A Novel Formal Specification Approach for Real Time Multi-Agent System Functional Requirements

Mohamed Amin Laouadi[1], Farid Mokhati[2], and Hassina Seridi-Bouchelaghem[1]

[1] Computer Science Department, Badji Mokhtar University, Annaba, Algeria. LabGED Laboratory
Laouadiamin@yahoo.fr, seridi@labged.net
[2] Computer Science Department, Oum El-Bouaghi University, Algeria
mokhati@yahoo.fr

Abstract. A novel formal functional requirements specification approach for real-time multi-agent system is presented in this paper. The methodology of our approach consists in translating extended AUML diagrams describing RT-MAS' functional requirements into a RT-Maude specification. The proposed approach considers jointly functional, static and dynamic aspects of real-time multi-agent systems. The functional aspects are described by a temporal AUML use case diagram and the static aspects are represented using a temporal AUML class diagram. Whereas the dynamic aspects are described using state chart (individual behavior) and an extended AUML protocol (collective behavior) diagrams. The aims of this approach are, on the one hand, to combine the advantages of the graphical modeling formalism Agent UML and the formal specification language RT-Maude in a single technique, and, on the other hand, to integrating the formal validation of the consistency of the models, since the analysis phase. The approach is illustrated using a concrete example.

Keywords: Formal specification, Real-time Multi-Agent System, Functional Requirements, Agent UML, RT-Maude, Supply Chain Management (SCM).

1 Introduction

A recent trend in the development of distributed real-time systems is the use of real-time multi-agent system. In contrast to conventional MAS, the real time MAS reflect intrinsic real-time systems characteristics, more precisely, the time constraints. For many years, MAS designers have development methodologies and modeling language without reflects the different temporal restrictions that these systems may have. Moreover, even the proposed methodologies for the development of real time MAS as: 'RT-Message' [1], 'BDI-ASDP extended for real time' [2] and 'Development Method of Lichen Zhang' [3], are inadequate. They have certainly made important responses in the development process of real-time MAS. However, the methodological aspect is not yet mastered. Indeed, none of these methodologies takes into account the functional requirements formalization of the future system.

J. Dix and C. Witteveen (Eds.): MATES 2010, LNAI 6251, pp. 15–27, 2010.

Formalizing the functional requirements of real-time MAS is in our opinion, an importance way for verification and validation activities. Furthermore, the MAS design requires the involvement with software engineering techniques. The main objective of this work is to offer a generic approach for a use case oriented specification of real time MAS functional requirements. Among these techniques: UML [4] is probably the best known and most widely used languages for object-oriented modeling. The MAS developers have recently the same facilities in particular the language Agent UML [5] [6]. However, there is currently no work applying Agent UML to real-time MAS specification and both to real time applications.

The proposed methodology differs from other modeling methodologies by the use of the AUML language extensions presented in [6] [7] [8], which have an agent-oriented development view inspired from object-oriented development. The AUML language is from our own point of view the future industry standard for agents oriented systems development. AUML models describe several complementary views of the same system but suffer as UML of a lack of formal semantics. AUML models may therefore contain inconsistencies which are difficult to detect manually.

Formal methods represent an interesting solution to this problem. The formal specifications will have the effect of eliminating the ambiguities in the models interpretation. The Agent UML combination with the RT-Maude formal specification language will formally validate developed AUML models.

This work takes place in the context of Software Engineering and Distributed Artificial Intelligence and aims to support the verification and validation of real-time MAS as an important discipline of Agent Based Software Engineering. Therefore, the main interest in this work is to describe, as a first step, the functional requirements of real time MAS using the graphical modeling formalism Agent UML, and translate these descriptions in RT-Maude.

The remainder of this paper is organized as follows: In section 2 we give a brief overview of major related works. Section 3 is devoted to the formal specification language RT-Maude. In section 4, we present the AUML extensions. Section 5 gives the proposed translation process. Section 6 illustrates the translation and validation processes using a case study. Finally, we give a conclusion and some future work directions in section 7.

2 Related Works

Several methodological proposals for software development exist and can be applied to agents systems. Inspired from knowledge based systems domain [9], or directly focusing the agents' properties [10], or object-oriented development methodologies and languages extensions as UML [6] but only a few of these methodologies are taking into account the agent temporal behavior [2].

