Complexity Measurement of Multi-Agent Systems

Toufik Marir[1], Farid Mokhati[1,2], Hassina Bouchelaghem-Seridi[3], and Zouheyr Tamrabet[1]

[1] Department of Mathematics and Computer Science, University of Oum El Bouaghi, Algeria
{marir.toufik,mokhati,tamrabet.zouheyr}yahoo.fr
[2] LAMIS Laboratory, University of Tebessa, Algeria
[3] Department of Computer Science, LABGED Laboratory, University of Annaba, Algeria
seridi@labged.net

Abstract. Multi-Agent Systems (MAS) is a promising software paradigm. Considered as a natural metaphor to modeling complex systems, MAS are applied to develop a wide range of applications. However, the developed system's complexity is a hard obstacle to understand and maintain them. In this paper, some metrics are presented to measure the complexity of MAS. The proposition of these metrics is passed through the proposition of a complexity model for MAS. To validate our proposal, a tool has been developed to measure the JADE-based applications complexity. Furthermore, the collected metrics can also be used as a base to estimate the required effort to maintain *JADE*-based applications.

Keywords: Multi-Agent Systems, Complexity, Measurement, JADE.

1 Introduction

Multi-Agent Systems (MAS) is a promising software paradigm. It is applied nowadays to develop a wide range of applications from games to space shuttles. Specifically, we can consider it as an ideal paradigm to develop complex systems [1]. In fact, this paradigm provides several characteristics which allow modeling complex systems in natural way. The distribution of execution, the flexibility of agents and the richness of interaction's modes are examples of characteristics that motivate the use of such paradigm to develop complex systems.

The complexity notion is associated to the difficulty degree to understand a system [2]. This notion is a key factor in the development cost estimation and effort [3]. Moreover, it influences the understandability of developed software product. Consequently, the software product complexity has an impact on the maintenance effort and cost. Therefore, measuring the complexity of software product can be used as an indicator to estimate the required effort during the maintenance phase.

We think that the above characteristics of MAS (distribution, flexibility and richness of interaction's modes) can deepen the complexity effects. For example, the flexibility of agents makes their behaviors unpredictable and the understandability of developed system more difficult. Thus, the maintenance phase becomes more

J.P. Müller, M. Weyrich, and A.L.C. Bazzan (Eds.): MATES 2014, LNAI 8732, pp. 188–201, 2014.

complex. We address in this work the measurement of the complexity of MAS code. We propose a metrics that can be used as means to assess a developed MAS or as an indicator to estimate the required effort during the maintenance process. Before going ahead to the metrics presentation, we must first explain the proposed model for the complexity of MAS in order to identify the different facades affecting it. A tool has been developed to collect the proposed metrics for *JADE* platform.

The remainder of this paper is organized as follow: some related works are presented in section 2. Section 3 is devoted to present a model for the complexity of MAS followed by the presentation of proposed metrics to assess it (in section 4). In section 5 we present a tool we developed for collecting automatically the above metrics. Section 6 discusses our actual research, draws some conclusions and gives some future work directions.

2 Related Works

The complexity of software is a *critical question* during all the software development phases. Consequently, it has been studied quite a long time. McCabe [4] proposed one of the influential metrics to measure the complexity of software, called the *cyclomatic number*. This measure allows, among others, to estimate the required effort to understand the software code. *Although* it is old, this metric is still used in new works [5].

It seems evident that MAS, as a software paradigm, require their own development approaches [6]. Especially, we are in need to specific metrics to measure the different aspects of agent-based software. Several proposed metrics are presented by Dumke et *al.* [7].Measuring the complexity of agent-based software is our main purpose in this paper. We think that measuring the complexity can be used as an indicator to control the development of agent-based software and estimate the required effort to maintain it.

In the MAS field, the complexity has been studied across different points of view. Some studies targeted the computational complexity of MAS [8, 9]; others studied the complexity of MAS code [10, 11]. Our work shares the same goal with this second category of studies. The complexity of mobile agents implemented with *AspectJ* is studied by Dospisil [10]. The proposed metrics are based on the entropy measure. This work is mainly devoted to study the influence of implementing the interaction between mobile agents using *AspectJ* on the complexity of the developed software. Thus, we believe that the limited context of this study (the mobile agents implemented with *AspectJ*) affects negatively the proposed metrics applicability for general MAS.

