BECAMEDA: A Customizable Method to Specify and Verify the Behavior of Multi-agent Systems

Abdelhay Haqiq(✉) and Bouchaib Bounabat

ENSIAS, Mohammed V University, Rabat, Morocco
{abdelhay.haqiq,b.bounabat}@um5s.net.ma

Abstract. Multi-Agent paradigm offers a viable solution to the increasing needs for smart and crisis system that reacts accordingly to the environment changes. The researches have largely focused on studying the development of the various approaches that deals with designing and implementing the multi-agent system. However, there is a lack in approaches that treat in depth the specification aspect of the Multi-Agent systems engineering process, which is important to better define and describe the behavior of the agents. This paper is interested in studying the specification and the verification of Multi-Agent behavior, it proposes in consequence an effective method called BECAMEDA. It is based on an iterative process that is useful to understand the system starting from goals identification that the system expects to achieve through the formal verification of the system properties. The method is illustrated by an example of a Crisis Management system.

Keywords: Multi-agent · Specification · Verification · Behavioral model
Model checking

1 Introduction

The most challenging thing in software engineering for Multi-Agent is to ensure that the requirement of the system is well identified and verified [1, 2], especially when the system is complex and appeals with many stakeholders. In this paper, we are interested to study the Multi-Agent system to cope with the dynamic, distributed and cooperative systems. A multi-agent system is a system made up of several agents which are in interaction among them, this interaction is very important to define the overall behavior of the system. The behavior of the agent is defined to be a set of properties that an agent outlines to give responses to its environment [3]. Basically, the agent reacts to events based on the received decisions and gives responses in the form of actions according to the experienced situation.

To specify MAS, it is important to have processes and methods that will help engineers to understand how to build up customized system [4]. In this paper, we are interested of studying the specification of five essential properties that the agents are supposed to exhibit: reactivity, adaptability, conflict management, and reusability. The reactivity refers to the ability to respond to the environment, the adaptability is the capacity to cope with changes and disruptions of the environment, the conflict management is the ability to manage resource sharing, and the reusability is the capability to reuse agent

M. Ben Ahmed and A. A. Boudhir (Eds.): SCAMS 2017, LNNS 37, pp. 74–86, 2018.
https://doi.org/10.1007/978-3-319-74500-8_7

components. In literature, there are approaches [10–14] that study some of these properties, but there is still a lack of an all-in-one approach that considers all of the mentioned properties.

To deal with the complexity of building up a multi-agent system, this paper proposes a method that helps to make the specification and the verification of such a system to avoid errors happening within the implementation phase. Since MAS can be applied in different domains and systems, this paper proposes to have demonstration of this method through a Crisis Management system case study.

Thus, this paper intends to present the difficulty of how to define the specification of Crisis Management system for both individual and collective behavior. We propose a method called BECAMEDA (Behavioral spEcification and verifiCAtion of RMAS based on E-MDRA) for the specification and the verification of Reactive Multi-Agent System based on the E-MDRA (Extended Multi-Decisional Reactive Agent) model. The E-MDRA is an extension of The MDRA, which is developed by [5], The MDRA is being deployed in various domains such as wireless sensor networks [6], mobile systems [7] and organizational systems [8], this model helps on modeling reactive agents by putting forward decisional aspect facilitating the specification of the agent according to their behavior and their intelligence. Throughout this paper, the Crisis Management system is used as an example to demonstrate our approach [9]. The overall goals of the Crisis Management System (CMS) are: facilitating the rescue mission of the police by offering detailed information about the location of the crash; managing the dispatch of ambulances or other alternate emergency vehicles to transport victims from the crisis scene to hospitals; facilitating the first-aid missions by providing relevant medical history of identified victims to the first-aid workers by querying databases of local hospitals.

The rest of the paper is organized as follows. Section 2 presents a related work. Section 3 presents a background related to the Extended Multi Decisional Reactive Agent. Section 4 describes our approach through an example of a Crisis Management System. Section 5 concludes the paper and draws future perspectives.