Among these methodologies that directly addressing the design of real-time multi-agent systems, we are interested by: the Methodology RT-Message [1], the extended BDI-ASDP methodology for real time [2] and the development method of Lichen Zhang [3]. For a description of real-time agents, these three methodologies use different models namely: domain model, role model, and timed model (Table 1.)

Table 1. Real-Time Agent identification Approaches

	Domain Model	Role Model	Timed Model	Functional Requirements
The RT- Message Methodology [1]	✓	✓	✓	
Extended BDI ASDP methodology for real time [2]	✓		✓	
Zhang development method[3]	✓		✓	

The RT-Message methodology [1], uses a domain model to define the concepts inherent to the environment where agents are located. The main result of this model is a domain diagram which is basically a class diagrams containing all the relevant variables and entities in the development process, like it was proposed in Zhang [3]. However, in the extended BDI-ASDP methodology for real time [2], a symbolic model of the environment is defined based on the decomposition of the problem with Beliefs, desires and intentions that represent agent's information, motivations and decisions.

The concept of role is only present in RT-Message [1], where the roles are identified independently of the agent system. Regarding the modeling of temporal constraints of real-time agents, each methodology offers an approach: for the RT-Message case, extensions made on the different models imported from the MESSAGE method [11] [12] allow analyzing the MAS for real-time environments. For example, "the Goal / Task model" has been modified to incorporate a taxonomy of goals (Goal taxonomy) which takes into account temporal criteria. When specifying the goal's different types, it is necessary to extend the goal and task patterns of the method "message" for integrating the real-time features. The artifacts obtained are a set of 'implications diagrams' showing the relationship between goals and tasks. Subsequent, in extended BDI-ASDP for real time, proposed by Melián et al. [2], the temporal constraints modeling is done through "the timing diagrams" specified in UML 2.0. To satisfy the need to model real-time systems by the agent approach, Zhang [3] proposed to extend UML by introducing a new stereotype, called <<agents>>. The timing characteristics are specified as an instance of this stereotype called <<TimeAspect>>. This stereotype uses a timed model developed independently, according to the principle of AOP (Aspect Oriented Programming), to express the temporal aspect of a real-time system.

However, the real-time MAS modeling is frequently linked to functional specifications in the sense that those specifications provide a basis for describing the functional requirements of agents, applying a set of software engineering techniques. These functional descriptions are often modeled by UML that is the most widely deployed standard, providing multiple notations. This concept has been neglected in these analyzed methodologies (Table 1).

In fact, these methodologies don't focus on the real time MAS functional requirements formalization during their development process. Hence, it is important that they will supplemented by methods that strongly encourage the formalization of

the functional requirements captured in the analysis for the upstream phases of software engineering process.

3 Real Time Maude

Real-Time Maude is a programming language (an extension of Maude [13]) that was designed to exploit the concepts of the real-time rewrite theory. A real-time rewrite theory is a Maude rewrite theory, which also contains the specification of [14]:

- sort *Time* to describe the time domain,
- sort *GlobalSystem* with a constructor '{_}": {_} : *System* -> *GlobalSystem*
- And a set of tick rules that model the elapsed time in the system that have the following form: {t} => {t'} in time if condition μ

Where μ is a term which may contain variables, of sort *Time* that denotes the length of the rule, and the terms *t* and *t'* are terms of sort *System*, which denotes the state of the system. The rewriting rules that are not tick rules are rules supposed to take a time instant zero. The initial state must always have the form {t"}, where *t"* is a term of sort *System*, so that the form of tick rules ensures that time flows uniformly in all parts of the system. Real-time rewrite theories are specified in Maude as timed modules or timed object-oriented modules.

4 Extended Agent UML

We present in this section some extensions to AUML diagrams in order to describe functional requirements of real-time MAS.

4.1 Temporal AUML Use Case Diagrams

Given that, the analysis based on use cases proven in specifying the requirements of MAS [15], [16], [17]. This motivated us to apply these same techniques but with some modifications on real-time multi-agent system modeling. By using the extension mechanism of "stereotyping" offered by the language Agent UML, use case diagrams will be enriched by the following 'five stereotypes':

- The stereotypes « Agent Use Case » (Fig. 1.a) and « Temporal Agent Use Case » (Fig. 1.b) are used to denote respectively, a use case that represent a functionality performed by agents and a use case that interact with real-time agents.

