Formalizing SIMBA RTMAS Models using Real-time Maude

Toufik Marir[1], Farid Mokhati[2] and Hassina Seridi-Bouchelaghem[3]
[1]*Mathematical and Computer Science Department, University of Oum El Bouaghi, Oum El Bouaghi, Algeria*
[2] *Mathematical and Computer Science Department, University of Oum El Bouaghi, LAMIS Laboratory*, Oum El Bouaghi, *Algeria*
[3]*Computer Science Department, LABGED Laboratory, University Of Annaba, Annaba, Algeria*

Keywords: Formal Specification, Real-time Multi-Agent System, Real-time Maude, SIMBA.

Abstract: Multi-agent systems paradigm is the most appropriate one to developing complex systems. Consequently, it is used to develop real-time intelligent systems which are one of complex systems categories. In the literature, several works have been proposed for formalizing many aspects of multi-agent systems. However, the application of formal methods upon real-time multi–agent systems (RTMAS) (where the time is the primordial aspect) stills in the immaturity stage. In this paper, we present the formalization of SIMBA real-time multi-agent systems using Real-Time Maude language as a main stone for formal development of based-SIMBA systems.

1 INTRODUCTION

Development of complex and distributed real-time systems is nowadays a challenge to many researchers. The multi-agent paradigm is presented in several cases as an ideal solution to develop these systems. However, there is no yet a mature methodology to develop real-time multi-agent systems (Zhang, 2006). We think that the remarkable recent advances of formal methods can be a good assistance to develop such systems. In this paper we propose the formalization of SIMBA real-time multi-agent systems using Real-Time Maude.

Several approaches have been proposed to formalize real-time agent using different real-time models like timed automata (Hutzler, Claudel and Wang, 2003; Moscato, et *al*., 2008). According to Ölveczky (2000), we think that logic-based models instead of others models of real-time systems (like timed automata) provides more expressiveness to describe different aspects of RTMAS. Marir, Mokhati and Seridi-Bouchelaghem (2009) proposed the formalization of ARTIS real-time agent using Real-Time Maude. In fact, the social aspect of RTMAS has been omitted in the formalization of ARTIS agent. Hence, we present in this paper an extension of the formalization of ARTIS agent to support the social aspect of RTMAS through the formalization of SIMBA platform.

SIMBA (Multi-Agent Systems Based-On ARTIS) can be considered as an extension of ARTIS real-time agent to support the social ability (Julian, et al., 2002). Thus, SIMBA real-time multi agent system consists of a set of communicating ARTIS real-time agent. Moreover, SIMBA requires the existence of a mediator agent which integrates the DF (Directory Facilitator) and the AMS (Agent Management System) services.

Real-Time Maude is an extension of Maude language to specify and analyzing real-time and hybrid systems (Ölveczky and Meseguer, 2007). Based on rewriting logic, Maude language provides a high expressiveness level to describe and analyzing distributed systems. More details about Maude can be found in (Clavel et al., 2007).

Due to the limited space of this paper, we focus in the following section only on the formalization of SIMBA platform. The ARTIS formal framework on which the formalization of SIMBA has been based is presented briefly. The formalization of ARTIS agent using Real-Time Maude can be found in Marir, Mokhati and Seridi-Bouchelaghem (2009).

2 TRANSLATION PROCESS

This section is devoted to explain the translation process of SIMBA RTMAS description to Real-

Marir T., Mokhati F. and Seridi-Bouchelaghem H..
Formalizing SIMBA RTMAS Models using Real-time Maude.
DOI: 10.5220/0004259904110414
In *Proceedings of the 5th International Conference on Agents and Artificial Intelligence* (ICAART-2013), pages 411-414
ISBN: 978-989-8565-38-9

Time Maude specification. In fact, the generated framework of SIMBA specification is an extension of the formalization of ARTIS agent proposed in (Marir, Mokhati and Seridi-Bouchelaghem, 2009). The formal specification of ARTIS agent using Real-Time Maude (Figure 1) is composed of six functional modules: *Knowledge*, *Blackboard*, *Knowledge-Source*, *MKS* (for Multi-Level-Knowledge-Source), *KS-List* (for Knowledge-Source-List) and *MKS-List* (for Multi-Level-Knowledge-Source-List); an object oriented module (*In-Agent* module) and a timed object-oriented module (*ARTIS-Agent*).

