on the most important details.
- Organisation/Hierarchy: It relies on identifying and managing the relationships between the various subsystems in the problem. It makes it possible to

group together various basic components and deal with them as higher-level units of analysis, and, provides means of describing the high-level relationships between several units.

Unfortunately, we think that these tools have not been carefully applied in the approaches that are appearing in this field. We have identified several problems in current methodologies that our approach tries to solve.

On the one hand, there exists a huge semantic gap in MAS protocol description methodologies because most of them first identify which tasks have to be performed, and then use low level descriptions such as sequences of messages to detail them. Although these messages may represent a high level view of a protocol, which shall be refined later, the tasks that are performed are formulated as a set of messages. This representation implies that the abstraction level falls dramatically since a task requires several messages to be represented. For instance, an information request between two agents must be represented with two messages at least (one to ask, and another to reply). This introduces a semantic gap between tasks and their internal design since it is difficult to identify the tasks represented in a sequence of messages. This representation becomes an important problem regarding readability and manageability of large MAS and can be palliated using the abstraction tool presented above.

On the other hand, in AOSE is widely accepted that describing the system as a set of role models that are mapped onto agents is quite adequate since it applies the *decomposition* tool [6,12,21,19,22,23,34]. Unfortunately, we have failed to find methodologies for MAS that use some interesting ideas about role modelling presented by Reenskaug and Andersen in the OORAM methodology [1,27]. Obviously, when we deal with a complex and large systems several role models may appear, and usually, they are interrelated. The role model synthesis operation [2], attempts to detail how role models are related, thus applying the *organisation tool*. This operation consists of describing new synthesised role models in terms of others. In a synthesised role model, new roles may appear and synthesised roles may also appear as aggregation of others. Unfortunately, OORAM also suffers from the first problem we have shown above since it deals with behaviour specification in terms of messages.

In this paper, we provide the first step towards the solution of these problems enumerated above using the tools proposed by Booch: i) In order to apply the *abstraction tool*, we have defined an abstraction called multi-role interaction (mRI) which encapsulates the interaction protocol (hereafter protocol) corresponding to a task that is performed by an arbitrary number of roles. mRIs are used as first modelling class elements to represent an abstract view of the protocol of a role model which can be refined with the techniques proposed in [25]. ii) In order to apply the *organisational tool*, we have also defined two operations on protocols (described in terms of mRIs) to automate and ease the synthesis operation since it operates on interaction protocols of a role instead of with the whole interaction protocol of a role model: the first one, called decomposition, infers a role protocol from a role model protocol automatically; and the second

one, that we called composition, infers a role model protocol from a set of role protocols automatically.

This paper is organized as follows: in Section 2, we present the related work and the advantages of our approach over others; in Section 3, we present the example we use; in Section 4, we present the protocol abstraction we have defined; in Section 5, we show how to describe the protocol of a role model; in Section 6, we present the algorithms to compose and decompose protocols, and, in Section 7, we present our main conclusions.

2 Related Work

In the context of distributed systems many authors have identified the need for advanced interaction models and have proposed multi-object interactions that encapsulates a piece of protocol between several objetcs [24]. Furthermore, most object-oriented analysis and design methods also recognise the need for coordinating several objects and provide designers with tools to model such multi-object collaborations. Different terms are used to refer to them: object diagrams [4], process models [7], message connections [8], data-flow diagrams [28], collaboration graphs [32], scenario diagrams [27], collaborations [13,29]. In MAS methodologies many authors have also proposed abstraction to model coordinated actions such as nested protocols [3], interactions [6] or micro-protocols [22], and so on. Unfortunately, the abstractions presented above are usually used to hide unnecessary details at some level of abstraction, reuse the protocol descriptions in new systems, and improve modularity and readability; however, most designers use message–based descriptions.