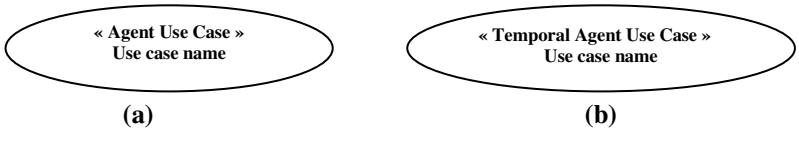

Fig. 1. Stereotyped use cases

- The stereotypes « Agent » (Fig 2.a), « External Agent » (Fig 2.b) and « Real Time Agent » (Fig. 2.c) describe in this order: the agents within the system, external agents to the system and real time agents which are internal agents.

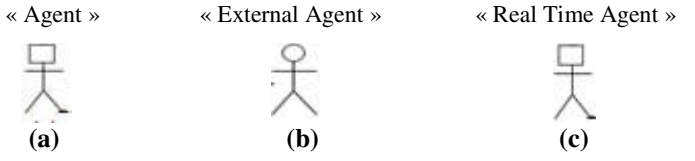

Fig. 2. Agent notations

4.2 Temporal AUML Class Diagrams

The two levels of abstraction proposed by Huget [8] are studied, when designing class diagrams: "the conceptual level and implementation level". The first level is unchanged and gives a high view of MAS, while 'second level' gives in detail the agents contents and modified as follows: to all compartments proposed in [8], a new compartment called "temporal constraints" is added (Fig. 3).

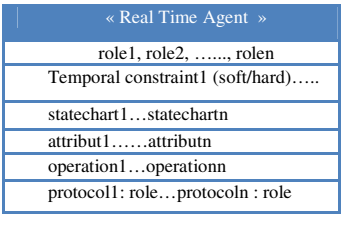

Fig. 3. The used AUML class diagram

To describe agents' individual and collective behaviors we use respectively the AUML state-chart and protocol diagrams.

5 The Proposed Approach

The proposed translation process aims to translate the AUML diagrams described above (Section 4) for describing real time MAS functional requirements to RT-Maude formal specifications. This process is divided into three major steps (Fig. 4): (1) Description of real-time multi-agent system functional requirements using AUML diagrams, (2) inter-diagrams validation, and (3) Generation of RT-Maude formal description.

The first step is the usual analysis phase of software development process. The second step aims to validate the coherence between the designed models. The last step is the systematic generation of RT-Maude source code from the considered AUML diagrams. The formal framework proposed (Fig. 5) is composed of several modules: nine functional modules, seven object-oriented modules, and four timed

object-oriented modules. For reason of limitation of space, we present only the main modules of the proposed framework. The Module STATE describes agents' states; the ACTION module describes the actions types that an agent can use. These two last modules and the CONDITION module (which is used to define the type *Condition*) are imported into STATE-OPERATIONS module to define the operations related to agent's states.