In order to compare agent-based simulation to other simulation paradigms, Klügl [11] proposed metrics to measure the complexity of multi-agent simulations. These metrics can be classified into three categories: the overall system-level, the agent-level and the agent-system-level metrics. Many metrics are proposed for each level. The *Number of Agent Types*, the *Number of Resources Types* and the *Maximum Number of Agents* are examples of overall system-level metrics. The proposed metrics are closely related to simulation models (i.e., by considering the specificities of these models). Hence, relation between the model and the original system should be considered [11].

The author noted that the list of metrics is not complete. However, we believe that the lack of important metrics that affect significantly the complexity of MAS (like the interaction between agents) is a real drawback. Several missed metrics are more significant than the presented ones. Moreover, he emphasized that some metrics (like the *Size of Procedural Knowledge*) can be measured only for some kinds of MAS. In fact, MAS can be implemented using various software paradigms (such as the object-oriented programming or the knowledge-based systems). We think that it is important to consider the used software paradigm specificities in implementing MAS in order to propose metrics. Obviously, the software paradigm used to implement MAS can influence not only the metrics proposition but also the measurement method used to collect the metrics. The ISO/IEC 9126 quality standard [12] considered the measurement method as an essential part of metrics. Nonetheless, this aspect is omitted in the above work.

We think that the proposition of the three levels of complexity (overall system-level, agent-level and agent-system-level) is important to analyze and understand the complexity of MAS. Even if the existence of the three levels is justified for simulation models, it is not the case for general MAS. In fact, MAS are defined as a set of interacting agents in an environment. Consequently, we see that the agent-system-level is a natural part of overall system-level.

As conclusion, designed for agent-based simulations, the proposed metrics cannot be used for any other agent-based software. Consequently, it seems important to propose metrics to assess the complexity of MAS. As it is shown above, the complexity metrics can be used to evaluate a developed MAS. Therefore, the complexity metrics provide strong base to plan the maintenance phase. In order to propose the complexity metrics, firstly, we should specify the complexity notion of MAS. The next section is devoted to present a complexity model of MAS.

3 Complexity Model for Multi-Agent Systems

Before starting the measurement of the complexity of multi-agent systems, we should formulate the complexity concept. This concept should also be specified by considering the features of multi-agent systems.

One of the most accepted definition of the complexity is proposed by IEEE in the Standard Glossary of Software Engineering Terminology [2] as "*the degree to which a system or component has a design or implementation that is difficult to understand and verify*". Thus, the complexity is closely related to the required effort to understand, verify and maintain a software product. However, given a general statement, like the definition proposed by IEEE, is not enough. This definition did not give the possible causes that can influence the difficulty to understand software product. Therefore, we propose a model of the complexity concept in the MAS context. The proposed model simplifies the understanding and studying the MAS complexity.

MAS can be informally defined as a set of interacting agents in an environment. Naturally, the number of agents and interaction between them are the main factors that influence the MAS complexity. Furthermore, at the lower granularity level, the

agent is not an atomic entity. Accordingly, the complexity of agents has a direct influence on the MAS complexity. To conclude, the MAS complexity may be viewed at two distinguished levels: agent-level and system-level (Fig. 1). By the agent-level complexity we intend to study, separately, the complexity of each agent (without taking its interaction with other agents into account). On the other side, the system-level specifies the complexity of the environment and the interaction between agents.

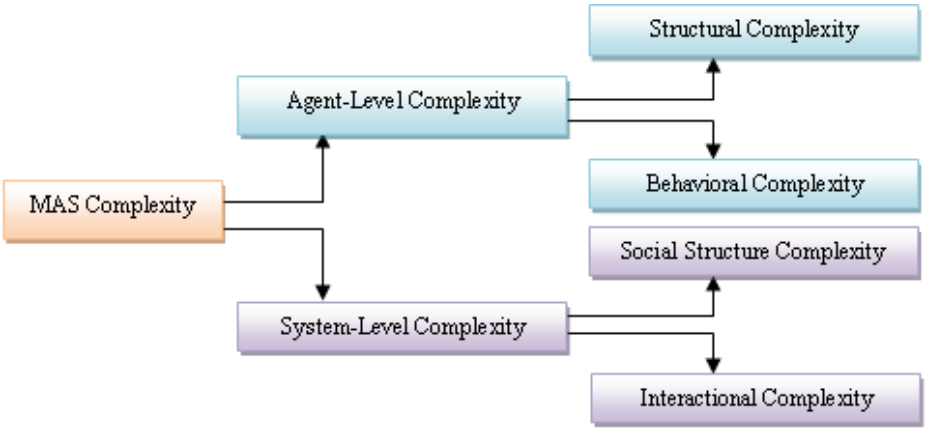

Fig. 1. A complexity model for multi-agent systems

The agent-level complexity can be analyzed using two orthogonal criteria: the complexity of the agent's structure and the complexity of agent's behaviors. The complexity of the agent's structure represents the internal elements composed the agent including the complexity of its knowledge. Obviously, the behavioral complexity is the complexity of the different behaviors implemented inner the agent in order to achieve its goal.

