Runtime Verification of Multi-agent Systems Interaction Quality

Najwa Abu Bakar and Ali Selamat

Faculty of Computer Science and Information System
Universiti Teknologi Malaysia
81300, Skudai,
Johor Darul Takzim
najwa.abakar@gmail.com, aselamat@utm.my

Abstract. Since multi-agent systems are inherently complex, there are possibilities that errors related to multi-agent systems interaction could occur. Currently, many verification approaches have been proposed by focusing on specific properties, using a particular technique and during certain development phase. However, each technique has its limitations. As interaction between agents and multi-agent systems environments evolve during runtime, not all multi-agent systems interaction requirements can be specified and verified during design and development. Thus, some new interaction properties such as agent availability and trustability need to be verified during runtime. In this research, a solution is proposed in which newly defined agents interaction quality requirements are specified, developed into metrics and verified within multi-agent systems runtime verification framework. It is aimed to improve the effectiveness of the verification of agent interactions during runtime. Finally, an experiment is set up to capture message passing between agents and to gather runtime system profiles to evaluate the proposed solution.

Keywords: Multi-agent systems, runtime verification, interaction quality, message passing.

1 Introduction

Existing MAS verification approaches [36] can be divided into three stages that are verification during 1) design, 2) development and 3) runtime. Although the approaches, that have been properly designed, tested using case studies and evaluated, have marginally improved correctness of MAS, each level of approaches has its own limitations and a lot of improvements are still needed. First, for verification during design, formal verification is performed automatically using model checking [1][5][8][29], automated theorem proving [32], or simulation [21]. As MAS is inherently complex [11], performing model checking only manages to verify correctness of certain properties [7][16][30]. Besides, as the model checking rely on the input which are properties specification and design model [9][15][17][20], the quality of these two inputs contribute to the accuracy of the checking. Coverage or

A. Selamat et al. (Eds.): ACIIDS 2013, Part I, LNAI 7802, pp. 435–444, 2013.

degree of thoroughness of the specified properties and accuracy of the modeling still need to be measured in order to assess and increase the accuracy of the model checking output. Second, similarly, during development [22], MAS debugging and testing [28] also suffer from the incomprehensiveness of the data captured during monitoring and the thoroughness of the test suites. Finally, verification during runtime [2][12][24][27] that focuses only towards agents, integration, and message passing without considering supporting contextual information also faces similar issues that are incompleteness of the data to be analyzed [4].

The above discussed problems highlight the current MAS correctness verification issues that need to be improved by tackling the comprehensiveness and pre-processing or data preparation issues of the verification. For the MAS that has been developed and executed, the only way to assess its correctness is by performing analysis towards MAS interaction and communication activities. The richness of the data captured during those activities has opened the opportunity for low-level analysis of the data, i.e. interaction messages that can be manipulated and prepared to fulfill the gap between low-level MAS infrastructure and the verification and analysis at higher-level. Observing this issue from software quality point of view, it is believed that the incomprehensive verification issue faced by existing verification approaches can be improved by considering the MAS interaction resources data and data captured during interaction activities. Implementing the framework during runtime complements the existing verification performed during design [3] using model checking or theorem proving [1][5][8][29][32] in which new requirements can be defined and verified after the systems have been executed [23][33].

In this paper, we propose a framework called Multi-agent Runtime Verification (MARV) framework. In this framework, we define new requirements of MAS interaction during runtime and develop new metrics to improve the effectiveness of MAS interaction verification. This framework will be implemented in order to evaluate the effectiveness of the metrics in improving MAS interaction error detection rate. Two interaction qualities to be verified are defined in this paper that are the agents *availability* and *trustability*.

The rest of the paper is organized as follows. Section 2 provides the research background and analysis of the existing works. Section 3 describes the proposed solution that is the requirements definition process within MARV framework and in Section 4, we describe the implementation and evaluation consideration. Finally, in Section 5, we present the discussion and future work.

