Metrics for the Rationality of Jadex-Based Applications

Toufik MARIR
Research Laboratory on Computer Science's Complex Systems (ReLa(CS)2)
Department of Mathematics and Computer Science
University of Oum El Bouaghi
Oum El Bouaghi, Algeria
marir.toufik@yahoo.fr

Sihem OUBADI
Research Laboratory on Computer Science's Complex Systems (ReLa(CS)2)
Department of Mathematics and Computer Science
University of Oum El Bouaghi
Oum El Bouaghi, Algeria
sissima91@gmail.fr

Hadjer MALLEK
Department of Mathematics and Computer Science
University of Oum El Bouaghi
Oum El Bouaghi, Algeria
mallekhadjer@gmail.com

***Abstract*— Rationality is an important characteristic of agents. This characteristic reflects, among others, the reasoning quality of agents. Consequently, it seems important to measure it in order to control the quality of developed agents. In facts, measurement in the multi-agent systems is now an active area. Several metrics are proposed to measure different attributes of multi-agent systems like the autonomy and the reactivity. In this paper, we propose metrics to measure the rationality of JADEX-based applications. Jadex is a well-known platform to develop multi-agent systems. We present also a tool we developed to measure automatically the proposed metrics. The developed tool is based mainly on aspect-oriented programming.**

***Keywords*— Rationality, Metrics, Multi-Agent Systems, Jadex, Aspect-Oriented Programming, AspectJ.**

I. INTRODUCTION

Software development as an engineering activity should be based on using metrics and measurement in order to make rational decisions and to avoid subjective ones [1]. In fact, measurement consists of assignment of a value to an attribute of object or event [2]. Consequently, the measurement activity increases our ability of understanding and mastering the studied object. However, the measurement in software engineering is a hard activity because of the abstract nature of software [1]. Thus, a special attention is made by the domain community by proposing different metrics like the complexity [3] and the quality of software [4].

Since the first studies that targeted the measurement of software attributes [3], an immense evolution of software paradigms has been made. In fact, several software paradigms have been emerged in order to improve different software quality attributes. For example, object-oriented programming paradigm introduced several concepts (like classes, objects and inherence) to improve the modularity, the reusability and the maintainability of software. Naturally, this evolution of software paradigms has its effects on the software measurement. It is not suitable to apply specific metrics of a software paradigm to another one. Following the previous example, several metrics are proposed to take in consideration the specificities of object-oriented programming (as instance, new metrics are proposed to measure the modularity of object-oriented software [5]).

As one of relatively recent software paradigms, multi-agent systems require also their own metrics. Indeed, several works proposed specific metrics for the agent-based software [6]. These works aimed to measure common characteristics of software like complexity [7], quality [8] and architecture [9] as well as specific characteristics of multi-agent systems like autonomy [10], social ability [11] and pro-activity [12]. We think that proposing new metrics for multi-agent systems is a very promising area because of the current application fields of this software paradigm. Nowadays, the agent technology is applied in various domains from simplest to the most complicated ones and it is considered as an ideal paradigm to develop complex systems. Consequently, proposing specific metrics for multi-agent systems allows controlling the quality and the performance of the developed systems, regarding the requirement because unsuitable values of some characteristics can produce catastrophic results. For instance, despite that the autonomy of agents is a useful characteristic, an agent with a very high level of autonomy become dangerous because it can search to satisfy its own goals instead of satisfying the user requirements.

The rationality is an important agent's characteristic [13]. However, there is no study addressed its measurement. In our opinion the lack of specific metrics for this characteristic is due to considering it by the research community as an optional agent's characteristic compared to the fundamental characteristics (like autonomy, social ability and pro-activity). Almost the majority of works proposed in this field aiming to measure only the fundamental characteristics [10, 11, 12]. However, we think that this field is reached a maturity level allowing the study of more optional ones. In this paper we proposed some metrics to measure the rational of agents developed using Jadex platform [14]. Moreover, we developed a tool allows calculating these metrics automatically.