At the system-level, the complexity is composed of the social's structure complexity and the interactional complexity. The social structure indicates the global structure of the MAS. It encompasses all the agents composing the MAS and the objects that exist in its environment. The interactional complexity designates the collective behavior of the MAS. The collective behavior of this latter is ensured by the direct interaction between agents or the indirect interaction between them through the manipulation of the environment objects.

After specifying the complexity model of MAS, we should propose some metrics in order to measure each of the four components of our model: structural, behavioral, social structure and interactional complexities. The next section is devoted to this purpose.

4 Measuring the Complexity of Multi-Agent Systems

Modeling the complexity concept of MAS is the first step to assess it. This step should be followed by the proposition of metrics in order to measure, objectively, the

complexity of the developed MAS. According to Alonso et *al.* [13], the measurement of metrics in MAS field is closely dependent on the implementation paradigm used to develop this system. Because the MAS can be implemented using various software paradigms (object-oriented paradigm, knowledge-base systems, etc), it seems hard difficult to propose adequate metrics for all these software paradigms. Consequently, we propose in this paper, only some metrics for MAS which are implemented using the object-oriented paradigm. As we will present bellow, the presented metrics will be applied to *JADE* platform.

The ISO/IEC 9126 quality standard [12] emphasized that the specification of the measurement method is essential in the proposition of metrics. According to this recommendation, we start this section by explaining the measurement method used in assessing the proposed metrics.

4.1 The Measurement Method

In software engineering, the metrics can be static or dynamic. The static metrics are assessed without executing the software. On the other side, the dynamic metrics are collected during the execution of the software. For difficulty reasons with assessment of dynamic metrics, the static ones are the most used [14]. However, we think that the two kinds of metrics are complementary. For instance, in dynamic system the static analysis of code cannot give the number of agent composed the MAS. Consequently, we opted in this paper, to both static and dynamic metrics.

The static metrics are collected by analyzing the code of the MAS. Based on the different language constructors of *JAVA* and specifically *JADE* platform, we can extract useful information about the complexity of the software.

The dynamic metrics are collected thanks to the aspect paradigm [15]. This software paradigm allows us to specify the metrics independently to the MAS as aspects. The implemented metrics are woven automatically in the adequate points of the MAS in order to pick up the execution trace of the software. Using the aspect paradigm we can measure the dynamic metrics without updating the code of the implemented MAS. Using *AspectJ* we can easily implement the metrics and use them with *JADE* applications.

4.2 The Proposed Metrics

This section presents the proposed metrics. Nevertheless, it is important to note that the proposition of a complete and exhaustive list of complexity metrics is beyond the scope of this paper. The proposed metrics to assess the complexity of MAS are:

1. **The structural complexity metrics:** we propose the following metrics to measure the structural complexity of agents:

   (a) ***The Size of the Agent's Structure (SAS):*** The agent's structure is presented using set of attributes. Generally, these attributes are used to specify the state of the agent. Consequently, increasing the number of attributes implies increasing

the number of agent's states represented by the attributes combination. The variety of the agent's states influence the analysis of the agent. Thus, we use the *SAS* metric as an indicator to calculate the complexity of the agent's structure.

(b) ***The Agent's Structure Granularity (ASG):*** some attributes that composed the agent's structure can be objects which are composed also of attributes. Obviously, the structure of the agent will be more complex if it is composed of several composite attributes. We can represent the attributes of an agent in a tree known that its root is the agent, each node is an attribute and the leaves are the primitive attributes. Known that the height of the root is zero, this metric represents the average height of the nodes.

$$ASG = \frac{\sum_{i=1}^{k} N_i}{K-1} \quad (1)$$

Where k is the number of the nodes in the tree, N_i is the height of the i^{th} node. We used $(K\text{-}1)$ as the divider in this metric in order to exclude the root of the tree because it is not an attribute. If an agent has not attributes, its structure granularity becomes naturally zero.