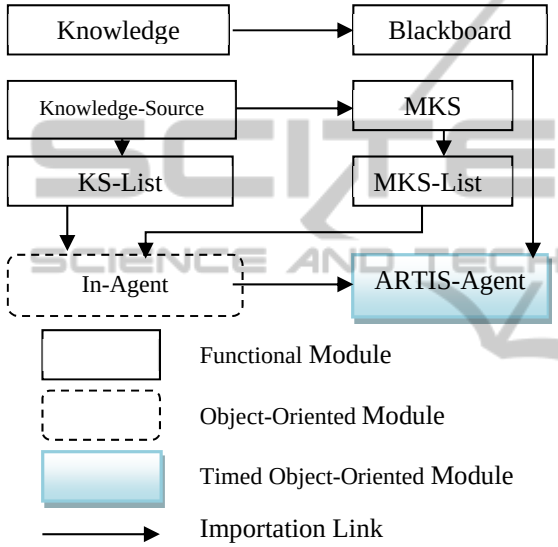

Figure 1: ARTIS framework's architecture.

Because SIMBA architecture is proposed as an extension of ARTIS agent to support the social aspect of multi-agent systems, it is naturally that the formalization of SIMBA platform based on the formalization of ARTIS agent. Consequently, the formalization of the social ability of SIMBA architecture addresses the following purposes:

- The formalization of the message and mailboxes structures to support the communication between agents;
- The formalization of the communication module within ARTIS agent;
- The formalization of the mediator agent;
- The formalization of the interaction between agents.

In order to formalize SIMBA architecture we reused *ARTIS-Agent* module from the formalization of ARTIS agent and we defined several new modules: the *Message* and *Mailbox* functional modules; *Communicating-ARTIS-Agent* and *Mediator-Agent* object-oriented modules; several object oriented modules to specify the different ARTIS agents (noted *ARTIS-Agent-i*) and *SIMBA-RTMAS* timed object-oriented module to define the interactions between agents. Figure 2 gives the architecture of the developed framework.

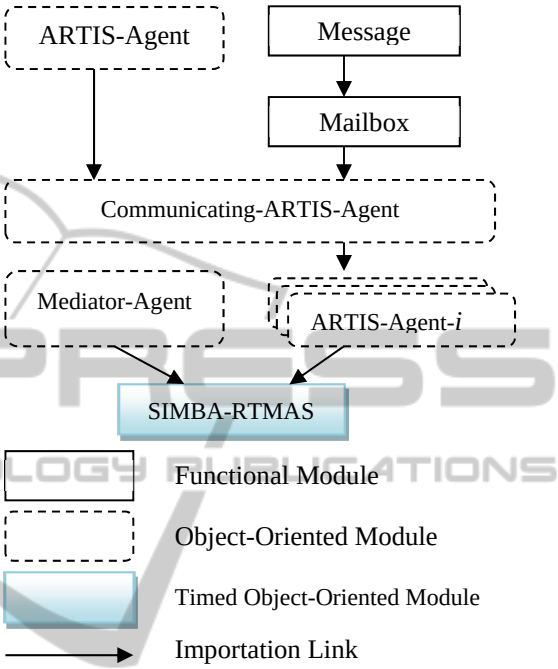

Figure 2: SIMBA framework's architecture.

Due to lack of space only some parts of our framework will be explained.

The two functional modules *Message* and *Mailbox* provide means to communication between agents. The *Message* functional module specifies the message's structure. Thus, the *Mailbox* functional module defines the mailbox's structure as a list of messages.

The mediator agent allows the interaction between agents. It integrates the addresses of the existed agents in the SIMBA architecture with the services offered by each agent. As is presented in Figure 3, the mediator agent is defined in the object-oriented module *Mediator-Agent* as a class with *yellow-pages* attribute. Hence, mediator agent provides the possibility of registration for the new agents (As is presented in Figure 3) and the possibility to unsubscribe for the existing agents.