2 Background and Related Works

2.1 MAS Interaction Verification

There are three levels of approaches to improve correctness of MAS interactions that are design level, development level and runtime level. During design, verification of MAS designs are performed against specified interaction properties and during development, debugging and testing are performed to find bugs. During runtime, there are two main components implemented that are monitoring and runtime verification [12][27]. During design, many studies have been performed in the area of

MAS verification. There are many approaches proposed. One of them is by inventing tools e.g. MCMAS (model checker) [25][29] and SOCS-SI (automated theorem proving) [32] customized to perform formal verification on MAS during design time by considering common agent abstractions. Another approach is by checking on selected critical smaller models extracted from the complex MAS and use general-purpose distributed model checking tools such as SPIN, UPPAAL, etc to verify against general properties such as temporal logic [3]. Both mentioned approaches are performed during design time. The framework proposed in this paper aims to complement these approaches. New interaction requirements are defined after the systems are executed based on MAS agents and environment contextual information.

2.2 Existing MAS Interaction Requirements

From the literature, MAS requirements can be divided into two categories that are MAS functional requirements and non-functional requirements. Functional requirements for a system are the definition of the behavior or functions of the systems, what the system "shall do" (behavior) while non-functional requirements define how the system "shall be" (constraints). The functional requirements are explained in the system design while the non-functional requirements are detailed out in the system architecture [13][19]. *Functional requirements* for agents are the characteristics of agents that are autonomous, reactive, proactive, and coordinating based on agents roles, types, goals, and tasks assigned to them. For example, mobile agents are able to migrate from one platform to another to achieve goals and complete tasks. *Non-functional requirements* are based on the constraints of the agents such as safety, integrity and quality of agents to protect and to ensure that the agents are reliable and can be trusted [31][34][35]. Examples of non-functional requirements of MAS are:

- Compatibility; the ability to follow the standard specified by FIPA [18] and application ontology.
- Interoperability; the ability to work with other agents and applications
- Safety; the ability of the agents to protect themselves from threats.
- Credibility; trust and reputation of agents.
- Integrity; to ensure that messages between agents, between agents and agent platform, and between agents with environment are not altered.
- Availability; to ensure services are not suffered from denial of service (DoS)
- Confidentiality; to ensure that only certain agents with certain roles, trusted levels, or reputation are allowed to access information.

Investigation towards existing works in MAS verification shows that these MAS requirements are specified as MAS properties that can be classified into interaction, behavior, and knowledge properties for verification. In this research, we are focusing on MAS interaction issues.

3 Requirements Definition Process in MARV

3.1 Definition Process

In the requirements definition phase in MARV, a few steps are performed: 1) Defining Agents Interaction Quality Criteria, 2) Defining the Agents Interaction Data, 3) Defining Agents Interaction Quality Requirements and Parameters, and 4) Defining the Agents Interaction Quality Rules and Metrics. These steps are explained in the following subsections.

3.2 Defining Agents Interaction Quality Criteria

The first task in designing MARV is to determine what factors constitute a successful agent conversation or interaction. The quality of agents interaction here means the success of the conversation according to the specifications and application's contextual condition. In other words, the selected criteria are preferably in measurable form that can be used to quantify a message as having the characteristics to be successful in agent conversation. A good start in this direction is to refer to other works in data quality, such as Total Quality Management (TQM) [10] and Information Quality (IQ) [26]. These studies suggested that data quality can be defined as combination of scores from multiple data quality parameters. Some of the commonly used data quality parameters are accuracy, completeness, relevancy, timeliness and credibility. These are general parameters to describe the criteria for information or product. In this case, these general criteria have to be adapted and shaped towards the area of communication of multi-agent systems. Whatever criteria that are eventually selected, they need to be accurate and comprehensively selected since they represent the quality of the conversation messages in general. Furthermore, the metrics (or scores) generated for these criteria will determine the accuracy of the verification result, which has the direct impact in classifying messages into valid or invalid messages during analysis stage. Figure 1 shows the verification process cycle adapted from the TQM.

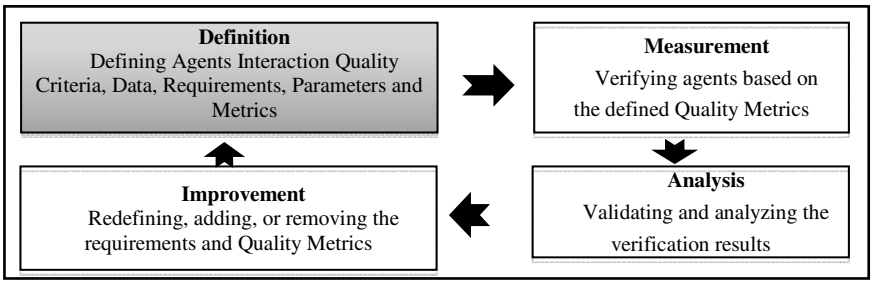

Fig. 1. The verification process cycle of the MARV components adapted from the TQM. The definition process is highlighted in grey box.