(c) The Dynamicity of the Agent's Structure (DAS): in addition to the composed attributes, the ones of the agent can be of a container nature. By the container attributes we mean the attributes allowing adding and removing variables like the list. Known the number of variables encompassed in the container allows knowing exactly the size of the agent's structure. Moreover, the extensive dynamicity of the agent's structure designates instability in the agent's structure which means more complexity. Hence, the dynamicity of the agent's structure is measured by identifying the structure update between two moments.

$$DAS = \frac{SC_t - SC_{t'}}{t - t'} \quad (2)$$

Where SC_t (respectively $SC_{t'}$) is the size of the container in the moment t (respectively t').

2. The behavioral complexity metrics: the behavioral complexity of the agent can be assessed using:

(a) ***The Behavioral Size of the Agent (BSA):*** is the number of behaviors ensured by agent. This metric gives an indicator to the degree of agent's specialization. Obviously, an agent that ensures various behaviors is more complex than the one which ensures fewer behaviors.

(b) The Average Complexity of Behaviors: the previous metric seems not sufficient because an agent who ensures several simple behaviors may be simpler than an agent that ensures only one complex behavior. The *JADE* platform, for example, gives the possibility to define composite behaviors. We can specify the composite behaviors by finite state machine. Hence, the complexity of a behavior can be calculated using the *cyclomatic number* proposed by McCabe [4]. Taking a composite behavior (B) presented by graph (G), its *cyclomatic* complexity (CC_B) is calculated by the equation:

$$CC_B = (E_G - N_G) + 2P_G \quad (3)$$

Where E_G is the number of edges of the G; N_G is the number of node of G and P is the number of connected component of G. It is clear that the complexity of simple behaviors is 1. The complexity of an agent's behaviors is calculated as the average of all its behaviors.

(c) ***The Average Number of Scheduled Behaviors (ANSB)*:** an agent can launch several behaviors. Then, a started behavior is inserted into a scheduling list until its turn. The scheduling behaviors are those that are waiting for their execution. This mechanism is made for ensuring the concurrency in the agent's functionalities. However, this concurrency may be the cause of the difficulty to understand the functionalities of an agent which change frequently the executing behavior. Moreover, this concurrency requires managing the behaviors' priorities for ensuring coherent and validating results. In the complex systems, the management of the scheduling behaviors is a difficult task. Hence, we propose to calculate the average number of scheduling behaviors in each moment as an indicator of the concurrency intra-agent. As it is explained above, despite the advantages of the concurrency inner the agent, it represents a cause of the complexity.

3. **The social structure complexity metrics:** a MAS is composed of set of agents existing in an environment. The environment is composed of set of objects. Consequently, we can measure the complexity of the social structure of MAS by:

(a) The Heterogeneity of Agents (HA): this metric indicates the number of classes of agents. We think that MAS composed of heterogeneity agents is more difficult to be understood than a homogeny MAS. Moreover, to maintain heterogeneous MAS we need to update several agents.

(b) The Heterogeneity of the Environment's Objects (HEO): the environment of the MAS is composed of objects. Like the previous metric, we think that the existence of heterogeneous objects composed the environment can be considered as an indicator to the complexity of the MAS. The heterogeneity of the environment's objects metric corresponds to the number of classes of environment's objects.

(c) The Size of the Agent's Population (SAP): in the MAS, the agents can be created and killed dynamically. Known that the agents operate in concurrence, increasing their number increases the complexity of the MAS. Moreover, increasing the number of agents introduces more interactions between them. Consequently, the complexity of the MAS increases.

4. **The interactional complexity metrics:** the interaction between agents is ensured directly by means of messages or indirectly by manipulating the object of the environment. We measure the interaction between agents by:

(a) ***The Rate of Interaction's Code (RIC):*** this metric presents the rate of source code devoted to ensure the communication between agents. Consequently,

$$RIC = \frac{SC}{SB} \tag{4}$$

Where *SC (Size of Communication)* is the total number of line of code devoted to the communication between agents composed the MAS and *SB (Size of Behaviors)* is the size of all the behaviors of agents composed the MAS. Computing the size of communication includes all the operation related to the communication process like messages processing, sending and receiving. Despite that this metric does not give the rate of interactions because the interaction's code can be repeated several times; but it can be used as an indicator to estimate the required effort to maintain the MAS. Increasing the result of this metric means that the behaviors of agents are probably most coupled. Thus, updating the code of an agent may require updating the code of the others.