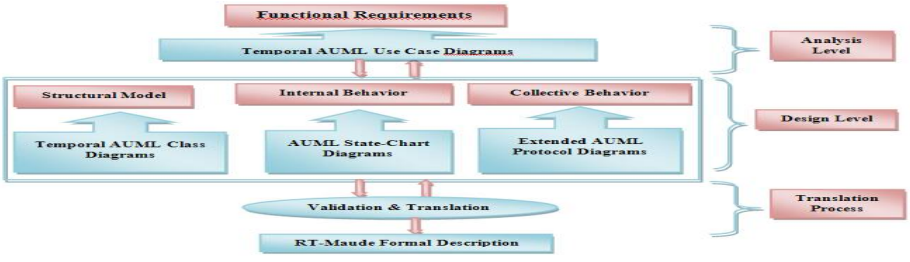

Fig. 4. Methodology of the approach

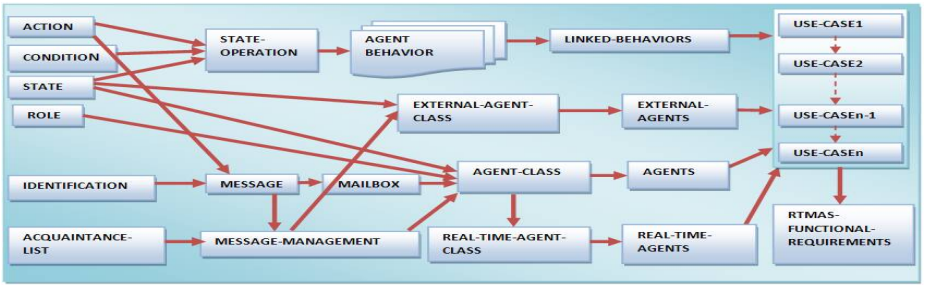

Fig. 5. Generated Modules

AGENTi-BEHAVIOR modules that import STATE-OPERATIONS module are used to illustrate the behavior of individual agents. In order to respect interactions between the different agents, connections between them are performed through the LINKED-Behavior module, which reuses AGENTi-BEHAVIOR modules. The identification mechanism for agents is defined by the IDENTIFICATION module, and message structure description exchanged between the various agents is done via MESSAGE module that imports the IDENTIFICATION, and ACTION modules.

Communicating agents are generally endowed with a Mailbox containing the received messages of other agents and a list of its acquaintances. For that, we define the functional modules MAILBOX and ACQUAINTANCE-LIST to manage respectively Mailboxes and acquaintance lists of agents. Agents' roles are defined in the module ROLE. To describe the sending/ receiving operations, we define module MESSAGE-MANAGEMENT which imports AQUAINTANCE-LIST and MESSAGE modules.

The object oriented module EXTERNAL-AGENT-CLASS (Fig. 6) is used to define the base class of external agents, with attributes *CurrentState* and *AcqList* (line [1])

```
(omod EXTERNAL-AGENT-CLASS  is
protecting  STATE . protecting  MESSAGE-MANAGEMENT .
class ExtAgent | CurrentState : State, AcqList :
AcquaintanceList .        ---[1]   endom)
```

Fig. 6. The O.O Module EXTERNAL-AGENT-CLASS

which represent the agent's current state and its acquaintances list. This module imports STATE, and MESSAGE-MANAGEMENT modules.

In the object oriented module AGENT-CLASS (Fig. 7), we define the internal agents' base class structure. This class (line [1]) has as attributes: *PlayRole*, *CurrentState*, *MBox* and *AcqList* to contain in this order: the role played by the agent, its current state, its mailbox and its acquaintances list. This module imports all the modules: STATE, ROLE, MAILBOX, and MESSAGE-MANAGEMENT.

```
(omod AGENT-CLASS is protecting STATE. protecting ROLE.
protecting MAILBOX .  protecting MESSAGE-MANAGEMENT .
class Agent | CurrentState : State, PlayRole : Role,
AcqList: AcquaintanceList, MBox : MailBox .--[1] endom)
```

Fig. 7. The O.O Module AGENT-CLASS

To describe the real-time agents, we have defined the *RealTimeAgent* class with the attribute *Clock* (line [1]) in the timed object oriented module REAL-TIME-AGENT-CLASS (Fig. 8) as a subclass of *Agent* Class (line [2]).

```
(tomod REAL-TIME-AGENT-CLASS is extending AGENT-CLASS .
class RealTimeAgent | Clock : Time . ---[1]
subclass RealTimeAgent  <  Agent .   ---[2]  endtom)
```

Fig. 8. The Timed O.O Module REAL-TIME-AGENT-CLASS

To each use case is associated one timed O.O module USE-CASEi (Fig. 9), which has the same name as the corresponding use case. In each module USE-CASEi are defined the rewriting rules describing the different interaction scenarios between the agents defined in the different AUML Protocol diagrams, instances of the use case. Note that these rules may be instantaneous rules or tick rules, conditional or unconditional.

```
(tomod USE-CASEi is inc EXTERNAL-AGENTS . inc AGENTS .
including REAL-TIME-AGENTS. including LINKED-BEHAVIORS.
rl [1] : Configuration1 => Configuration2. …
rl [m] : Configuration 2m-1 =><Configuration2m. endtom)
```