The remainder of this paper is organized as follows: section 2 is devoted to present related works. Then, we present in section 3 an overview about Jadex platform followed by presenting the proposed metrics and the developed tool. Finally, we present conclusion and some future works.

II. Related Work

Proposing new metrics for multi-agent systems is in full evolution [6]. In fact, we can distinguish mainly two kinds of metrics proposed in this field. Firstly, some works proposed new metrics for common characteristics of software with taking into account the specificities of multi-agent systems. The quality [8], the complexity [7] and the architecture [9] are examples of these characteristics. Marir et *al.* [7] proposed to measure the complexity of multi-agent systems. They proposed to measure this attribute in two levels: the agent and the system level. Then each level is measured by a set of metrics which addressed structural, behavioral, social and the interactional aspects. In addition, a model of the quality of multi-agent systems is proposed in [8] with several metrics that allow measuring the quality of such systems.

The architecture of multi-agent systems can also be measured by using a suite of metrics [9]. The architecture of such systems can be evaluated according to three attributes: modularity, extensibility and complexity. Each attribute is measured by using a set of metrics inspired from object-oriented programming.

Secondly, several works targeted the measure of specific attributes of multi-agent systems. On this context, works proposed by Alonso and *al.* [10, 11, 12] represented the most prominent. In a serial of works, the authors proposed metrics for the most important characteristics of multi-agent systems like autonomy, social ability and pro-activity. In each work, the targeted characteristic is devised into its main attributes and these attributes are measured using a set of metrics.

Despite works presented in this field, we can notice that several characteristics of multi-agent systems are omitted. In fact, we think that the research community targeted in the first times to the fundamental characteristics of such systems. In addition we believe that this software paradigm has reached a maturity level that allows addressing some characteristics considered as optional. In this work we addressed one of these characteristics called the rationality.

III. Jadex Platform

Jadex [14] is an extension of JADE platform to allow implementing of BDI (Belief-desire-Intention) model [15]. This later is a model conceived to explain the complex human behavior in a simple way. Thus, the BDI model is based on mental attitudes (beliefs, desires and intentions) that are modeled as possible world states. Hence, beliefs represent the state of the environment as is perceived by the agent. On the other hand, desires are the world states that the agent wants to reach. Intentions represent desires that the agent started to realize. Consequently, Jadex can be seen as a reasoning engine allows selecting the suitable plan based on the beliefs and the goals of the agent.

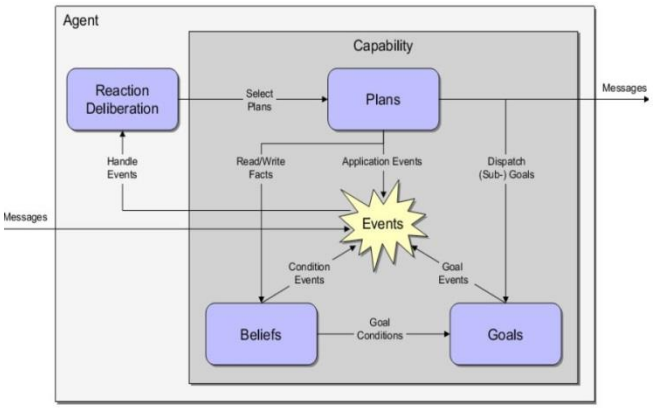

Fig. 1. The abstract architecture of Jadex platform [13].