(b) ***The Average of Exchanged Messages per Agent (AEMA):*** as we indicate previously, the *Rate of Interaction's Code* metric gives only partial information about the collective behavior of agents because the possible repeat of the sent messages. Therefore, we propose the *Average of Exchanged Messages per Agent (AEMA)* metric to estimate the real number of exchanged messages. Obviously, this metric is of dynamic nature. Using this metric we can estimate the required effort to understand the collective behavior of the MAS. It seems evident that the higher exchanging of messages can influence the complexity, because it means the difficult to study each agent alone.

(c) ***The Rate of the Environment's Accessibility (REA):*** the agents can be interacting indirectly by manipulating the environment's objects. In order to apply this kind of interaction, the environment's objects should be accessible. Hence,

$$REA = \frac{NPOA}{NAOA} \tag{5}$$

Where *NPOA* is the *Number of Public Objects' Attributes* and *NAOA* is the *Number of All Objects' Attributes*. This static metric is an indicator to the complexity of the MAS because of the existing of public attributes increases the agents coupling. For that reason, it becomes difficult to understand and maintain the collective behaviors of multi-agent systems.

The above metrics are proposed to assess the complexity of MAS that are implemented using the object-oriented paradigm. A multi-agent system is firstly software. Consequently, the proposed metrics to assess the conventional software (like the size in *Line of Code* metric) can be combined with our metrics to provide more information about the complexity of agent-based software. Especially, the proposed metrics for object-oriented software (like the depth of inheritance) can be used to calculate the complexity of the environment's objects.

Naturally, the comprehensive complexity of MAS is obtained by computing the average of all the above metrics. As appropriate, we can associate to each metric a weight which reflects its importance. We think that it is difficult to propose unanimously these weights. They are left to the appreciation of the users of the proposed metrics.

5 A Tool to Assess JADE Complexity

In order to assess the complexity of *JADE*-based MAS, we developed a tool that calculates automatically the above metrics. Figure 2 illustrates the architecture of the tool we developed.

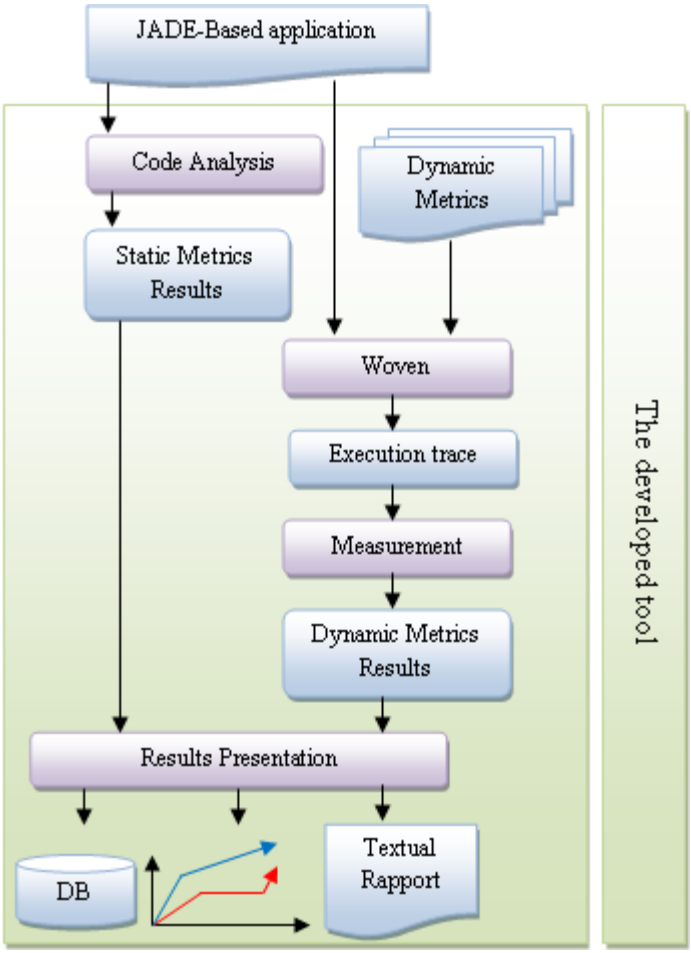

Fig. 2. Tool architecture

The measurement process passes through two phases. First, the MAS code should be analyzed to calculate the static metrics. Calculating the static metrics is based on the search of specific language construct of *JADE*. Second, *AspectJ* wove the developed dynamic metrics (as aspects) with the *JADE*-based software. As a result, we obtain the execution trace of the MAS. This latter is used to measure the different dynamic metrics. Both static and dynamic results will be presented in several forms

(textual rapport, graphical presentation and saved in data-base) for increasing the readability of the obtained results.