Communicating-ARTIS-Agent object-oriented module (Figure 4) is an extension of *ARTIS-Agent* object-oriented module with the communication module.

```
(omod Mediator-Agent is

   ***( The mediator-agent class )***
   class Mediator-Agent | Yellow-
     Pages : Agent-Service-List .

   **********(The registration of the
     new agent in the Mediator-
     Agent's base )******************

   rl [Registration] :
      < Agent: Mediator-Agent |
        Yellow-Pages : Agent-
        Service-List >
      Inscribe(Agent-Service)
      =>
      < Agent: Mediator-Agent |
        Yellow-Pages : Insert-Agent-
        Service(Agent-Service,
        Agent-Service-List) > .
endom)
```

Figure 3: Some parts of Mediator-Agent object-oriented module.

In fact, the communication module of ARTIS agent can be defined by two mailboxes inserted in the definition of ARTIS agent class. One of these mailboxes is used to store the received messages and the other is used to store the message before sending. Obviously, a *Communicating-ARTIS-Agent* class includes blackboard structure and its own timer. Known that the tick rule does not support the concurrency rewriting, we should define *only* one tick rule in *SIMBA-RTMAS* module which used to progress a universal time in uniform way.

In order to ensure the concurrency execution of ARTIS agents' clock, we rewrite all the clock unit of ARTIS agent using the same time unit (called *Universal-Time-Unit*). Thus, an ARTIS agent (A) clock should be rewritten according to this *Universal-Time-Unit* using *Clock-Base* parameter (for example, *Timer (A) = Universal-Time / Clock-Base (A)*). The *Clock-Base* is defined as an attribute of *Communicating-ARTIS-Agent* class. Despite the progression of the universal time, an ARTIS agent does not change its clock value only one time each its *Clock-Base* value of *Universal-Time-Unit*. As is presented in the rewriting rule (Figure 4), the ARTIS agent should update the validating time of the messages and the knowledge in each own passed time unit.

The behaviour of the communication module is specified using two rewriting rules (one for receiving messages case and the other for sending messages case) in the *Communicating-ARTIS-Agent* object-oriented module.

```
(omod Communicating-ARTIS-Agent is
   protecting ARTIS-AGENT .
   protecting MAILBOX .

   ********( The Communicating-ARTIS-
     Agent class definition )********
   class Communicating-ARTIS-Agent |
     Received-Mailbox : Mailbox,
     Sending-Mailbox : Mailbox,
     Blackboard-Content : Blackboard,
     Clock-Base-Is : Nat, Timer-Is :
     Time .

   *********( The rule that specifies
     the progression time of an ARTIS
     agent ) ************************

   crl [Progression-Time-Of-ARTIS] :

     < Agent1 : Communicating-ARTIS-
       Agent | Received-Mailbox :
       Received-Mails, Sending-
       Mailbox : Sending-Mails,
       Blackboard-Content :
       Blackboard1, Clock-Base-Is :
       Clock-B1, Timer-Is : T >
     Universal-Time-Is(Global-T)
              =>
     < Agent1 : Communicating-ARTIS-
       Agent | Received-Mailbox :
       Updating-Time(Received-
       Mails), Sending-Mailbox :
       Updating-Time(Sending-
       Mails), Blackboard-Content :
       Updating-Time(Blackboard1),
       Clock-Base-Is : Clock-B1,
       Timer-Is : T plus 1>
     if (Global-T rem Clock-B == 0).

   ***************( The ARTIS agent's
     communication module behavior
     specification ) ***************
   ...
endom)
```

Figure 4: Some parts of *Communicating-ARTIS-Agent* module.

Known that the communication in SIMBA architecture is not critical task, the communication module can execute only if the ARTIS agent is not in critical phase. As is presented in Figure 5, an ARTIS agent in not-critical phase can extract a message from its sending mailbox (if it is not empty) to send it (*Sending-Mail* rule) or extract a message from its received mailbox (if it is not empty) to process it (*Receiving-Mail* rule). For legibility reason, only the necessary attributes of