2 Related Work

Many approaches have been proposed for the specification and the verification of MAS. [10] presents a formal method for specifying and verifying a Multi-Agent system based on design patterns, refinement, and event-B. The objective of the method is to satisfy the global properties of the system based on the agent's local behavior. [11] proposes an approach to specify by refinement in event-B the collaboration between agents in a critical system where the consideration of the fault tolerance property is essential. The authors define the SMA through four applets (A, M, E, R). A represents a collection of different classes of agents. M represents a middleware system that has the ability to ensure the communication flow between agents and resolve the communication disconnect issues, E represents a collection of system events, R represents a set of dynamic relationships between agents allowing making connections between active agents belonging either to the same classes or to different classes. [12] proposes High Level Multi-Agent Petri-Net (HMAP) to describe, model and analyze the dynamic behavior

of MAS based on the Petri Net. HMAP is based on a formal logical method describing the behavior of the agent in nine sequential steps. [13] proposes an approach called ForMAAD (Formal Method for Agent-based Application Design) to specify the agent's behavior at the individual and the collective levels. The behavior is specified through a formal language called Temporal Z that integrates the first-order temporal logic with the Z notation. The approach proposes to divide the refinement process into four levels: cooperation, organizational, collective behavior and individual behavior. [14] proposes a translation of the AUML (Agent UML) diagrams; which is an extension of the UML language to model Multi-Agent system; into the RT-Maude formal language. RT-Maude supports the specification and the analysis of real-time systems, and more specifically object-oriented real-time systems. It is based on the programming environment based on the rewriting logic.

The Table 1 presents a comparison between the different approaches according to the reactivity, adaptability, conflict management, and reusability properties. We note that there is no specific approach that considers all the properties mentioned. Besides of this, there is a lack of a real process that helps engineer to model the specification phase, which is the most important phase on the software engineer process.

Table 1. Comparison of approaches to specify MAS.

	Reactivity	Adaptability	Conflict management	Reusability	Method support
[10]	**	–	*	*	**
[11]	**	*	*	–	–
[12]	**	*	**	*	–
[13]	**	–	*	*	**
[14]	**	**	–	*	–
[15]	***	**	**	*	–

Caption: ***: strongly; **: moderately; *: Weakly; –: not supported or not described

In our previous works [15], we have proposed an extension of MDRA model called E-MDRA to deal with the properties mentioned. However, we are still in need to have a method that helps engineers to correctly identify and define the specification. Thus, this paper focuses on the proposed method based on E-MDRA. The next section gives an overview of the E-MDRA specification model.

3 E-MDRA Model Overview

3.1 E-DRA

The E-DRA model is based on an external specification & an internal specification [15]. The external specification denotes a set of information sent to or received from the environment. It is characterized by:

- A: Set of actions exerted on the agent, each action represents a possible operation to be performed on this agent

- D: Set of decisions generated by the agent to provide solutions concerning the system behavior. A decision d is characterized by its decision horizon DurDec(d), that indicates the time during which this decision remains valid
- S: Set of Signal received by the agent. The signal represents the acknowledgement response to confirm the execution of a decision
- E′: Set of external states delivered by the agent. Each external states represents the object state emitted to the environment

The internal specification represents a set of information generated and exchanged inside the agent. It is characterized by:

- E: Set of agent's internal states. Each one indicates the current state of the agent. An agent may exhibits parallel operations
- O: Set of agent's internal objectives. Each internal objective denotes the expected state after the execution of a decision
- O′: Set of agent's external objectives, which can be achieved. These objectives represent the agent interpretation of each action, and define therefore its different behaviors

From a dynamic perspective, these sets determine the events received from the environment (A, S), events sent to the environment (D, E) and the internal events (E, O, O′).

3.2 E-MDRA

The organization of Extended Multi Decisional Reactive (E-MDRA) is defined by a set of agents connected together with communication interfaces, thus forming a hierarchical structure based on two-level tree: an E-DRA Supervisor (E-DRAS) and two or several sub-agent components (E-MDRASi). The connection between supervisor and its sub-agents is realized through two communication interfaces: Decisional Interface (DI) and Signaling Interface (SI). Such a system interacts with its environment with Actions exerted by the environment and External States emitted to the environment.