Fig. 1 shows the abstract architecture of Jadex platform. It is composed of three main components [14]:

- **Beliefs**: in contrary to other BDI systems that represents beliefs using some kinds of first order logic or relational models (like Jason and JACK), Jadex used an object-oriented representation of beliefs. This representation is motivated by one of the objectives of Jadex project which is adoption of software engineering perspective for describing agents. Hence, beliefs are stored as simple facts (called belief) or as a set of facts (called belief set).
- **Goals**: are representing the desires of an agent. After creating a goal, this later can be in active, option or suspended state. Actually, an option is a new goal created and adopted by an agent (it was added to the goalbase of the agent). After goal deliberation, the agent can decide to realize this goal (activate it) or suspend it if its context is not valid. Moreover, we distinguish four types of goals: perform, achieve, query and maintain goals. Perform goals are one defined by a specific plan. Hence, an agent will reach this kind of goal when it executes the related plan. An achieve goal is a goal specified by the desired state of the world without specifying the related plan to reach this state. Is to agent to find the plan to reach this desired state. The query goals are similar to achieve goals but they are related to a state of the agent instead of a state of the world. Finally, in the case of a maintain goal, the agent will keep track a desired state, and he will execute an appropriate plan to re-establish this state to the desired one whenever needed.
- **Plans**: behaviors of agents are specified by plans. Each plan is composed of head and body. The head specifies goals related to the plan, events handled by it and preconditions for its execution. The body of a plan represents the actions to be executed when the plan selected.

In addition to these three main components, Jadex introduced the capabilities component that represents a grouping mechanism for the above elements (beliefs, goals and plans) to ensure the reusability attribute.

Jadex provides a hybrid approach to develop multi-agent systems. It is based on existing agent programming language (based on Java) with an XML extension to specify the static aspects of the agent. Hence, the static aspects of an agent are specified with Agent Definition File (ADF) and it includes description of beliefs, goals, head of plans and their initial values. Contrariwise, the dynamic aspects of an agent which represents the plan bodies are represented by Java language with an API that allows accessing to BDI facilities.

IV. The Proposed Metrics and the Developed Tool

Obviously, understanding a concept is the cornerstone of proposing metrics to evaluate it. However, formalizing unanimous definition of ambiguous concepts (like the rationality, the intelligence and the quality) is a hard task. One of reasons that made of the rationality concept an ambiguous one is its nature as one of concepts studied by various fields. In fact, the rationality is a concept which is studied in different fields from philosophy, sociology and economy to applied sciences like artificial intelligence [16]. Naturally, each field studied the concept from its own point of view. For example, a rational behavior is viewed in philosophy and sociology as a behavior emerged and influenced by reasoning process [17]. However, in economy the rationality is defined as using reasoning process to reach a goal within an optimized way [18]. Hence, the rationality in economy implies enhancing profits and minimizing costs.

The rationality of agents can be studied according to the artificial intelligence point of view because multi-agent systems are emerged from the interaction between artificial intelligence, distributed systems and software engineering. So, Russel and Norvig [16] define the rational agent as an agent which executes the adequate behavior. An adequate behavior is a behavior that makes the agent more close to his goal. Moreover, an *ideal* rational behavior is a rational one that minimizes the cost of reaching its goal [16].Also; a rational agent can always justify his decisions because they are the result of deliberation reasoning [19].

Choosing the adequate behavior should be based on the current state of the agent and its environment. In fact, this state is represented by the agent's beliefs. In addition, an agent should be able to update its beliefs according to the environment state. For this reason, a rational agent has perception abilities that allow to him to react to its environment.

Finally, we can conclude that the rationality of an agent is determined depending on the following characteristics:

- The existence of a reasoning process that justify the decision of the agent.
- A set of actions that allows the agent to choose the adequate action depending on the current situation.
- A set of beliefs allows representation of the agent's state and its environment to ensure a suitable decision.
- A minimum using of agent's resources.
- A successful execution of plans which results the reasoning process.