We think that most AOSE approaches model protocols at low level of abstraction since they require the designer to model complex cooperations as message-based protocols from the beginning. This issue has been identified in the GAIA Methodology [33], and also in the work of Caire *et. al.* [6], where the protocol description process starts with a high level view based on describing tasks as complex communication primitives (hereafter interactions). We think that the ideas presented in both papers are adequate for this kind of systems where interactions are more important than in object-oriented programming. As the methodologies GAIA and Caire's Methodology, we also use interactions (mRIs) to deal with the first stage of protocol modelling.

In the GAIA methodology, protocols are modelled using abstract textual templates. Each template represents an interaction or task to be performed between an arbitrary number of participants. In [6], Caire *et al.* propose a methodology in which the first protocol view is a static view of the interactions in a system. Later, the internals of these interactions are described using AUML [3].

Unfortunately, the operations we propose are difficult to be integrated with these methodologies. The reason why this happens is that we have found neither an interaction model for MAS able to describe formally a sequence protocol abstractions, nor operations on these high level protocol definitions. GAIA protocol descriptions, for example, are based on textual description thus it is difficult to

reason formally on them. In Caire's methodology, it is not shown how to sequence interactions. Although Koning *et al.* describe the sequence of execution of their abstraction using a logic-based formulae (*CPDL*), which consists of an extension of transition function of Finite State Automata (hereafter FSA), they do not define operations to operate with protocols. In our approach, we also define the sequence of mRI by means of Finite State Automaton (FSA) which has been also used by others authors at message level. We have chosen FSAs because this technique has been proved to be adequate for representing the behaviour of reactive agents [11,14,16,22].

Regarding the operations we present to the best of our knowledge the decomposition operation has not been defined before in this context. This operation can be useful for reuse, performing synthesis of role models since it operates with the protocol of a role instead of with the whole protocol and to map several protocol onto the same agent class. Unfortunately, in OORAM methodology such operation has to be applied manually to UML sequence diagrams.

The inverse operation, that we call composition, has been already defined by other authors, but, to the best of our knowledge, they do not use interaction with an arbitrary number of participants as we do [9,16,30,31]. This operation can be useful for building new role models reusing already defined role protocols stored in a behaviour repository, performing tests for adaptive behaviours [16], deadlock detection or to understand easily the protocol of a new role model [25]. Unfortunately, in OORAM this operation has to be also performed manually.

3 The Example

To illustrate our approach, we present an example in which a MAS delivers satellite images on a pay per use basis. We have divide the problem into two role models: one whose goal is obtaining the images (*images role model*) and the other for paying them (*purchase role model*). This decomposition of the problem allow us to deal with both cases separately.

In the *Images role model* the user (role *Client*) has to specify the images features that he or she needs (resolution, target, format, etcetera). Furthermore, we need a terrestrial centre (role *Buffer*) to store the images in a buffer because the throughput of a satellite (role *Satellite*) is higher than the average user can process and we need to analyse images features in order to determine their total price which is the goal of role *Counter*.

In the *Purchase role model* we need to contact the payment system to conclude the purchase. It involves three different roles: a customer role (Customer), a customer account manager role (*Customer's Bank*), and a terrestrial centre account manager role (*Buffer's Bank*). When a customer acquires a set of images he uses his or her debit–card to pay them, the agent playing role *Customer* agrees with a *Customer's Bank* agent and *Buffer's Bank* agent on performing a sequence of tasks to transfer the money from the customer account to the buffer account. If the *Customer's Bank* cannot afford the purchase because it has not enough money, the *Customer's Bank* agent then pays on hire–purchase.

4 Our Protocol Abstraction: Multi-role Interactions

The description of the protocol of a role model is made by means of mRIs. This provides an abstract view of the protocol that makes it easier to face the problem at the first stages of system modelling. Thus, we do not have to take into account all the messages that are exchanged in a role model in stages where these details have not been identified clearly.