The description of a MDRA is defined as a quadruplet S: < E-DRAS, DI, SI, S-MDRA >, with:

- E-DRAS: represents the E-DRA type that supervises S.
- DI: Decisional Interface of S, implements a translation function of a decision into several parallel actions, each of these actions belongs to a low-level sub-agent.
- SI: Signaling Interface of S, implements a translation functions of several external states into one and only signal.
- S-MDRA: characterizes a set of 2 or more E-MDRA components.

4 BECAMEDA Method

BECAMEDA (Behavioral spEcification and verifiCAtion of RMAS based on E-MDRA) method provides support for the specification and verification of multi-agent with the involvement of the reactivity, the adaptability, the conflicts management and the reusability properties. It captures the requirements and goals of the system, identifies

the goal's actions, determines the system's organizational structure and models the internal and external behavioral specification of the agents.

BECAMEDA is made up of two layers: specification layer and verification layer. The specification layer is consisted of three phases: identification phase, definition phase and modeling phase. The identification phase is the phase where the system identifies its goals. A goal represents a desirable objective of the system. Goals are organized as a hierarchical structure made up through the refinement of higher-level goals towards lower-level goals. The lower-level goal is achieved through actions. The definition phase consists of two steps: the first step represents the agents and their organizational structure which is also classified into levels. Whereas, the second step determines the actions to be assigned to agents under each level. During this phase, the actions are a priori assigned to agents of level 1, then, according to the modeling phase, they are assigned to the following level. Basically, the modeling phase is made up through three steps for each agent under a level i greater than or equal 1. The First step, models the internal behavior actions of level i. the Second step, assigns actions to agents of the following level (level i + 1), and finally, the third step, models the external behavior actions of level i.

After having establishing the models of the agent for both internal and external behavior, the verification layer indicates the system properties to be verified, and performs afterwards a formal verification with the use of model-checking technique. Figure 1 illustrates the BECAMEDA phases.

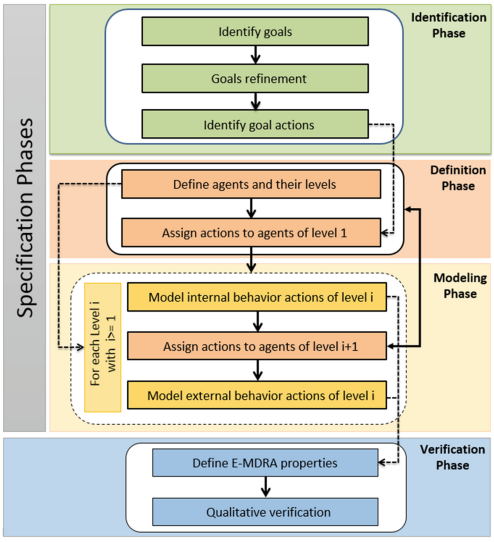

Fig. 1. BECAMEDA phases.

4.1 Identification Phase

The identification phase captures the system requirements and identifies the goals of the system. A goal defines the global objectives that the system desires to fulfill. This phase

is made up following three steps: identifying goals, refining goals and identify goal actions. The two first goals are represented with two models: the goals model and the refined goals model. The goal model facilitates the understanding of a system with the generation of a set of abstract goals that offer a high-level understanding of the defined objective of the system, as for the refined goals model, it identifies the concrete goals that highlight the goal activities. The goal-actions model identifies the actions that realize the concrete goal. The entities forming this phase are built up following the meta-model of Fig. 2 where is defined the concept of goal, action and their different relationships.

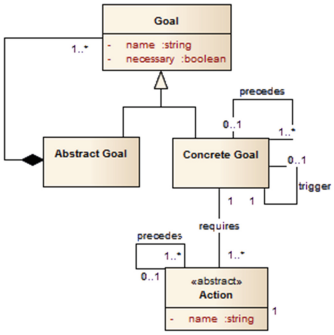

Fig. 2. Identification phase metamodel.

In the following, we present the BECAMEDA method throughout an example of a Crisis Management system.

Identify goals. In this step, we represent the system requirements as a set of structured goals consisted of sub-goals. The goals called "abstract goals" express the high-level objective of the system; they are connected by AND/OR/XOR logic gates. The AND indicates that all of the sub-goals must be completed to satisfy the parent goal, the OR indicates that there is an alternative way to represent the sub-goals, and the XOR indicates that the goal parent can be satisfied with one of its sub-goals.