3.3 Defining the Agents Interaction Data – The Messages

The raw data to be verified in MARV is the set of agents interaction "messages". A message is the basic element that forms conversation during agents communication. Hence, it is logical to start the definition stage by looking at the characteristics of a typical message. Table 1 below shows a sample message labeled with the attributes on the left column that are AgentId, ConversationId, Sender, Receiver, Performative, Content, Language, Ontology, Protocol, and Reply-by.

Table 1. Sample of agents interaction message

Message Attributes	Example
AgentId	1
ConversationId	1
Sender	ipa@mydomain.com
Receiver	aa@yourdomain.com
Performative	request
Content	""((action (AgentId :name coordinator@yourdomain.com)(contact-student :offer-study-level Master Degree :course Computer Science :registration 01/09/2012)))""
Language	FIPA-SL
Ontology	university-online-application
Protocol	fipa-request
Reply-by	null

3.4 Defining Agents Interaction Quality Requirements

In order to verify the agent interaction messages, the general data quality requirements have been identified to show the validity of a message as depicted below. There are many other requirements but here, the most important, relevant, and related to agents, platforms, hosts, environments or networks, and devices/hardware (contextual information) are identified. The list is not exhaustive; rather it should be considered as the minimal interaction correctness requirements for interaction verification. The list can be extended further as the systems profile, knowledgebase and configuration evolve during runtime. The requirements, R1, R2 and R3 are listed below:

R1: The receiver agent is available during message transmission. The assumption here is that if the receiver is available, there is high chance that the message will be received successfully compared to unavailable or busy receiver agent.

R2: The receiver agent is authorized to receive the content of INFORM message. Here, it is assumed that critical information can only be received by agents that have the authority. Only certain agents with specific roles and located in trusted location (platform, host, and network) can receive certain information.

R3: The sender agent is trusted to send an INFORM or a REQUEST message. Only certain agents with specific roles and located in trusted location (platform, host, or environment) can send certain information and request.

The abstraction of *agent interaction quality requirements* stated above can be translated into a more concrete form of representation known as *agent interaction quality metrics.* Rules and indicators (in the form of scores) are associated for each parameter based on the related conditions, and the scores. In the following subsection, agents interaction quality metrics for each interaction quality requirements are defined.

3.5 Defining the Agents Interaction Quality Rules and Metrics

A data quality metric is a measure of some property of a piece of information or its specifications. In order to measure the agents interaction quality, based on the abstract agents interaction requirements defined above, in this phase, the agents interaction quality rules are constructed. The message correctness requirements identified during definition process are transformed into rules and measurable formulations or indicators (in the form of scores) that reflect the specifications and real condition (supported by contextual information) of the messages. Table 2 presents the proposed rules for each interaction quality requirement. The rules are constructed based on the interaction quality requirements for interaction verification identified in Section 3.4 above.

The rule for each agents interaction metric is a simple if-then-else statement typical in many programming algorithm. Basically, for each metric, the rule checks via its if-then-else statement, the message's attributes value against the monitored information in agent profiles and MAS knowledgebase resources, and then assign score accordingly as shown in Table 2 below.

Table 2. The constructed agents interaction rules and metrics

Require -ments	Metrics	Message Attributes	Rules
Layer 1: Agent Verification Level			
R1	Availability-receiver (AR)	Receiver	For each message, if the receiver agent is available to receive a message, score=1 else score=0.
R2	Trustability-receiver (TR)	Performative Receiver	For INFORM message, if the receiver agent is authorized, score=1 else score=0.
R3	Trustability-sender (TS)	Performative Sender	For INFORM message, if the sender agent is trusted, score=1 else score=0.

As shown in Table 2, the verification of sender and receiver agents includes the checking of receiver agent's availability and trustability of the sender and receiver. Agents as the main players in MAS interactions are the main factors of the success of the interaction and also the main sources for the communication errors to occur. Thus, agents are required to be available and trustable during interaction.