A multi-role interaction (mRI) is an abstraction that we propose to encapsulate a set of messages for an arbitrary number of roles. At conceptual level, an mRI encapsulates each task that a role model should execute to perform its goal. These tasks can be inferred in a hierarchical diagram [20] where we can identify which tasks shall execute each role model.

mRIs are based on the ideas presented in two interaction models for distributed systems [15,10]. We have made that choice because both models have a set of formal tools that may be used for MAS systems improving the power of our approach, this allows, to perform deadlock testing and automatic interaction refinements [25] or efficient distributed implementations [26]. The definition of an mRI is:

$$\{(G(\beta)\} \ \& \ mRI\_name[r_1, r_2, \ldots, r_N]$$

Where mRI_name is an unified identifier of the interaction and $r_1, r_2, \ldots, r_N$ are the roles that execute the mRI mRI_name. β is the set of beliefs of agents playing the roles implied in the mRI and $G(\beta)$ is a boolean condition over β. This guard is partitioned in a set of subconditions, one for each role. $G(\beta)$ holds iff the conjunction of all subconditions of each role is *true*.

The idea behind guarded interactions has been adapted from the interaction model in which our proposal is based; furthermore, Koning *et al.* also adopt a similar idea. It promotes the proactivity of agents as we can see in [10,22] because agents are able to decide whether executing an mRI or not.

Thus, an mRI x shall be executed if the guard of the mRI holds and all roles that participate on it are in a state where the x is one of mRIs that can be executed. Furthermore, all of them must not be executing other mRIs since the interaction execution is made atomically and each role can execute only one mRI at the same time. For example, if we consider FSAs in Figure 3 after executing an mRI sequence that makes the the *Satellite* to be in state 1, the *Buffer* in state 4, the *Client* in state 8 and the *Counter* in state 11, if all the guards holds, we can execute *Receive*, *Send* or *LastSat*. In this case *LastBuffer* cannot be executed because it requires the *Buffer* to be in state 5.

Finally, for each interaction we should describe some details that we enumerate roughly below since it is not the purpose of this paper. To describe an mRI internally, we should include the sequence of messages using AUML. Furthermore, we may use coordination or negotiation patterns from a repository if its possible (FIPA has define a repository of interaction patterns [1]) and an objective

[1] http://www.fipa.org/specs/fipa00025/XC00025E.html

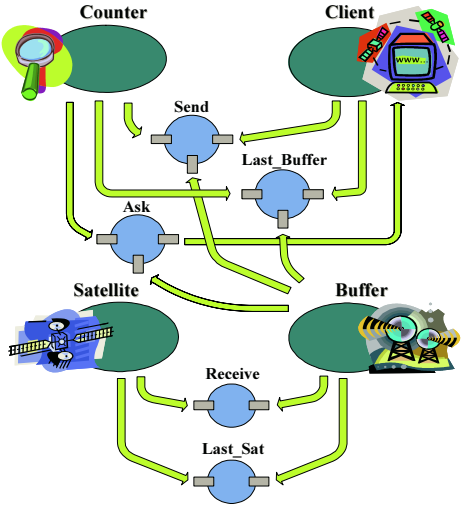

Fig. 1. Collaboration diagram of *Images role model*

function that determines which of available mRIs shall be better to execute if several of them can do so at the same moment.

Regarding the example, the description of one of the mRIs of the *Images role model* which it is used to ask for images (see Figure 3) is:

$$\{Counter.Connected(Buffer.ID())\&Counter.enable()\}\& \\ ask[Client, Buffer, Counter]$$

The rest of the mRIs in the *Images role model* are: *ask*, which is used to ask for images, *send*, which sends an image from the *Satellite* to the *Buffer*, *receive*, which sends an image from *Buffer* to *Client*, *LastSat*, which indicates the last image for transferring from *Satellite* to *Buffer* and stores information about images in a log file, and, $LastBuffer$, which indicates the last image for transferring from *Buffer* to *Client* and makes the *Counter* to calculate the bill. The static relation between these mRIs and the roles that perform them is represented in the collaboration diagram in Figure 1.