The following code is an example of aspect used to pick up an event during the execution of the MAS. This aspect is used in order to collect the information about the created agents. Thus, before creating an agent by the method *createNewAgent,* the aspect should be executed. In the same manner, the library of our dynamic metrics is composed of aspects to pick up the execution of the *main* method, the creation of objects, the updating of containers' size, the exchanging of messages and the scheduling of behaviors.

```
public aspect CreationOfNewAgent {
     pointcut CreationOfAgent():execution(*
            *createNewAgent(..));
     before() : CreationOfAgent(){
       // Saving information about the created agent
       }
}
```

The tool we developed has been validated using two case studies. We choose the *FSMAgent* and *ContractNet* interaction protocol examples available in [16]. The first example is to show the finite state-based behaviors. The agent executes a composite behavior which is composed of several simple behaviors. The composite behavior is presented using finite state machine. The following code presented the skeleton of the *FSMAgent.*

```
public class FSMAgent extends Agent {
  protected void setup() {
    FSMBehaviour fsm = new FSMBehaviour(this) {
      public int onEnd() {
           // the onEnd method
      }
    };
  // The composite behavior's description using the FSM
    addBehaviour(fsm);
  }
  private class NamePrinter extends OneShotBehaviour {
    //the code of the NamePrinter behavior
  }
  private class RandomGenerator extends NamePrinter {
    //the code of the RandomGenerator behavior
  }
}
```

The second example presents the implementation of the famous *Contract Net* interaction protocol. This example is composed of two classes of agents: the initiator and the participant. The *Initiator* starts its behavior by sending call for proposal to all the

participants. Then, it processes the responses provided by the participants in order to select the best proposal. After the reception of a *CFP* message, a *participant* agent formulates and sends its proposition if it wants to participate in the interaction protocol, or sends a refuse message in the other case. Depending on the response of the *Initiator*, the *participant* terminates its execution if it received reject message or executes the adequate action in order to provide the results to the *Initiator*. The limited size of this paper prevents us to present the code of the *ContractNet* example. However, the complete codes of the both examples are available in [16].

Table 1 gives the results of the collected static metrics of the two examples. We omit the *Heterogeneity of the Environment's Objects (HEO)* and *the Rate of the Environment's Accessibility (REA)* metrics because their values are zero (there is no objects in the environment of the presented examples). For instance, three behaviors are specified inner the agent *FSMAgent* which is presented by the *Behavioral Size of the Agent (BSA)* metric. Because one of these behaviors is a composite behavior which is presented with finite-state machine (with eight edges and six nodes), the *Average Complexity of Behaviors* metric becomes two. This metric corresponds, as explained above, the average *cyclomatic* number of all the behaviors of the agent.

Against by the *ContractNet* example is composed of two agents with single simple behavior in each one. Consequently, its *Average Complexity of Behaviors* metric becomes one. However, we can easily remark that the *Rate of Interaction's Code (RIC)* metric of this example is 0.46. Thus, almost half of the behaviors' code of the agents composed, this example is devoted to the communication.

Table 1. The results of the static metrics

The Metrics	FSMAgent	ContractNet
The Size of the Agent's Structure (SAS)	**06**	**0.5**
The Agent's Structure Granularity (ASG)	**01**	**0.5**
The Behavioral Size of the Agent (BSA)	**03**	**01**
The Average Complexity of Behaviors	**02**	**01**
The Heterogeneity of Agents (HA)	**01**	**02**
The Rate of Interaction's Code (RIC)	**00**	**0.46**

Thanks to the aspect-oriented programming, we can capture the events execution of the *JADE*-based applications in order to calculate the dynamic metrics. Hence, we use the tool we developed to execute the two above examples. We note that the *ContractNet* example is executed with single *Initiator* and three *participants*.