Goal refinement. We decompose in this step the abstract goal into concrete goals to capture the dynamic aspects of the goals model and defines each goal through "precede" and "trigger" attributes. The "precedes" determines that the goal X must be completed before the goal Y is pursued. Whereas "triggers", means the instantiation of a new goal when an event occurs while another goal is still in activity.

The Fig. 3 presents the step 1 and 2 of the identification phase. The Goal "capture info Crisis (Goal 1)" depends on the achievement of three concrete goals: "receive information from witnesses (Goal 1.1)", "receive information from video surveillance (Goal 1.2)" or "receive information from the phone company (Goal 1.3)". We note that the three sub-goals are successive goals, since they are defined by the key word "preceded" and are alternative as well. Indeed, when a crisis event is reported, the first step to do is to get the maximum information from witnesses, in case of doubt, it is possibility to verify the statements of witnesses via either the surveillance system, if

installed on the incident scene, or by checking the eligibility of witnesses by contacting the phone company. For the goal 2.1, it depends on the achievement of several sub-goals. Basically, depending on the context of the environment, we note different missions: "Establish communication with IR (Internal Resource) (Goal 2.1.1)" mission, "IR confirms mission (Goal 2.1.2)" mission and "Select IR (Goal 2.1.3)" mission.

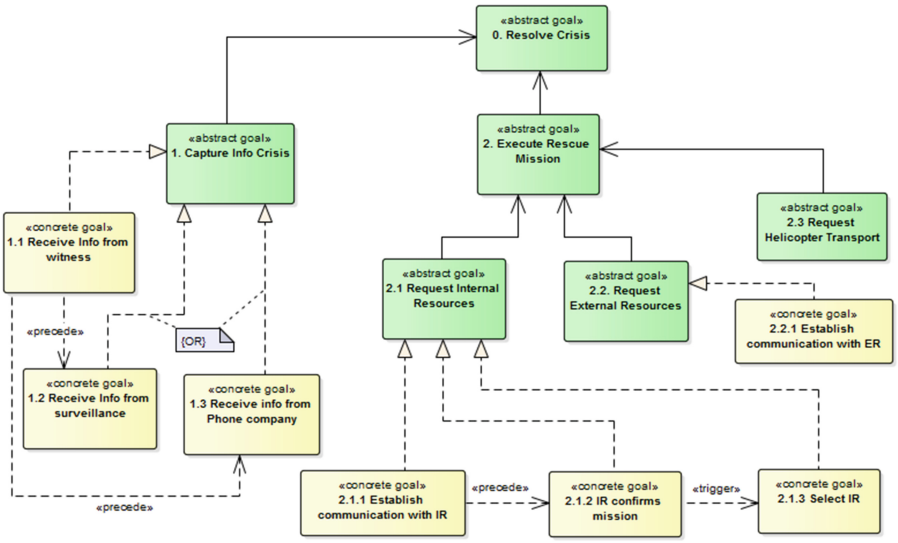

Fig. 3. Abstract & concrete goals crisis system.

Identify goal actions. In this step, we select the actions to carry out to satisfy and realize a specific goal. The figure below shows the model that associates the concrete goals to actions (Fig. 4).

Fig. 4. Goal actions model.

4.2 Definition Phase

The definition phase represents the agents and their organizational structure classified with levels, it also presents the actions to be assigned to agents under each level. This phase is intrinsically dependent on the modeling phase, since there is a back-and-forth

process between the two phases. This process helps at recognizing actions associated to each agent level. The entities of this phase are described in the meta-model of Fig. 5.

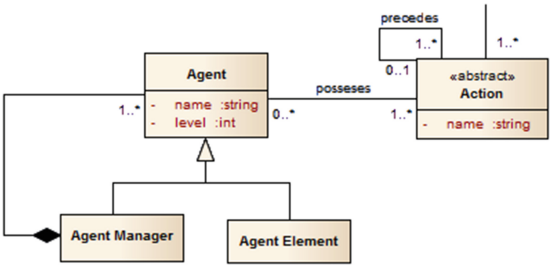

Fig. 5. Definition phase metamodel.