5 Modelling the Protocol of a Role Model

Once the roles and its mRIs have been identified we must describe how to sequence them. Thus, the protocol of a role model is defined as the set of sequences of mRIs execution it may performs. We can use two equivalent representations to describe the protocol of a role model (see Figure 2):

– Representing the protocol of the role model as a set of FSAs, one for each role (see Figure 3). Thus, in a role model with N roles we have N FSAs

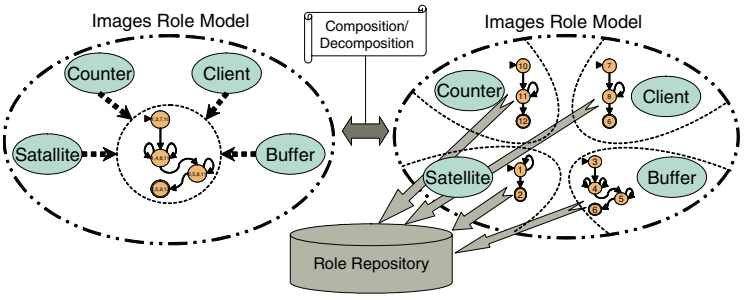

Fig. 2. Composition/Decomposition of protocol for the *Images role model*

A_i where each $A_i = (S_i, \Sigma_i, \delta_i, s_i^0, F_i)$, where S_i is a set of states, Σ_i is a vocabulary where each symbol $\sigma \in \Sigma_i$ represents an mRI, $\delta_i : S_i \times \Sigma_i \to S_i$ is the transition function that represents an mRI execution, $s_i^0 \in S_i$ is a initial state and $F_i \subseteq S_i$ is the set of final states. Thus, the set of words produced by this set of FSAs is set of possible traces of execution of mRIs. All this FSAs executes its transitions coordinately as it is shown in Section 4. Roughly speaking, when an mRI is executed by more than one role we must perform a transition in all of its participant roles. Each of these transitions represents the part of the mRIs that each of them performs. Whereby, to execute an mRI we must transit from one state to another in all the roles that participate in it.

– Representing the protocol of the role model as a whole using a single FSA for all the roles (see Figure 4). This FSAs is of the form $B = (S, \Sigma, \delta, s^0, F)$ where S is a set of states that represents one state for each FSA of roles, Σ is a vocabulary where each symbol $\sigma \in \Sigma$ represents an mRI, $\delta_i : S \times \Sigma \to S$ is the transition function that represents an mRI execution, $s_i^0 \in S_i$ is the initial state and $F \subseteq S$ is the set of final states. Thus, the set of words produced by this FSA is set of possible traces of execution of mRIs.

If we are dealing with a new role model, it may be more adequate to use a single FSA than one for each role since we see the problem in a centralised manner. The protocol for the *Images role model* by means of a single FSA is shown in Figure 4.

Once the protocol of all role models in our system have been described we can synthesise those role models that are interrelated. In our example, both role models are interrelated since the images obtained in the *Images role model* have to be paid using the *Purchase role model*.

In order to synthesise role models, we have to identify which roles are related and we have to merge their protocols to create the new synthesised role model. In our example, the synthesised role model *Purchase-Images role model* in Figure 5 is build by creating a new role where the protocol of the *Customer* and the *Client* is merged.