Table 2 gives the execution trace of the two above examples with the corresponding dynamic metrics: *The Average of Exchanged Messages per Agent (AEMA), The Size of the Agent's Population (SAP)* and *The Average Number of Scheduled Behaviors (ANSB)*. The *Dynamicity of the Agent's Structure (DAS)* metric is omitted because it has the zero value and it does not changed during the examples execution. Obviously, this value is justified by the lack of container in the agents' structures.

For limited size reasons, we will present only the *Average of Exchanged Messages per Agent (AEMA)* metric of the *ContractNet* example. This metric starts with the

zero value as is presented in Table 2. At the time 4672 ms the *Initiator* agent sent a *CFP* message to the three *participants* and the value of *AEMA* metric becomes 0.75. During the execution of the MAS this metric progresses until reaching the value 2.5 messages per agent at the time 4938 ms.

Table 2. The results of the dynamic metrics

	Time	Event	AEMA	SAP	ANSB
ContractNet	00	Starting the execution	**00**	**00**	**00**
	4297	Creation of the agents: *Participant_01, Participant_02* and *Participant_03*	**00**	**03**	**00**
	4375	Creation of the *Initiator* agent	**00**	**04**	**00**
	4422	Scheduling the behaviors of the agents: *Participant_01*and *Participant_02*	**00**	**04**	**0.5**
	4485	Scheduling the behavior of the agent *Participant_03*	**00**	**04**	**0.75**
	4672	The *Initiator* send *CFP* message to the three participants: *Participant_01, Participant_02* and *Participant_03*	**0.75**	**04**	**0.75**
	4719	Scheduling the behavior of the *Initiator*	**0.75**	**04**	**1**
	4766	The three participants (*Participant_01, Participant_02* and *Participant_03)* send their propositions	**1.5**	**04**	**1**
	4875	The *Initiator* agent send a response to the participants (*Participant_01, Participant_02* and *Participant_03)*	**2.25**	**04**	**1**
	4938	The agent *Participant_02* send *Inform* message to *Initiator* agent	**2.5**	**04**	**1**
FSMAgent	00	Starting the execution	**00**	**00**	**00**
	3407	Creation of the agent	**00**	**01**	**00**
	3422	Scheduling the behavior of the agent	**00**	**01**	**01**

We conclude after the above results analysis that the individual aspect is the complexity's source of the first example because it is composed of a single agent with a composite behavior. On the other side, the complexity of the second example is the result of its social aspect. The second example is composed of four agents that are instantiated from two different classes. Moreover, the interaction between these agents used most of the code. We can also remark that the results of the both kinds (static and dynamic) of metrics are not contradictory. In fact, both kinds of metrics associate the MAS complexity to the individual aspect in the first example and associate it to the social aspect in the second one. However, it will not strange to find inconsistency results between the static and dynamic metrics in some cases. Known that

the two kinds of metrics are complementary, we think that a deeply analysis of all the results may conduct us to identify the real reason of the complexity.

For increasing the readability of the obtained results, our tool gives also the collected metrics using graphical presentation. Figure 3 gives the evolution of *Average of Exchanged Messages per Agent (AEMA)* metric of the *ContractNet* example explained above.

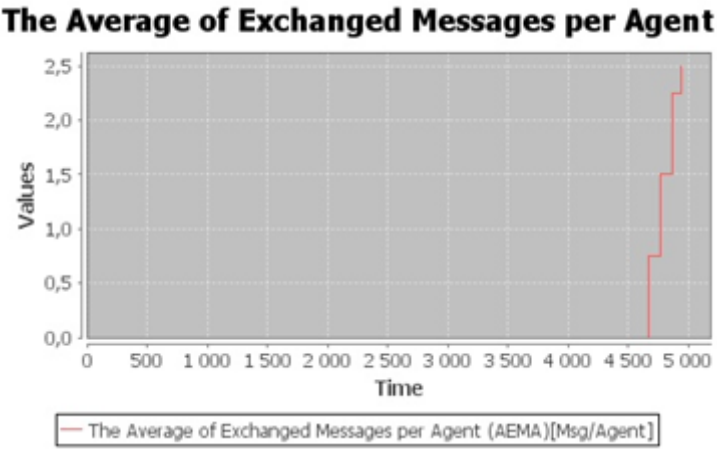

Fig. 3. The results of the *AEMA* metric of the *ContractNet* example

6 Conclusion

The software complexity is considered as an important indicator to estimate the required effort to understand and to maintain it. Recognizing its importance, this concept has been studied since the 1970s. However, the evolution of the software paradigms requires updating the complexity measurement according to the novelties provided by these paradigms. Nowadays, the multi-agent systems is one of the most applied software paradigms. Consequently, it seems very useful to measure the MAS complexity. This paper introduced a set of static and dynamic metrics in order to assess their complexity. These metrics are relative to a novel complexity model proposed for MAS. The proposed model allows simplifying and understanding the main characteristics that influence the complexity of MAS. The proposed metrics can be used, among others, as indicators to estimate the required effort to understand and maintain the implemented MAS. The proposed metrics can be calculated automatically for *JADE*-based applications using a tool we developed.