Define agents and their level. In this step, agents are defined and are organized under the specified hierarchical structure of the expected system. We note two kinds of agents: Agent Manager and Elementary Agent. The Agent Manager is the top-level agent who is responsible of taking decisions and maintaining the relationships between agents. Whereas, the Elementary Agent is the lower-level agent who interacts directly with resources. In some cases, an agent manager may supervise other managers and elementary agents.

Referring to the example mentioned above. To initiate the crisis management process, we need different agents: Coordinator agent, Surveillance system, Phone Company, Internal resource manager that supervises First Aid worker and the super Observer. The Coordinator and Internal resource manager represent the "agent manager" that coordinates the sub-agents. These agents denote the "Elementary Agent" since they are directly interacting with the resources to get information (Fig. 6).

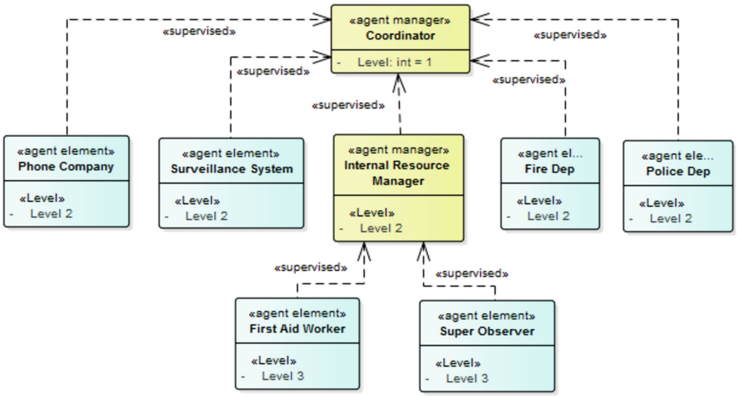

Fig. 6. Agent structure.

Assign actions to agents. In this step, we note the different actions that an agent will need to perform to accomplish its goal. An action is assigned to an agent if only it meets the following rules:

- If an action belongs to a level, then there has to be an agent associated to it
- An agent cannot comply with two actions of different levels

Figure 7 displays the assigning of the actions to agents of level 2 according to the CMS case study. The agent manger "Internal Resource Manager" possess two actions: "Mission analysis" and "Assign Internal Resource". Indeed, it has as role to evaluate the crisis, and then sends the appropriate resource to scene. For the other agents, there are elementary agent that possess each one of them a unique action to be accomplished.

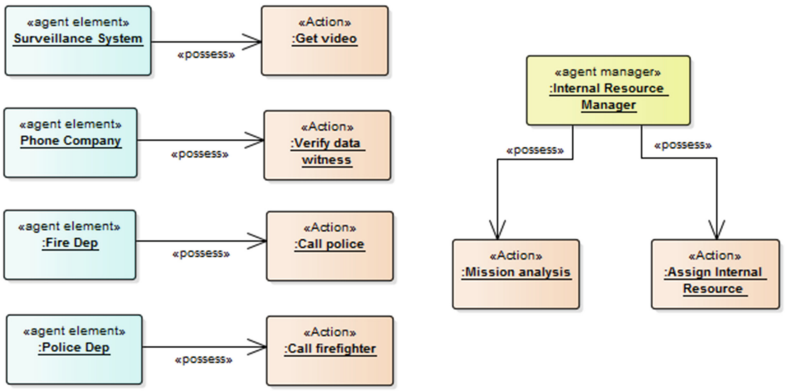

Fig. 7. Assign actions to agents of level 2.

4.3 Modeling Phase

The modeling phase models the high-level description of the internal behavior of the agent's action as well as the external behavior that defines the interaction between agents. There is an iterative process of three steps to model the behavior for each level i (where i >= 1) identified in the definition phase:

1. Model the internal behavior actions of level i
2. Assign actions to agents of level i + 1
3. Model the external behavior actions of level i

Model the internal behavior actions. Modeling the internal behaviour is mainly based on the representation of the decisions and states in terms of transition state diagram, which is based on the profile that we have proposed in [16]. Let us take the example of "Assign Internal Resource" action described in Fig. 8.