Fig. 9. The Timed O.O Module USE-CASEi

Once generated, all USE-CASEi modules are imported in the timed object oriented module RTMAS-FUNCTIONAL-REQUIREMENTS (Fig. 10) which describes all system's functional requirements.

```
(tomod  RTMAS-FUNCTIONAL-REQUIREMENTS  is
including USE-CASE1. …  including USE-CASEm.   endtom)
```

Fig. 10. The Timed OO Module RTMAS-FUNCTIONAL-REQUIREMENTS

The tick rule used to ensure the progress of time in the system is given in Fig. 11, where we have defined the message *Timer* to change the real time agent clock defined by the attribute *Clock* (line [1]). Obviously, this change also depends on the agent's current state: if the agent is in its *wait* state and the Timer has not reached the value zero, the clock is incremented by 1, until that this condition will no longer be valid.

```
crl [tick] :{Timer(TimeOut) < A : RealTimeAgent |
CurrentState : S, Clock : T, PlayRole: Initiator> --[1]
REST:Configuration} => { Timer(TimeOut monus 1)
< A : RealTimeAgent |CurrentState: S, Clock : T plus 1>
REST:Configuration }   in time 1
if (TimeOut > zero)and(S==AgentState(WaitI, ordinary)).
```

Fig. 11. The Tick Rule

6 Case Study: Supply Chain Management (SCM)

The supply chain management has been realized using a multi-agent system [18]. The agent decomposition that we select for this application is as follows (Fig. 12): different types of agents are involved in this application, there are two external agents: (1) the Client who passes, modifies and deletes the orders, (2) the Provider of materials for the

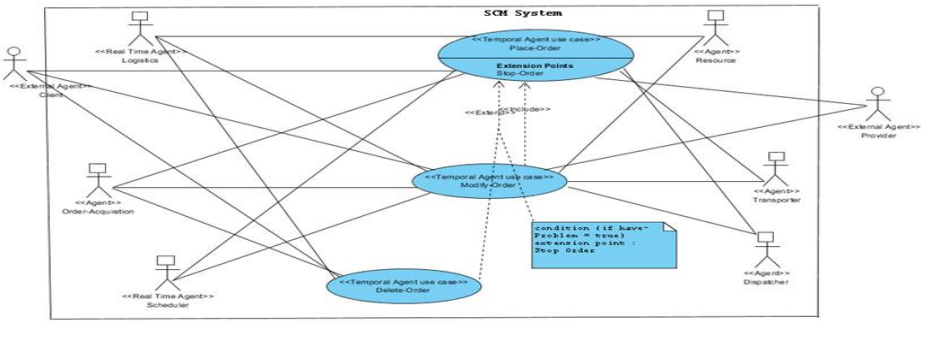

Fig. 12. AUML Use Case Diagram of SCM

realization of products, and six internal agents (Order-Acquisition, Dispatcher, Resource, Transporter, Logistics and Scheduler). Logistics and Scheduler are real time agents. These agents interact with the three temporal agent use cases: Place Order, Modify Order, and Delete Order. The use case Place Order is linked to the use case-Modify Order by the relationship "Include" and to Delete-Order use case by the relationship "Extend".

Fig.13. illustrates the temporal AUML Class diagram of the SCM. This diagram gives a detailed view of agents and their relationships. For example in the Real Time Agent 'Logistics' class, the following six compartments are defined: Role (Logistics), Attributes (Acqlist, MBox), Operations (Request-Plan), Protocols (Create-Order, Modify- Order, and Delete-Order), statecharts (Logistics), Temporal Constraint (Time of Negotiation).