As future work directions, we plan to extend this work to support some specific kinds of multi-agent systems (e.g., mobile agent and adaptive agent). Moreover, we will target the complexity of multi-agent systems at early development stages (the specification and design of MAS).

References

1. Gutiérrez, T.N., Ciarletta, L., Chevrier, V.: Multi-Agent Simulation Based Control of Complex Systems. In: Lomuscio, A., Scerri, P., Bazzan, A., Huhns, M. (eds.) Proceedings of the 13th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2014), Paris, France, May 5-9 (2014)
2. IEEE: IEEE Standard Glossary of Software Engineering Terminology. IEEE Std 610.12-1990 (1990)
3. Tran-Cao, D., Abran, A., Lévesque, G.: Functional Complexity Measurement. In: International Workshop on Software Measurement (IWSM 2001), Montréal, Québec, Canada, August 28-29 (2001)
4. McCabe, T.J.: A Complexity Measure. IEEE Transactions on Software Engineering SE-2(4) (December 1976)
5. Tiwari, U., Kumar, S.: Cyclomatic Complexity Metric for Component Based Software. ACM SIGSOFT Software Engineering Notes 39(1), 1–6 (2014)
6. Gómez-Sanz, J.J., Gervais, M.P., Weiss, G.: A Survey on Agent-Oriented Oriented Soft-ware Engineering Research. In: Bergenti, F., Gervais, M.P., Zam-bonelli, F. (eds.) Methodologies and Software Engineering for Agent Systems - The Agent-Oriented Software Engineering Handbook, pp. 33–62. Springer US (2004)
7. Dumke, R., Mencke, S., Wille, C.: Quality Assurance of Agent-Based and self-Managed Systems. CRC Press (2010)
8. Dekhtyar, M., Dikovsky, A., Valiev, M.: Complexity of Multi-agent Systems Behavior. In: Flesca, S., Greco, S., Leone, N., Ianni, G. (eds.) JELIA 2002. LNCS (LNAI), vol. 2424, pp. 125–136. Springer, Heidelberg (2002)
9. Dziubiński, M., Verbrugge, R., Dunin-Kęplicz, B.: Complexity Issues in Multiagent Logics. Fundamenta Informaticae 75, 239–262 (2007)
10. Dospisil, J.: Code Complexity Metrics for Mobile Agents Implemented with Aspect/J™. In: Mařík, V., Müller, J.P., Pěchouček, M. (eds.) CEEMAS 2003. LNCS (LNAI), vol. 2691, pp. 647–657. Springer, Heidelberg (2003)
11. Klügl, F.: Measuring Complexity of Multi-Agent Simulations – An Attempt Using Metrics. In: Dastani, M., El Fallah Seghrouchni, A., Leite, J., Torroni, P. (eds.) LADS 2007. LNCS (LNAI), vol. 5118, pp. 123–138. Springer, Heidelberg (2008)
12. ISO: ISO/IEC 9126-1:2001 Software Engineering - Product Quality (2001)
13. Alonso, F., Fuertes, J.L., Martinez, L., Soza, H.: Towards a set of Measures for Evaluating Software Agent Autonomy. In: Proceedings of the Eighth Mexican International Conference on Artificial Intelligence (2009)
14. Tahir, A., Ahmad, R., Kasirun, K.M.: Maintainability Dynamic Metrics Data Collec-tion Based on Aspect-Oriented Technology. Malaysian Journal of Computer Science 23(3) (2010)
15. Kiczales, G., Lamping, J., Mendhekar, A., Maeda, C., Lopes, C.V., Loingtier, J.M., Irwin, J.: Aspect-Oriented Programming. In: Akşit, M., Matsuoka, S. (eds.) ECOOP 1997. LNCS, vol. 1241, pp. 220–242. Springer, Heidelberg (1997)
16. The Official JADE site web, `http://jade.tilab.com/`