4 Implementation and Evaluation Consideration

The definition processes are implemented within MARV framework. The architecture is shown in Figure 2 below. MARV consists of four main verification process components: Definition, Measurement, Analysis, and Improvement adapted from TQM (refer Section 3.2). In addition, MARV also includes the additional components that are the *Agent Communication Language* (*ACL) messages capturing*, *Agents and MAS infrastructure profiling* and *MAS Knowledgebase gathering*.

During ACL messages capturing, the collected message's attributes are transformed into database fields. The database fields include: SessionId, MessageId, Timestamp, Valid, Size, Late, ConversationId, Sender, Receiver, Performative, Content, Language, Ontology, Protocol, and Reply-by. Next, to gather agents and MAS infrastructure profiles, the list of agents and the attributes related to each agent and its infrastructure are stored in database fields. The database fields include: SessionId, AgentId, and AgentName. In this research, the tables are expanded by adding several more database fields to provide extra information related to the infrastructure, environment, and location of the agents: Container, Platform, Host, and Network. Finally, for MAS knowledgebase gathering, there are two main sources of the knowledgebase that are the ontologies and interaction protocols. Both the MAS's specified ontology and used interaction protocol are identified and the information are extracted from the coding and FIPA standard website to be stored in database fields.

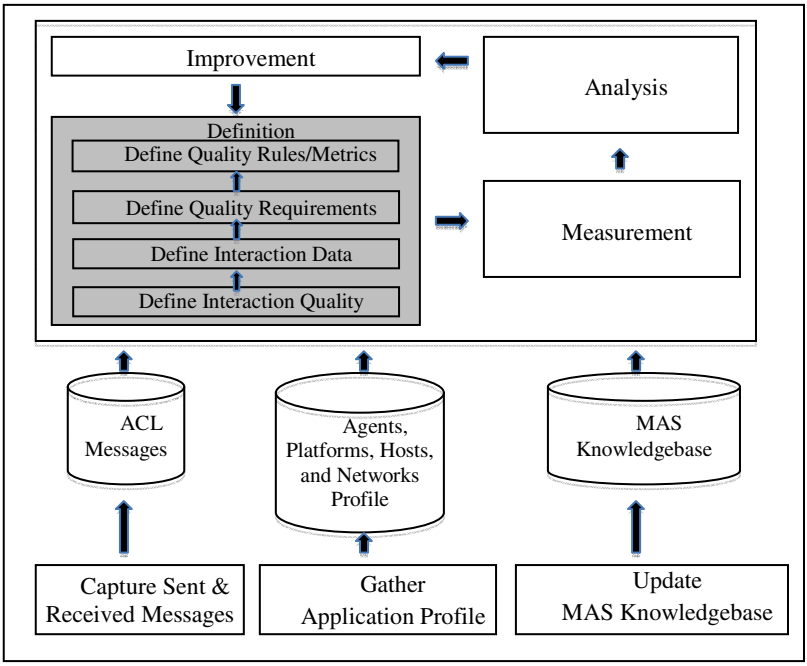

Fig. 2. The definition processes within MARV framework architecture (highlighted in grey boxes)

To test the framework, experiments are performed. Given a set of exchanged messages, considering the MAS knowledgebase and infrastructure profiling, benchmark data are prepared by manually classifying the messages into correct and incorrect messages. The incorrect messages identified during this manual classification stage is called *known incorrect messages*. These known incorrect messages are the messages that contain errors supposed to be detected using MARV during runtime. From the known incorrect messages, *known unique errors* are extracted. These errors are supposed to be detected by the defined metrics and rules (properties). Next, using the MARV, the messages are verified during runtime and classified automatically into correct and incorrect messages. The identified suitable evaluation metrics to measure the effectiveness of the MARV tool in performing MAS interaction verification are *precision* and *recall*, *properties coverage* and *time* taken to perform the verification.

5 Discussion and Future Work

We have presented the MARV, a runtime multi-agent verification framework to verify MAS interaction. This effort to detect agents message passing errors during runtime can increase the effectiveness of MAS verification. This is because the correctness of the MAS not only considers design issues but also other runtime factors such as agents profile, knowledgebase and configuration that can evolve during runtime. The experiment is set up using JADE [6][14] environment as one of the case studies. It is to implement the MARV framework that includes the capturing of message passing between agents and the gathering of runtime system profiles to evaluate the proposed solution. In the future, the presented rules and metrics presented in this study will be properly developed into mathematical formula that will be used to design a verification algorithm.