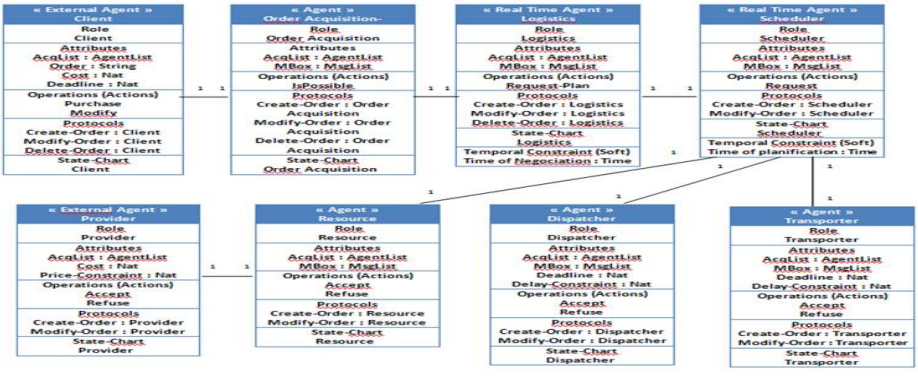

Fig. 13. AUML class diagram of SCM

As example of state-chart of real-time agents, we give in the Fig.14 the internal behavior of the real-time agent Logistics.

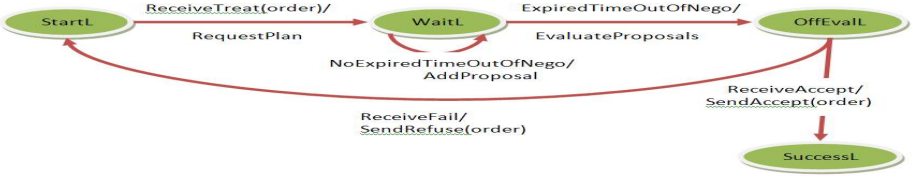

Fig. 14. AUML State-Chart diagram of the real time agent Logistics

The different interaction scenarios that implement the three above use cases are described using AUML protocol diagrams. We only present the protocol diagram of the use case Place-order (Fig.15), where different interaction modes are used (AND, exclusive OR) and the notions of reference and alternate (Alt, Ref).

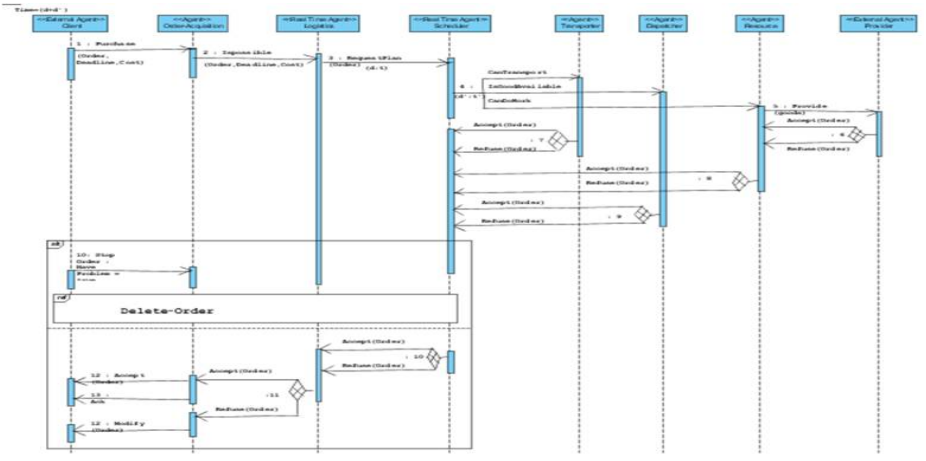

Fig. 15. AUML protocol diagram corresponding to use case Place-Order

6.1 Translation Process Application

The generated description implies the modules STATE, ACTION, CONDITIONS, STATE-OPERATION, MESSAGE-MANAGEMENT, IDENTIFICATION, MESSAGE, MAILBOX, ACQAINTANCE-LIST, EXTERNAL-AGENT-CLASS-AGENT-CLASS, and REAL-TIME- AGENT-CLASS, which remain unchanged with the definition of the other modules: EXTERNAL-AGENTS (Fig. 16), AGENTS (Fig. 17), 'PLACE-ORDER' (Fig. 18), RTMAS-FUCTIONAL-REQUIREMENTS (Fig. 19) ... etc.

```
(omod EXTERNAL-AGENTS is
extending EXTERNAL-AGENT-CLASS. inc STRING. inc NAT.
subclass Client  Provider  <  ExtAgent .
class Client |Order : String, Deadline: Nat. Cost: Nat.
class Provider |PriceConstraint: Nat,Cost : Nat. endom)
```