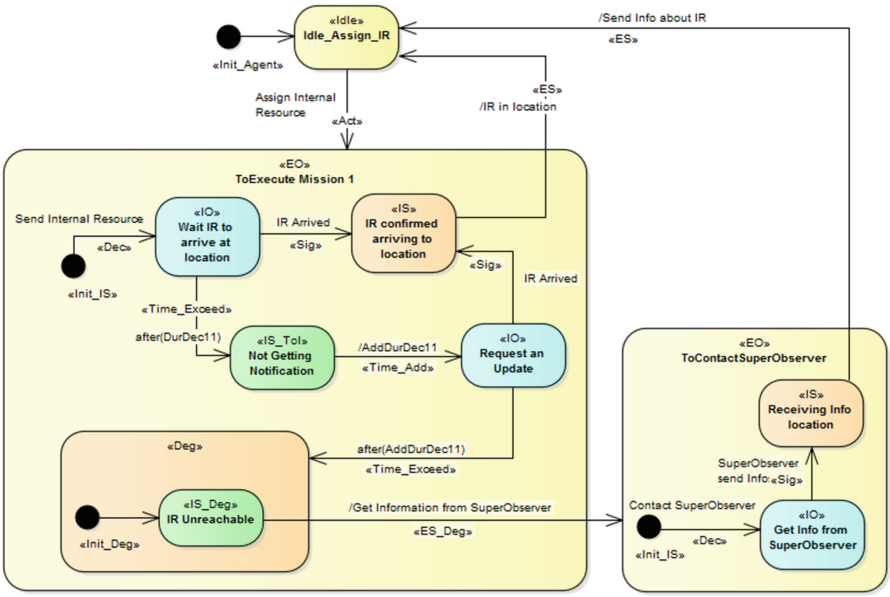

Fig. 8. Internal behavior.

Once the agent receives action "Assign Internal Resource", he makes the decision to "send the resource to the scene crisis". If he does not get any notification form IR (Internal Resource), he adds an extra time waiting for any information, if after that time he does not receive anything, he changes his behavior by contacting the super Observer agent. When all information gathered, he sends a report about IR.

Model the external behavior actions. In the external behavior, we describe the interaction between an agent manager and its sub-agents across the communication interface. This interaction is represented by the activity diagram, which is designed to describe the organizational behavior. The Fig. 9 describes the interaction between the internal resource manager and its sub-agents: the first Aid worker and the Super Observer agent.

4.4 Verification Phase

In the verification phase, we verify the behavioral properties of the system throughout the model checking technique. The model checking technique starts with making a comparison between two descriptions of a system behavior, one is considered as the requirement and the other is considered as the actual design. Then it verifies the system by exploring the state space to determine if the system has satisfied a series of properties. This model provides a counter-example in case the model does not meet the specification [17].

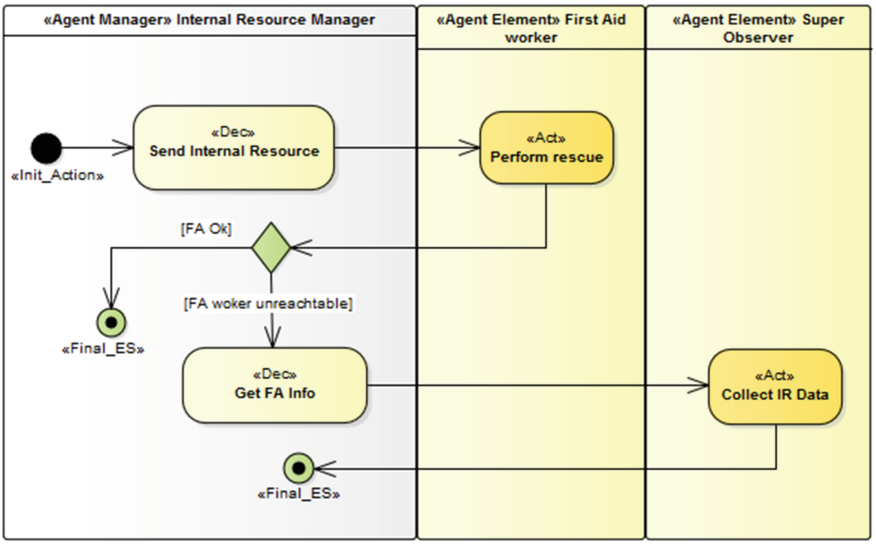

Fig. 9. External behavior.