Acknowledgments. The authors thank Ministry of Higher Education Malaysia (MOHE) under Fundamental Research Grant Scheme (FRGS) Vot 4F031 and Universiti Teknologi Malaysia under Research University Funding Scheme (Q.J130000.7110.02H47) for supporting the related research.

References

1. Latif, N.A., Hassan, M.F., Hasan, M.H.: Formal Verification for Interaction Protocol in Agent-Based E-Learning System Using Model Checking Toolkit - MCMAS. In: Zain, J.M., Wan Mohd, W.M.b., El-Qawasmeh, E. (eds.) ICSECS 2011, Part II. CCIS, vol. 180, pp. 412–426. Springer, Heidelberg (2011)
2. Abdul Bujang, S., Selamat, A.: Verification of Mobile SMS Application with Model Checking Agent. In: Proceedings of the 2009 International Conference on Information and Multimedia Technology, ICIMT 2009, pp. 361–365. IEEE Computer Society, Washington, DC (2009)

3. Abu Bakar, N., Selamat, A.: Analyzing model checking approach for multi agent system verification. In: 2011 5th Malaysian Conference on Software Engineering, MySEC, pp. 95–100 (2011)
4. Abu Bakar, N., Selamat, A.: Towards Implementing Dynamic Multi-agent V&V Framework. In: The Third Software Engineering Postgraduates Workshop, SEPoW 2011, JB, Malaysia (2011)
5. Alechina, N., Logan, B., Nguyen, H.N., Rakib, A.: Automated Verification of Resource Requirements in Multi-Agent Systems Using Abstraction. In: van der Meyden, R., Smaus, J.-G. (eds.) MoChArt 2010. LNCS, vol. 6572, pp. 69–84. Springer, Heidelberg (2011)
6. Bellifemine, F.L., Caire, G., Greenwood, D.: Developing multi-agent systems with JADE. John Wiley & Sons, Ltd., West Sussex (2007)
7. Ben Ayed, L.J., Siala, F.: Event-B based Verification of Interaction Properties In Multi-Agent Systems. Journal of Software 4(4), 357–364 (2009)
8. Benerecetti, M., Cimatti, A.: Validation of Multiagent Systems by Symbolic Model Checking. In: Giunchiglia, F., Odell, J.J., Weiss, G. (eds.) AOSE 2002. LNCS, vol. 2585, pp. 32–46. Springer, Heidelberg (2003)
9. Berard, B., Bidoit, M., Finkel, A., Laroussinie, F., Petit, A., Petrucci, L., et al.: Systems and Software Verification: Model-Checking Techniques and Tools (1999)
10. Besterfield, D., Besterfield-Michna, C., Besterfield, G., Besterfield-Sacre, M., Urdhwareshe, H., Urdhwareshe, R.: Total Quality Management. Pearson Education (2011)
11. Bordini, R., Dastani, M., Dix, J., Seghrouchni, A.: Multi-Agent Programming: Languages, Tools and Applications. Springer, NY (2009)
12. Botía, J.A., Gómez-Sanz, J.J., Pavón, J.: Intelligent Data Analysis for the Verification of Multi-Agent Systems Interactions. In: Corchado, E., Yin, H., Botti, V., Fyfe, C. (eds.) IDEAL 2006. LNCS, vol. 4224, pp. 1207–1214. Springer, Heidelberg (2006)
13. Burnstein, I.: Practical Software Testing: A Process-Oriented Approach. Springer, NY (2003)
14. Caire, G., Pieri, F.: JADE. Java Agent Development Framework (2011), `http://jade.tilab.com/doc/tutorials/LEAPUserGuide.pdf` (retrieved May 18, 2012)
15. Clarke, E.M., Grumberg, O., Peled, D.A.: Model Checking. The MIT Press, Cambridge (1999)
16. Dekhtyar, M.I., Dikovsky, A.J., Valiev, M.K.: Temporal Verification of Probabilistic Multi-Agent Systems. In: Avron, A., Dershowitz, N., Rabinovich, A. (eds.) Pillars of Computer Science. LNCS, vol. 4800, pp. 256–265. Springer, Heidelberg (2008)
17. Dennis, L., Fisher, M., Matthew, W.P., Bordini, R.H.: Model Checking Agent Programming Languages. Automated Software Engineering 19(1), 5–63 (2012)
18. FIPA.: The Foundation for Intelligent Physical Agents (2012), `http://www.fipa.org/` (retrieved May 18, 2012)
19. Fulcher, J.: Advances in Applied Artificial Intelligence. Idea Group Publishing, London (2006)
20. Gammie, P., van der Meyden, R.: MCK: Model Checking the Logic of Knowledge. In: Alur, R., Peled, D.A. (eds.) CAV 2004. LNCS, vol. 3114, pp. 479–483. Springer, Heidelberg (2004)
21. Giese, H., Klein, F.: Systematic verification of multi-agent systems based on rigorous executable specifications. International Journal of Agent-Oriented Software Engineering 1(1), 28–62 (2007)
22. Gómez-Sanz, J.J., Botía, J., Serrano, E., Pavón, J.: Testing and Debugging of MAS Interactions with INGENIAS. In: Luck, M., Gomez-Sanz, J.J. (eds.) AOSE 2008. LNCS, vol. 5386, pp. 199–212. Springer, Heidelberg (2009)