Communicating-ARTIS-Agent class are presented in the two rewriting rules.

```
*****( Sending Mail case rule )*****

crl [Sending-Mail] :
   <Agent1: Communicating-ARTIS-Agent
     | Sending-Mailbox: Mails >
            =>
   Mail(Head-Mailbox(Mails))
   <   Agent1:   Communicating-ARTIS-
     Agent  |  Sending-Mailbox:  Tail-
     Mailbox(Mails) >
   if   (   not   Is-In-Critical-Phase
     (Agent1)   and   not   Is-Empty-
     Mailbox(Mails)) .

*****( Receiving Mail case rule )***

crl [Receiving-Mail] :
   <   Agent1:   Communicating-ARTIS-
     Agent | Received-Mailbox: Mails>
                  =>
   Received-Mail(Head-Mailbox(Mails))
   <   Agent1:  Communicating-ARTIS-
     Agent | Received-Mailbox: Tail-
     Mailbox(Mails)>
   if   (   not   Is-In-Critical-Phase
     (Agent1)   and   not   Is-Empty-
     Mailbox(Mails)) .
```

Figure 5: The communication module behaviour specified by rewriting rules.

The *Communicating-ARTIS-Agent* module specifies only the kernel of communicating ARTIS agent. The specific behaviour of each ARTIS agent should be specified in separated modules (called *ARTIS-Agent-i* object-oriented module). By specific behaviour, we intend the pre-condition and the post-condition of ARTIS agents' execution.

SIMBA-RTMAS timed object-oriented module is used to specify the global behaviour of the real-time multi-agent system. By the global behaviour of the real-time multi-agent system we intend the interaction between ARTIS agents to achieve their purposes.

Using our framework to specify based SIMBA software necessities the customizing of some modules to the specific needs of user. Indeed, the users of our framework should specify the specific behaviours of each ARTIS agent in *ARTIS-Agent-i* modules and the global behaviour in the *SIMBA-RTMAS* module.

Real-Time Maude provides several analysis tools which can be applied on our framework to detect possible errors.

3 CONCLUSIONS

The development of real-time multi-agent system is a difficult activity in which the consequences of errors can be catastrophic. Indeed, the uses of formal methods represent the best solution for this problem. Obviously, the formalization of RTMAS is the first step to apply the different formal techniques such as model checking or simulation. In this paper we extent the formalization of ARTIS real-time agent to support the social aspect of RTMAS. Our approach is based on Real-Time Maude language which is characterized by the expressiveness of description and the variety of its techniques of verification and validation. In a future work, we will study the different properties to verify using Real-Time Maude's model checker.

REFERENCES

Clavel, M., Durán, F., Eker, E., Lincoln, P., Martí-Oliet, N., Meseguer, J., and Talcott, C., L., (Eds.), 2007, All About Maude - A High-Performance Logical Framework, How to Specify, Program and Verify Systems in Rewriting Logic. *Lecture Notes in Computer Science 4350.* Springer.

Julian, V., Carrascosa, C., Robello, M., Soler, J., and Botti. V., 2002., SIMBA: An Approach For Real Time Multi Agents Systems, In *Proc. of V Conferncia Catalana d'Intelligncia Artificial, Castell. Springer-Verlag*e.

Hutzler, G., Claudel, H., Wang, D., Y., 2004, Designing Real-Time Multi-Agent Systems Using Timed Automata, In *Ghidiniet, C., al (Eds) EUMAS'04.*

Marir, T., Mokhati, F., and Seridi-Bouchelaghem, H., 2009, Formalizing ARTIS Agent Model using RT-Maude, In L. Braubach and al. (Eds.): *MATES 2009*, LNAI 5774, pp. 226–231.

Moscato F., Venticinque, S., Aversa, R., and Di Martino, B., 2008, Formal Modeling and Verification of Real-Time Multi-Agent Systems: The REMM Framework, In *Badica C., et al. (Eds.): Intel. Distributed Comput., Systems & Appl., SCI 162*, pp. 187–196.

Ölveczky. P., C., 2000, Specifying And Analysis of Real-Time And Hybrid Systems In Rewriting Logic. Dr. Scient. Thesis, Department of Computer Science, University of Bergen.

Ölveczky, P., Meseguer, J., 2007. Semantics and pragmatics of Real-Time Maude, In *Higher-Order and Symbolic Computation*, Volume 20 Issue 1-2, June 2007, Pages 161 – 196.

Zhang, L., 2006, Development Method for Multi-Agent Real-Time Systems, In *International Journal Of Information Technology*, Vol. 12, No. 5, 19-28.