Fig. 16. The OO Module EXTERNAL-AGENTS

```
(omod AGENTS is extending AGENT-CLASS. inc STRING. Inc
NAT. subclass Transporter Dispatcher < Agent.
class Transporter |DelayConstraint : Nat,Deadline: Nat.
class Dispatcher| DelayConstraint: Nat, Deadline : Nat.
endom)
```

Fig. 17. The OO module AGENTS

```
(tomod PLACE-ORDER is inc EXTERNAL-AGENTS . inc AGENTS .
inc REAL-TIME-AGENTS. inc LINKED-BEHAVIORS.  …  endtom)
```

Fig. 18. The Timed O.O Module PLACE-ORDER

```
(tomod RTMAS-FUNCTIONAL-REQUIREMENTS is inc PLACE-ORDER
inc MODIFY-ORDER . inc DELETE-ORDER .  endtom)
```

Fig. 19. The Timed O.O Module RTMAS-FUNCTIONAL-REQUIREMENS

6.2 Generated Description Validation

The RT-Maude offers a great flexibility in terms of simulation of a specification, in particular, concerning the choice of the initial configuration. This choice plays a primordial role in the validation of the description of a system. Using all the system's description, we can validate a part of the system without involving the rest. In this example of' SCM, we considered : (1) the behavior that starts by passing an order by the external agent *Client* and finishes by satisfying the requirements of the other agents, (2) the decision taken by the client for deleting a passed order, and (3) the incapacity of internal agents to treat a passed order. Fig. 20 illustrates the timed O.O module SUPPLY-CHAIN-MANAGEMENT, which imports the module RTMAS-FUNCTIONAL-REQUIREMENTS and contains an initial configuration. This later describes agents in their initial states with empty mail boxes. Real time agents' clocks are initialized to zero. Two messages are defined *TimerOfNeg*, and *TimerOfSched,* with the event *Event("client", AgentState(StartC),IsInitialized)* that starts the SCM process.

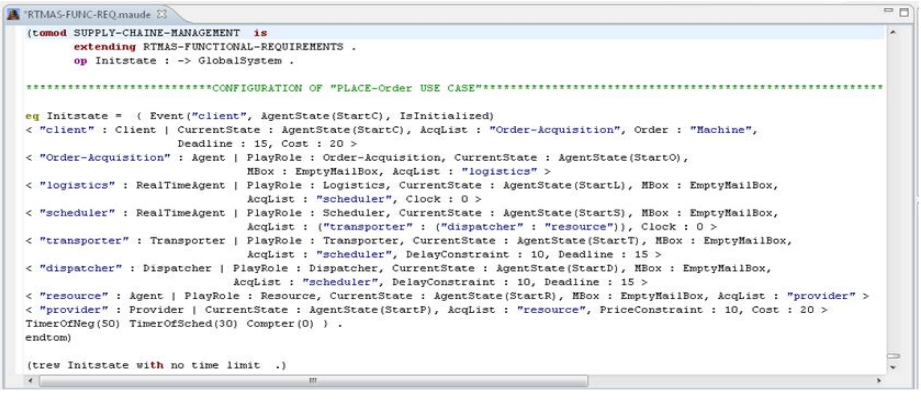

Fig. 20. Initial Configuration

The result of the unlimited rewriting (with no time limit) of such a configuration is illustrated by Fig.21. This result configuration shows the agents in their success states which explains that the constraints imposed by the client have been accepted. Subsequently, the client also passes to its success state.