A. The Proposed Metrics of the Rationality of Jadex Agent

After specifying the main characteristics of a rational agent, we will present in this section some metrics to measure them. Thus, the rationality can be measured by using the following metrics:

- **The average size of plans:** as is already explained, the rationality is strongly dependent to the cost of reaching the goal. So, to reach its goal, an agent should execute plans. The average size of plans can give us an idea about some resources like the execution time and memory space. Knowing that the plans are coded as Java classes in Jadex platform, we can calculate the average size of plans as the average number of instructions used to code all the plans.
- **The average number of plans:** a rational agent is the one that selects the suitable plan to reach its goal. In fact, choosing a plan is a key factor for rational agent because we can not consider an agent as a rational one if he has not the choice even if the executed plan is adequate. Consequently, we propose to calculate the number of plans compared to the number of goals as a measure of rationality. While the plans are coded as a Java classes, in Jadex platform the goals are described in the description of the agent (ADF).
- **The ratio of beliefs per goal:** beliefs represent a fundamental element to ensure reasoning process. This later is a key factor of a rational behavior. In the other hand, beliefs are one of elements that are using memory resources. Consequently, we calculate the number of beliefs described in the agent compared to the number of its goals as a measure of the rationality.
- **The number of executed plans:** an agent can use various resources dependently to the application. As a result, it is hard difficult to propose generic metrics that specify all the used resources. However, we can remark that using resources is closely dependent to executed plans. Hence, we propose the number of executed plans as a measure of rationality that reflects using of resources. We can calculate the number of executed plans by calculating the number of executed ***body()*** method.
- **The average execution time of plans:** despite that we proposed to measure the rationality using

the size of plans, we think that this measure is not enough. In fact, the size of plans considers that *all* the instructions are equal during the execution. Practically, the instructions have not the same execution time because the existence of loops and alternative instructions. Hence, we propose to measure the average execution time of plans as an indicator of used resources. This metric is calculated by calculating the execution time of ***body()*** method compared to number of ***body()*** execution.

- **The number of accessing to beliefs:** access to beliefs by reading or updating them is an indicator of reasoning. In fat, reasoning process consists of using the current state of beliefs to generate new beliefs or actions by means of a set of plans. Moreover, updating the beliefs' state can also make because updating the environment state. It is an indicator of using updated information about the environment during the reasoning process which enhances the rationality. Access to the beliefs in Jadex platform can be made by two functions: ***setFact()*** and ***getFact()***. Thus, we calculate the number of execution of these functions as a measure of the rationality.
- **The ratio of achieved plans:** we explained above that the rationality is related to the plans success. Hence, we propose the ratio of achieved plans as a measure of the rationality. This metric is calculated as the number of achieved plans compared to the number of executed plans. An achieved plan in Jadex Platform is recognized by the execution of the method ***passed().***

B. The Developed Tool and a Case Study

In order to calculate automatically the proposed metrics, a tool has been developed as a form part of this work. This tool allows the user to measure static and dynamic metrics. Static metrics (like the average size of plans) are calculated from the code of the application under the evaluation. In the other hand, dynamic metrics (like the ratio of achieved plans) require the execution of the application.

Dynamic metrics are calculated using aspect-oriented programming [20].This paradigm is relatively a new one which is proposed to enhance the modularity of software by separating crosscutting concerns from core concerns. The basic element in aspect-oriented programming is the aspect. An aspect includes several entities like pointcuts, Join points and advices. A join point represents a well-defined point in the execution of a program in which an aspect will be integrated. Contrariwise, a pointcut allows the specification of join points in the aspect. It represents a method execution, call of a method or update of an attribute. Finally, an advice is a specification of code that will be executed by the aspect. Marir et *al.* [21] demonstrated the advantages of using aspect-oriented programming to measure dynamic metric of multi-agent systems. In fact, it provides the simplicity, reusability and extensibility [21]. Fig. 2 shows the principle of this software paradigm.

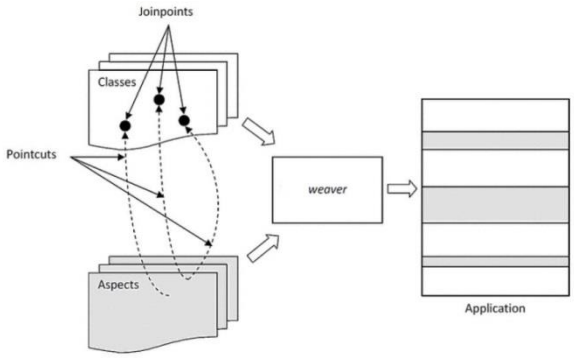

Fig. 2. The principle of aspect-oriented programming.