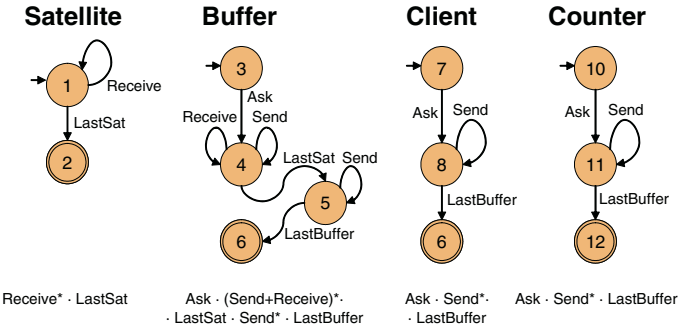

Fig. 3. FSAs of roles of *Images role model*

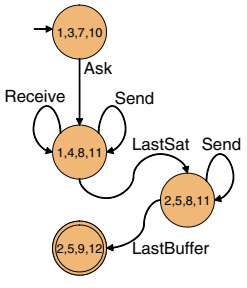

Fig. 4. FSA of *Images role model*

Thus, we have to know the protocol of both the *Customer* and the *Client* in order to build the new role model. This can be done using the decomposition operation.

Once we have built the protocol of the *Client/Customer* role, it is difficult to infer mentally which shall be the protocol of the new synthesised role model. Then, we can use the composition operation to infer it. In addition, we can perform deadlock testing in order to assure the correctness of the new protocol [25].

6 Composition and Decomposition of Interaction Protocols

These operations perform a transformation from one representation of protocol to another. As it is shown in the followings sections, these operations do not take guards into account. As we have shown above, a guard allows agents to decide if they want to execute an mRI or not. Thus, guards can make some execution traces of the protocol impossible. Unfortunately, we cannot determine this at design time. Even, if we are dealing with adaptive agents these decision

can change at runtime. Thus, in both operations, we work with the set of all possible traces leaving proactivity as a runtime feature.

6.1 Composition

The composition operation is an algorithm that builds a role model FSA from a set of FSA of roles obtained from a behaviour repository or from synthesis of role models.

To represent the role protocol of each role in a role model we use the FSAs $A_i = (S_i, \Sigma_i, \delta_i, s_i^0, F_i)$ $(i = 1, 2, \ldots, N)$. Thus, the composition algorithm is defined as a new FSA of the form $B = (S, \Sigma, \delta, s^0, F)$, where:

- $S = S_1 \times \ldots \times S_N$,
- $\Sigma = \bigcup_{i=1}^{n} \Sigma_i$,
- $\delta(a, (s_1, \ldots, s_n)) = (s'_1, \ldots, s'_N)$ iff $\forall\ i \in [1..N] \cdot (a \notin \Sigma_i \wedge s_i = s'_i) \vee \vee\ \ (a \in \Sigma_i \wedge \delta(a, s_i) = s'_i)$,
- $s^0 = (s_1^0, \ldots, s_N^0)$, and
- $F = F_1 \times \ldots \times F_N$.

This algorithm builds the new FSA exploring all the feasible executions of mRIs. Their states are computed as the cartesian product of all states. Each state of this FSA is formed by a N-tuple that stores a state of each role. To execute an mRI, we have to preform it from a tuple-state where the mRI can be execute to change to a new tuple-state where the states of roles implied in the mRI shall only change. Thus, for each new tuple-state we check if an mRI may be executed (all their roles can do it from its corresponding state in the tuple-state); if so, we add it to the result. Finally, the final state of the role model FSA is formed of all possible combinations of final states of each A_i and the initial state is a tuple with the initial state of each A_i.

Intuitively, it is easier to comprehend a protocol if it is described by means of a single FSA than if we use a set of them. Furthermore, we can perform deadlock testing on it to assure that the synthesis we have made is deadlock free and results in what we have thought when we synthesised them. Furthermore, this representation is easier to understand than several separated FSAs. With this operation, we can obtain automatically the FSA in Figure 4 that represents the protocol executed by the FSAs in Figure 3 of *Images role model*.