The BECAMEDA's verification phase proposes to translate the models designed in the modeling phase to the SMV model checker [18, 19], the system's properties is then expressed with temporal logic formulas.

We have used the gNuSMV tool [20] to verify the temporal properties expressed in CTL (Computational Tree Logic) formulas [21]. The Fig. 10 presents an example of a formal verification of the following system properties:

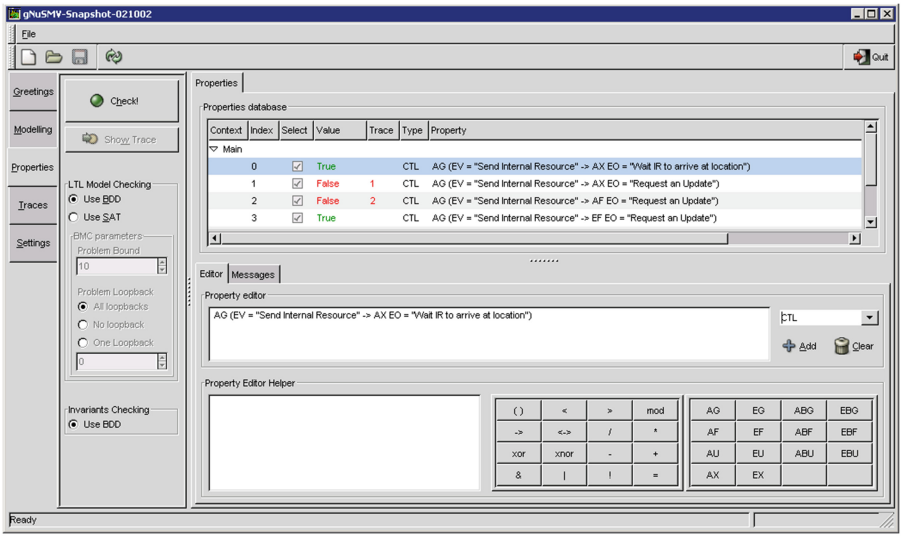

Fig. 10. Properties expressed in CTL.

SPEC AG ((EV = “Send Internal Resource”) - > AX EO = “Wait IR to arrive at location”)

When the agent takes the decision “Send Internal Resource”, he will be in the state to wait for internal resource to arrive at location.

SPEC AG ((EV = “Send Internal Resource”) - > AX EO = “Request an Update”)

When the agent takes the decision “Send Internal Resource”, he will be in the state to request an update

SPEC AG ((EV = “Send Internal Resource”) - > AF EO = “Request an Update”)

When the agent takes the decision “Send Internal Resource”, he will always in the future be in the state to request an update

SPEC AG ((EV = “Send Internal Resource”) - > EF EO = “Request an Update”)

When the agent takes the decision “Send Internal Resource”, he will in some point in the future be in the state to request an update.

5 Conclusion

In this paper, we have introduced a method called BECAMEDA for the specification and the verification of Multi-Agent System. This method is based on different processes that help understanding the system starting from goals identification that captures the system needs, and ending with a formal verification of the system properties. The process specifies four phases: identification, definition, modeling and verification. This approach uses model-checking technique to perform formal verification of the agents’ behavior.

In order to illustrate the specification and the verification aspect of the method, we have used a Crisis Management System, the case study was kept simple to make the concepts of the method clear enough. Thus, the method has been effectively applied to a realistic example, which allows understanding more the system and being able to prove its accuracy. For more details about the verification phase, the reader can refer to our previous works [18, 19].

As future work, we are currently investigating to combine the BECAMEDA method with an existing approach that considers in depth the designing phase. This will be an important extension of the BECAMDA method to build up a complete method that deals with the different MAS engineering phases.