The static metrics are calculated by analyzing the code of the application. Knowing that a Jadex-based application is composed of two files (one XML file to describe the agent (ADF) and the another a Java file to describe the plans), the static metrics consisted in analyzing both of them. Fig. 3 presents the architecture of the developed tools. We used an open source API (called JDOM) [22] to represent and manipulate the ADF files in order to calculate the static metrics. Moreover, the tool is composed of a library of aspects developed using *AspectJ* [23]. These aspects allow picking out the execution of essential events related to the proposed metrics (like the execution of ***body()*** and ***passed()*** methods). Fig. 4 presents an example of aspects that is used to pick out the deferent methods relevant to our metrics.

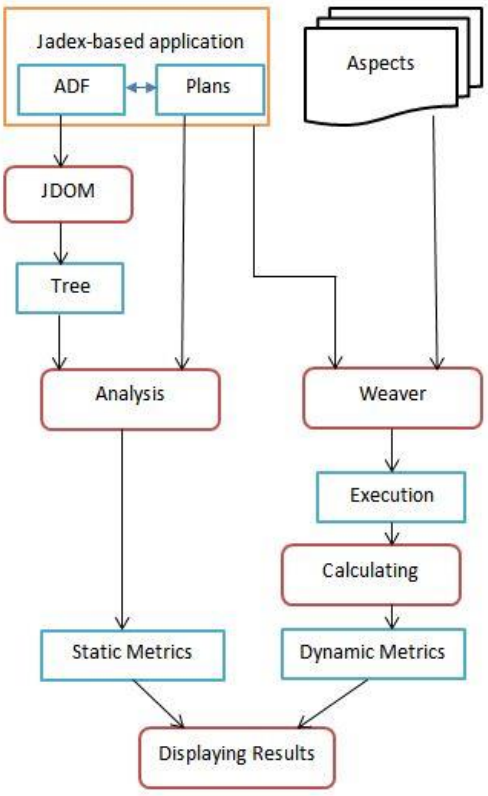

Fig. 3. The abstract architecture of the developed tool.

```
package aspects;
import metriques.Evenements;
public aspect CaptMethode {
    List_event l = new List_event();
    pointcut P1() : call(boolean IGoal.isSucceeded());
    //call(public * Line.* (..))
    pointcut P2(): call (* IBelief.getFact(..));
    pointcut P22(): call (* IBelief.setFact(..));
    pointcut P3(): call (* IBeliefSet.add*(..));
    pointcut P33(): call (* IBeliefSet.remove*(..));
    pointcut P5(): call (* IGoalbase.createGoal(..));
    pointcut P6(): call (void IGoal.drop());
    after() :P1(){
        System.out.println("***** Je suis l'aspect after isSucceeded()");
        long x = System.currentTimeMillis();
        int typeEvent = 4; // 4 = isSucceeded
        Evenements e1 = new Evenements(typeEvent,x);
        l.ajouter_event(e1);
        System.out.println("The Time is : " + x);
        e1.afficherEvent();

    }
    after() : P6(){
```

Fig. 4. An example of used aspects.

In this paper, we applied the proposed metrics to Mars World application [24]. This later is composed of three agents that explore the environment (Mars World) for ore resources and bring as much ore as possible to the agents' homebase. When the mission time has expired the agents have to abort their current actions and return to the homebase. The first agent is the sentry agent. It has the task to find ore resources inspect them if they can be exploited. Therefore the sentry agent has the greatest vision of all agent types. To find the ore resources more quickly all other agents report to the sentry about resources they explored. The second one (Production Agent) is called to a target from a sentry to produce as much ore as the capacity of the resource permits. When finished the agent calls for carry agents to bring the ore to the homebase. Finally, the third agent (called Carry Agent) has the task to bring ore from targets to the homebase. It is called by the production agent. Fig. 5 shows the interface of this application.