6.2 Decomposition

To obtain the protocol of a role we must take into account the mRIs a role execute only. That is to say, we can take all the possible traces that the FSA of the role model produces and ignore the mRIs that the role does not execute. For instance, if we take the a trace *(Ask, Receive, Receive, Send, LastSat, Send, LastBuffer)* from the FSA of the *Images role model*, the trace that the role *Satellite* executes is *(Receive, Receive, LastSat)* since it participates only in mRIs *Receive* and

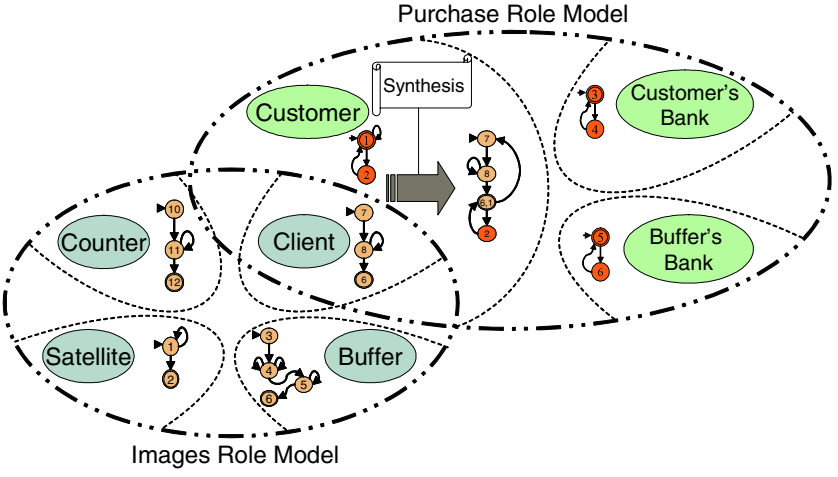

Fig. 5. Example of synthesis operation. Images and Purchase role models

LastSat. If we perform the same operation for all the traces produced by the FSA of a role model for each of its roles, we can obtain their protocol.

In [17, page 60] it is proved that if we take a regular expression r we can obtain a new regular expression $r' = h(r)$ where h is a function that replaces each occurrence of symbols a in r by another regular expression obtaining r'. If $L(r)$ is the language accepted by r, it is proved that $h(r) = h(L(r)) = r'$, that it is to say, the language accepted by r' is the same language accepted by applying h(r) to every word in $L(r)$.

Thus, we can perform the decomposition operation to obtain the FSAof each role from the FSA of a role model as follows:

- We obtain the regular expression of the FSA of the role model using one of the algorithms presented in [17]. In our example, it is: $Ask \cdot (Send + Receive)^* \cdot LastSat \cdot Send^* \cdot LastBuffer$.
- We take the regular expression of each role and we replace the mRISs where the role do not participate by ϵ (empty word). For example, the result for the *Client* role is: $Ask \cdot Send^* \cdot LastBuffer$.
- Finally, we build the FSA corresponding to each regular expression obtained in the previous step using the algorithms presented in [17].

As we illustrate in Figure 2, we can use this algorithm to obtain the FSAs showed in Figure 3 from the FSA in Figure 4. If we compose then again, we obtain the same result.

As we have sketched in Section 5, we can use one of the FSA obtained to build a new synthesised role where the *Customer* and the *Client* are merged generating the protocol represented in Figure 5. In this Figure the mRI *Approval* is used to check if the *Customer* has enough money to perform the purchase, $Transfer$ is used to transferring the money from *Customer's Bank* to *Buffer's Bank*, and

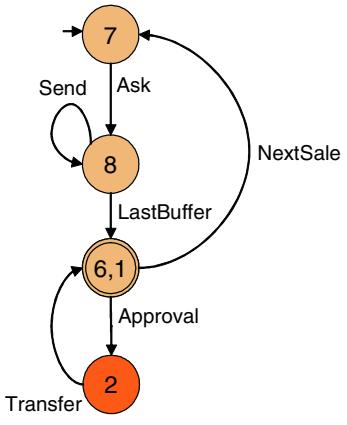

Fig. 6. FSAs of synthesised role: Customer and Client

NextSale prepares the *Customer* and *Client* for next purchase. Finally, we can compose the protocol of both role models in order to improve the readability of the description or to perform deadlock testing.