References

1. Mattei, S., Bisgambiglia, P.A., Delhom, M., Vittori, E.: Towards discrete event multi agent platform specification. In: Third International Conference on Computational Logics, Algebras, Programming, Tools, and Benchmarking, pp. 14–21 (2012)
2. Chuanjun, R., Hongbing, H., Shiyao, J.: Specification of agent in complex adaptive system. In: Computer Science and Computational Technology, ISCSCT 2008, vol. 2, pp. 210–216 (2008)
3. Sharma, M., Firdaus, M., Chatterjee, R.K., Sarkar, A.: Constraint specification in multi-agent system. In: Region 10 Conference (TENCON), pp. 2404–2409. IEEE (2016)

4. Subburajand, V.H., Urban, J.E.: Issues and challenges in building a framework for reactive agent systems. In: Complex, Intelligent and Software Intensive Systems (CISIS), pp. 600–605. IEEE (2010)
5. Bounabat, B., Romadi, R., Labhalla, S.: Designing multiagent reactive systems: a specification method based on reactive decisional agents. In: Pacific Rim International Workshop on Multi-Agents. LNAI, vol. 1733, pp. 197–210. Springer, Heidelberg (1999)
6. Aaroud, A., Labhalla, S.E., Bounabat, B.: Modelling the handover function of global system for mobile communication. Int. J. Model. Simul. **25**(2), 99–105 (2005)
7. Romadi, R., Berbia, H., Bounabat, B.: Wireless sensor network simulation of the energy consumption by a multi-agents system. J. Theor. Appl. Inf. Technol. **25**(1), 50–56 (2011)
8. Berrada, M.: Qualitative verification of multi-agents reactive decisional system using business process modeling notation. In: Intelligent Agent Technology, pp. 747–751. IEEE (2006)
9. Kienzle, J., Guelfi, N., Mustafiz, S.: Crisis management systems: a case study for aspect-oriented modeling. In: Transactions on Aspect-Oriented Software Development VII, pp. 1–22. Springer, Heidelberg (2010)
10. Graja, Z., Migeon, F., Maurel, C., Gleizes, M.P., Kacem, A.H.: A stepwise refinement based development of self-organizing multi-agent systems: application to the foraging ants. In: International Workshop on Engineering Multi-Agent Systems, pp. 40–57. Springer, Heidelberg (2014)
11. Pereverzeva, I., Troubitsyna, E., Laibinis, L.: Formal development of critical multi-agent systems: a refinement approach. In: Ninth European Dependable Computing Conference, pp. 156–161 (2012)
12. Chatterjee, R.K., Neha, N., Sarkar, A.: Behavioral modeling of multi agent system: high level petri net based approach. Int. J. Agent Technol. Syst. **7**(1), 55–78 (2015)
13. Hadj-Kacem, A., Regayeg, A., Jmaiel, M.: ForMAAD: a formal method for agent-based application design. Int. J. Web Intell. Agent Syst. **5**(4), 435–454 (2007)
14. Laouadi, M.A., Mokhati, F., Seridi-Bouchelaghem, H.: A novel organizational model for real time mas: towards a formal specification. In: Intelligent Systems for Science and Information, pp. 171–180. Springer International Publishing, Cham (2014)
15. Haqiq, A., Bounabat, B.: An extended approach for the behavioral and temporal constraints specification of reactive agent. In: 15th International Conference on Intelligent Systems Design and Applications, pp. 329–334. IEEE (2015)
16. Haqiq, A., Bounabat, B.: UML profile for modeling multi decisional reactive agent system. J. Lect. Notes Softw. Eng. **1**(3), 224 (2013). ISSN:2301-3559
17. Clarke, E.M.: The birth of model checking. In: 25 Years of Model Checking, pp. 1–26. Springer, Heidelberg (2008)
18. Haqiq, A., Bounabat, B.: Model checking of multi decisional reactive agent system. In: 9th International Conference on Intelligent Systems: Theories and Application, Rabat, Morocco, vol. 1, pp. 133–140 (2014)
19. Haqiq, A., Bounabat, B.: Verification of multi decisional reactive agent using SMV model checker. In: 8th IEEE International Design and Test Symposium, Marrakesh, Morocco, pp. 1–6 (2013)
20. Cimatti, A., Clarke, E., Giunchiglia, E., Giunchiglia, F., Pistore, M., Roveri, M., Sebastiani, R., Tacchella, A.: NuSMV 2: an open source tool for symbolic model checking. In: 14th Conference on Computer Aided Verification, LNCS, vol. 2404. Springer, Heidelberg (2002)
21. Bolotov, A.: A clausal resolution method for CTL branching-time temporal logic. J. Exp. Theor. Artif. Intell. **11**(1), 77–93 (1999)