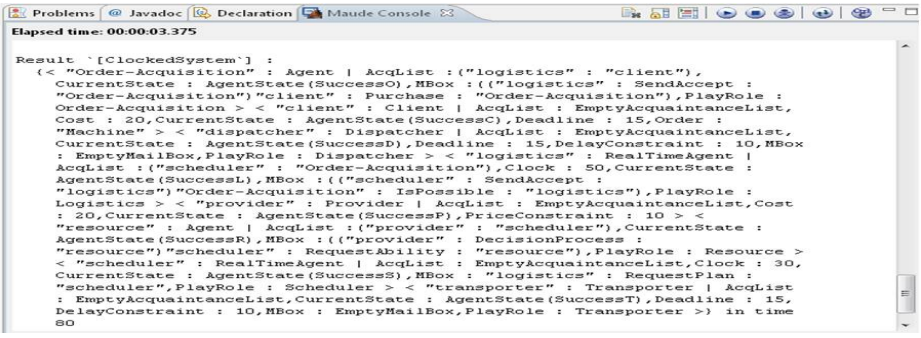

Fig. 21. Result of the unlimited rewriting of the initial configuration

7 Conclusion and Future Work

Using formal notations to specify RT-MAS' requirements makes it possible to produce precise descriptions. This also offers a better support to their verification and validation processes. In this paper, we presented a novel and generic approach supporting the formal description and validation of RT-MAS' functional requirements. The proposed approach allows translating functional aspect (described by extended AUML use case), static aspects (described by extended AUML class diagram), and dynamic aspects (described by AUML protocol diagrams together with AUML state-chart diagrams) of RT-MAS into a RT-Maude formal specification. As future work directions, we plan to extend our approach by integrating possibilities offered by RT-Maude to verify some properties of the specification of RT-MAS' functional requirements.

References

1. Julián, V., Soler, J., Moncho, M.C., Botti, V.: Real-Time Multi-Agent System Development and Implementation (2004)
2. Melián, S.F., Marsá, I., Ukrania, M., Miguel, D.-R., Carmona, A.-L.: Extending the BDI ASDP methodologie for Real Time (2005)
3. Zhang, L.: Development Method for Multi-Agent Real Time Systems. Faculty of Computer Science and Technology Guangdong University of Technology. International Journal of Information Technology 12(6) (2006)
4. Booch, G., Rumbaugh, J., Jacobson, I.: The Unified Modeling Language User Guide. Addison-Wesley, Reading (1999)
5. Bauer, B., Muller, J.P., Odell, J.: An extension of UML by protocols for multiagent interaction. In: International Conference on MultiAgent Systems (ICMAS'00), Boston, Massachussetts, pp. 207–214 (2000)
6. Odell, J., Parunak, H.V.D., Bauer, B.: Extending UML for Agents. In: Wagner, G., Lesperance, Y., Yu, E. (eds.) Proceedings of the Agent-Oriented Information Systems Workshop at the 17th National Conference on Artificial Intelligence, Austin, Texas. ICue Publishing (2000)
7. Huget, M.P.: Extending agent UML protocol diagrams. In: Giunchiglia, F., Odell, J.J., Weiss, G. (eds.) AOSE 2002. LNCS, vol. 2585, pp. 150–161. Springer, Heidelberg (2003)

8. Huget, M.P.: Agent UML class diagrams revisited. Technical Report, Department of Computer Science, University of Liverpool, p. 1–13 (2002)
9. Ferber, J.: Les systèmes Multi-Agents: vers une intelligence collective, Inter edn., Paris, France (1995)
10. Omicini, A.: Soda: Societies and infrastructures in the analysis and design of agent-based systems. In: Ciancarini, P., Wooldridge, M.J. (eds.) AOSE 2000. LNCS, vol. 1957, pp. 185–193. Springer, Heidelberg (2001)
11. Message, http://www.eurescom.de/public/projects/P900-series/p907/
12. Message, Metamodel, http://www.eurescom.de/~public-webspace/P900-series/P907/MetaModel/index.Htm
13. Clavel, M., Duran, F., Eker, S., Lincoln, P., Marti-Oliet, N., Meseguer, J., Talcott, C.: Maude Manual (version 2.2). In: SRI International, Menlo Park, CA 94025, USA (2005)
14. Olveczky, P.C.: Real-Time Maude 2.3 Manual. Department of Informatics, University of Oslo (2007)
15. Heinze, C., Papasimeon, M., Goss, S.: Specifying Agent behaviour with use Case (2000)
16. Papasimeon, M., Heinze, C.: Specifying Requirement in Multi-agent System with use Cases (2000)
17. Bauer, B., Odell, J.: UML 2.0 and Agents: How to Build Agent-based Systems with the New UML Standard (2005)
18. Shen, W., Norrie, D.-H.: Agent-Based Systems for Intelligent Manufacturing: A State-of-the-Art Survey. Knowledge and Information Systems 1, 129–156 (1999)