7 Conclusions

In this paper, we have presented an approach that puts together several ideas from different research contexts to ease protocol modelling of large MASs since our approach provides the tools to decompose, abstract and organise the descriptions.

With this purpose, we have presented two operations to automatically tackle with protocols. These operations are useful for large MASs since they allow us to define the interrelation of several role models as a new synthesised role model without dealing with protocols manually. Furthermore, they ease the reuse of protocol already defined for other systems.

In our approach, we also propose mRIs for avoiding to determine all the messages that agents have to interchange in early modelling stages. This allows us to describe protocols in a layered method since we can describe the tasks that a role model has to perform at a high level of abstraction to describe each mRI internally later.

References

1. E. Andersen. *Conceptual Modeling of Objects: A Role Modeling Approach*. PhD thesis, University of Oslo, 1997.
2. E. P. Andersen and T. Reenskaug. System design by composing structures of interacting objects. In Ole Lehrmann Madsen, editor, *ECOOP '92, European Conference on Object-Oriented Programming, Utrecht, The Netherlands*, volume 615 of *Lecture Notes in Computer Science*, pages 133–152. Springer-Verlag, New York, NY, 1992.

3. B. Bauer, J. Muller, and J. Odell. Agent uml: A formalism for specifying multiagent interaction. In M. Wooldridge and P. Ciancarini, editors, *Proceedings of 22nd International Conference on Software Engineering (ISCE)*, LNCS, pages 91–103, Berlin, 2001. Springer-Verlag.
4. G. Booch. *Object-Oriented Design with Applications*. Benjamin/Cummings, Redwood City, CA, 1990.
5. G. Booch. *Object–Oriented Analysis and Design with Applications*. Addison-Wesley, 2 edition, 1994.
6. 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.
7. D. de Champeaux. Object-oriented analysis and top-down software development. In *Proceedings of the European Conference on Object-Oriented Programming, ECOOP'91*, volume 512 of *Lecture Notes in Computer Science*, pages 360–375. Springer–Verlag, 1991.
8. P. Coad and E. Yourdon. *Object-Oriented Analysis*. Computing Series. Yourdon Press, Englewood Cliffs, NJ, 1990.
9. J. C. Corbett. Evaluating deadlock detection methods for concurrent software. *IEEE Transactions on Software Engineering*, 22(3):161–180, March 1996.
10. J. C. Cruz. OpenCoLaS a coordination framework for CoLaS dialects. In *Proceedings of COORDINATION 2002*, York, United Kingdom, 2002.
11. D. Denning. *Information warfare and security*. Addison-Wesley, Reading, MA, USA, 1999. ACM order number 704982.
12. R. Depke, R. Heckel, and J. M. Kuster. Improving the agent-oriented modeling process by roles. In Jörg P. Müller, Elisabeth Andre, Sandip Sen, and Claude Frasson, editors, *Proceedings of the Fifth International Conference on Autonomous Agents*, pages 640–647, Montreal, Canada, May 2001. ACM Press.
13. D.F. D'Souza and A.C. Wills. *Objects, Components, and Frameworks with UML: The Catalysis Approach*. Addison-Wesley, Reading, Mass., 1999.
14. L. J. Fogel, A. J. Owens, and M. J. Walsh. *Artificial Intelligence through Simulated Evolution*. John Wiley & Sons, New York, 1966.
15. N. Francez and I. Forman. *Interacting processes: A multiparty approach to coordinated distributed programming*. Addison-Wesley, 1996.
16. D. F. Gordon. APT agents: Agents that are adaptive, predictable, and timely. *Lecture Notes in Computer Science*, 1871:278–293, 2001.
17. J. E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, Reading, Massachusetts, 1979.
18. N. R. Jennings. Agent-Oriented Software Engineering. In Francisco J. Garijo and Magnus Boman, editors, *Proceedings of the 9th European Workshop on Modelling Autonomous Agents in a Multi-Agent World : Multi-Agent System Engineering (MAAMAW-99)*, volume 1647, pages 1–7. Springer-Verlag: Heidelberg, Germany, 30– 2 1999.
19. E. Kendall. Agent software engineering with role modelling. In P. Ciancarini and M. Wooldridge, editors, *First International Workshop of Agent-Oriented Software Engineering, AOSE 2000*, number 1957 in LNCS, pages 163–170, Limerick, Ireland, June 2001. Springer-Verlag.
20. E. Kendall, U. Palanivelan, and S. Kalikivayi. Capturing and structuring goals: Analysis patterns. In *Proceedings of the* 3^{rd} *European Conference on Pattern Languages of Programming and Computing*, Germany, July 1998.