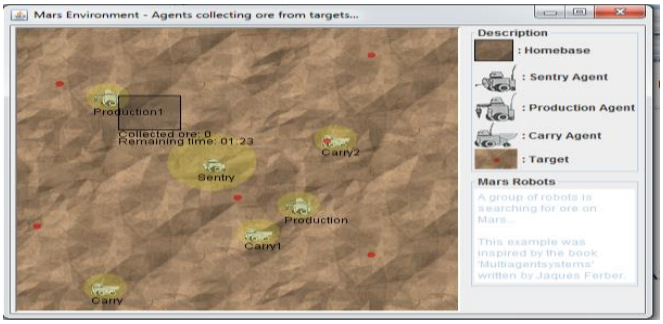

Fig. 5. The interface of Mars World application.

After the execution of this application using our tool, we can show the results of the different metrics. Fig. 6 shows the results of the average size of plans of the three agents. As example, the average size of plans of the sentry agent is 24 because it has three plans with sizes equal to 18, 39 and 15 instructions. Moreover, the developed tool allows also calculating of dynamic metrics. Fig. 7 presents the number of accessing to beliefs metric for the agents of Mars World. We can remark that this metric started from zero for all the agents and progress continuously according to the application execution.

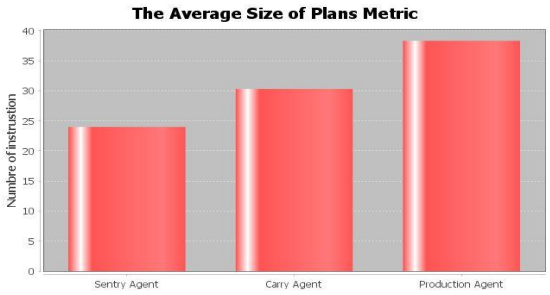

Fig. 6. The results of the average size of plans metric.

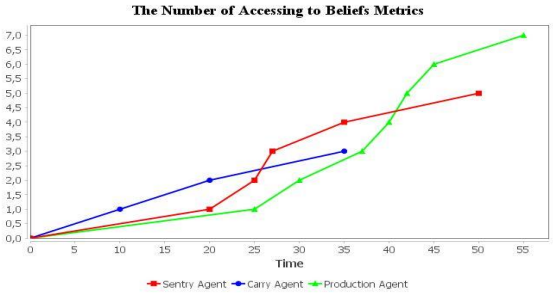

Fig. 7. The results of the number of accessing to beliefs metric.

V. CONCLUSION

Measurement is a vital task in software engineering. It allows controlling the quality of software and their process of developing. In fact, many metrics are proposed to evaluate the different factors of software such as quality and complexity. Moreover, proposing of metrics is in full evolution because of the continuous emergence of new paradigms. For example, multi-agent systems are relatively a new software paradigm which drew attention of the research community by proposing specific metrics for its different attributes as autonomy, reactivity and pro-activity. Despite that rationality is an important characteristic of intelligent agents, proposing metrics for it is omitted. For this reason, we proposed in this paper some metrics that allow measuring the rationality of agent. Naturally, proposing these metrics is passed by analyzing this concept. Moreover, we presented in this paper a developed tool as part of this project that allows calculating the proposed metrics. This tool is developed using aspect-oriented programming to assess the dynamic metric, as it analysis the code of the application (XML and Java files) to assess the static metrics.

We propose as a future work to enhance this work to assess other attributes like the flexibility and the intelligence. In addition, we think that the proposed metrics will be more beneficial if they are generic. So, we propose to generalize the proposed metrics to be suitable with other multi-agent platforms.

REFERENCES

[1] Roger S. Pressman and Bruce R. Maxim, Software Engineering: A Practitioner's Approach, 6th Edition, McGraw-Hill Higher Education, 2005.