23. Hallé, S., Villemaire, R.: Runtime Verification for the Web: A Tutorial Introduction to Interface Contracts in Web Applications. In: Barringer, H., Falcone, Y., Finkbeiner, B., Havelund, K., Lee, I., Pace, G., Roşu, G., Sokolsky, O., Tillmann, N. (eds.) RV 2010. LNCS, vol. 6418, pp. 106–121. Springer, Heidelberg (2010)
24. Selamat, A., Lockman, M.T.: Multi-agent Verification of RFID System. In: Nguyen, N.T., Katarzyniak, R.P., Janiak, A. (eds.) New Challenges in Computational Collective Intelligence. SCI, vol. 244, pp. 255–268. Springer, Heidelberg (2009)
25. Lomuscio, A., Qu, H., Raimondi, F.: MCMAS: A Model Checker for the Verification of Multi-Agent Systems. In: Bouajjani, A., Maler, O. (eds.) CAV 2009. LNCS, vol. 5643, pp. 682–688. Springer, Heidelberg (2009)
26. Naumann, F.: Quality-Driven Query Answering. LNCS, vol. 2261. Springer, Heidelberg (2002)
27. Osman, N.: Runtime Verification of Deontic and Trust Models in Multiagent Interactions. Phd Thesis (2008)
28. Poutakidis, D.: Debugging Multi-Agent Systems With Design Document. Phd Thesis (2008)
29. Raimondi, F.: Model checking multi-agent system. Phd Thesis (2006)
30. Sabri, K.E., Khedri, R., Jaskolka, J.: Verification of Information Flow in Agent-Based Systems. In: Babin, G., Kropf, P., Weiss, M. (eds.) MCETECH 2009. LNBIP, vol. 26, pp. 252–266. Springer, Heidelberg (2009)
31. Silva, C., Pinto, R., Castro, J., Tedesco, P.: Requirements for Multi-Agent Systems. In: Workshop em Engenharia de Requisitos, WER, Piracicaba-SP, Brasil, pp. 198–212 (2003)
32. Singh, M.P., Chopra, A.K.: Correctness Properties for Multiagent Systems. In: Baldoni, M., Bentahar, J., van Riemsdijk, M.B., Lloyd, J. (eds.) DALT 2009. LNCS, vol. 5948, pp. 192–207. Springer, Heidelberg (2010)
33. Stoller, S.D., Bartocci, E., Seyster, J., Grosu, R., Havelund, K., Smolka, S.A., Zadok, E.: Runtime Verification with State Estimation. In: Khurshid, S., Sen, K. (eds.) RV 2011. LNCS, vol. 7186, pp. 193–207. Springer, Heidelberg (2012)
34. Sycara, K.P.: Multiagent Systems. AI Magazine 19(2), 79–92 (1998)
35. Wooldridge, M.: An Introduction to MultiAgent Systems, 2nd edn. John Wiley & Sons, United Kingdom (2009)
36. YAHODA.: Verification Tools Database (2011), `http://anna.fi.muni.cz/yahoda/` (retrieved May 18, 2012)