21. E. A. Kendall. Role modeling for agent system analysis, design, and implementation. *IEEE Concurrency*, 8(2):34–41, April/June 2000.
22. J. Koning, M.Huget, J. Wei, and X. Wang. Extended modeling languages for interaction protocol design. In M. Wooldridge, P. Ciancarini, and G. Weiss, editors, *Proceedings of Second Internationa Workshop on Agent-Oriented Software Engineering (AOSE'02)*, LNCS, Montreal, Canada, May, 2001. Springer-Verlag.
23. J. Odell, H. V. D. Parunak, and B. Bauer. Representing agent interaction protocols in uml. In *Proceedings of the 1^{th} Int. Workshop on Agent-Oriented Software Engeenering (AOSE'2000)*, number 1957 in LNCS, page Apendix, Limerick, Ireland, June 2000. Springer-Verlag.
24. G. Papadopoulos and F. Arbab. Coordination models and languages. In *Advances in Computers*, volume 46. Academic Press, 1998.
25. J. Peña, R. Corchuelo, and J. L. Arjona. A top down approach for mas protocol descriptions. In *ACM Symposium on Applied Computing SAC'03*, page to be published, Melbourne, Florida, USA, 2003. ACM Press.
26. J.A. Pérez, R. Corchuelo, D. Ruiz, and M. Toro. An order-based, distributed algorithm for implementing multiparty interactions. In *Fifth International Conference on Coordination Models and Languages COORDINATION 2002*, pages 250–257, York, UK, 2002. Springer-Verlag.
27. T. Reenskaug. *Working with Objects: The OOram Software Engineering Method*. Manning Publications, 1996.
28. J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen. *Object-Oriented Modeling and Design*. Prentice Hall, Schenectady, New York, 1991.
29. J. Rumbaugh, I. Jacobson, and G. Booch. *The Unified Modeling Language Reference Manual*. Object Technology Series. Addison Wesley Longman, Reading, Massachussetts, 1999.
30. C. Shih and J. A. Stankovic. Survey of deadlock detection in distributed concurrent programming environments and its application to real-time systems. Technical Report UM-CS-1990-069, 1990.
31. M. Y. Vardi and Pierre Wolper. An automata-theoretic approach to automatic program verification (preliminary report). In *Proceedings 1st Annual IEEE Symp. on Logic in Computer Science, LICS'86, Cambridge, MA, USA, 16–18 June 1986*, pages 332–344. IEEE Computer Society Press, Washington, DC, 1986.
32. R. Wirfs-Brock and B. Wilkerson. *Designing Object-Oriented Software*. Prentice-Hall, August 1990.
33. M. Wooldridge, N. R. Jennings, and D. Kinny. The gaia methodology for agent-oriented analysis and design. *Autonomous Agents and Multi-Agent Systems*, 3(3):285–312, 2000.
34. F. Zamboneli, N. R. Jennings, and M. Wooldridge. Organizational abstraction for the analysis and design of multi-agent system. In *Proceedings of the 1^{th} Int. Workshop on Agent-Oriented Software Engeenering (AOSE'2000)*, number 1957 in LNCS, pages 235–252, Limerick, Ireland, June 2000. Springer-Verlag.