[2] ISO, ISO/IEC 9126-1 Software Engineering – Product Quality – Part 1: Quality Model, International Organization for Standardization, Geneva, Switzerland, 2001.

[3] McCabe, T.J.: A Complexity Measure. IEEE Transactions on Software Engineering SE-2(4) (December 1976).

[4] McCall, J.A., Richards, P.K. and Walters, G.F. (1977) Factors in Software Quality, Vol. 1, ADA 049014, National Technical Information Service, Springfield, VA.

[5] Asadullah, Allahbaksh Mohammedali, and Basava Raju Muddu. "Assessing modularity of a program written in object oriented language." U.S. Patent No. 9,760,369. 12 Sep. 2017.

[6] Dumke, R., Mencke, S. and Wille, C., Quality Assurance of Agent-Based and Self-Managed Systems, CRC Press, 2010.

[7] Marir T, Mokhati F, Bouchelaghem-Seridi H, Tamrabet Z. Complexity measurement of multi-agent systems. InGerman Conference on Multiagent System Technologies 2014 Sep 23 (pp. 188-201). Springer, Cham.

[8] Marir T, Mokhati F, Bouchlaghem-Seridi H, Acid Y, Bouzid M. QM4MAS: a quality model for multi-agent systems. International Journal of Computer Applications in Technology. 2016;54(4):297-310.

[9] García-Magariño I, Cossentino M, Seidita V. A metrics suite for evaluating agent-oriented architectures. InProceedings of the 2010 ACM Symposium on Applied Computing 2010 Mar 22 (pp. 912-919). ACM.

[10] Alonso F, Fuertes JL, Martinez L, Soza H. Towards a set of measures for evaluating software agent autonomy. InArtificial Intelligence, 2009. MICAI 2009. Eighth Mexican International Conference on 2009 Nov 9 (pp. 73-78). IEEE.

[11] Alonso F, Fuertes JL, Martínez L, Soza H. Measuring the Social Ability of Software Agents. InSoftware Engineering Research, Management and Applications, 2008. SERA'08. Sixth International Conference on 2008 Aug 20 (pp. 3-10). IEEE.

[12] Alonso F, Fuertes JL, Martínez L, Soza H. Measures for Evaluating the Software Agent Pro-Activity. InComputer and Information Sciences 2011 (pp. 61-64). Springer, Dordrecht.

[13] Verschure, Paul FMJ, and Philipp Althaus. "A real-world rational agent: unifying old and new AI." Cognitive science 27.4 (2003): 561-590.

[14] Pokahr A, Braubach L, Lamersdorf W. Jadex: A BDI reasoning engine. InMulti-agent programming 2005 (pp. 149-174). Springer, Boston, MA.

[15] Rao AS, Georgeff MP. Modeling rational agents within a BDI-architecture. KR. 1991 Apr 22;91:473-84.

[16] Russell, Stuart J., and Peter Norvig. Artificial intelligence: a modern approach. Malaysia; Pearson Education Limited,, 2016.

[17] https://en.oxforddictionaries.com/definition/rationality

[18] https://www.ecnmy.org/learn/you/choices-behavior/what-is-rationality/

[19] N. Kabachi, Modélisation et Apprentissage de la Prise de Décision dans les Organisations Productives: Approche Multi-Agents, Thèse de doctorat de l'Université Jean Monnet et de l'Ecole Nationale Supérieure des Mines de Saint-Etienne, 1999.

[20] Kiczales G, Lamping J, Mendhekar A, Maeda C, Lopes C, Loingtier JM, Irwin J. Aspect-oriented programming. InEuropean conference on object-oriented programming 1997 Jun 9 (pp. 220-242). Springer, Berlin, Heidelberg.

[21] Marir T, Mokhati F, Bouchelaghem-Seridi H, Benaissa B. Dynamic Metrics for Multi-agent Systems Using Aspect-Oriented Programming. InGerman Conference on Multiagent System Technologies 2016 Sep 27 (pp. 58-72). Springer, Cham.

[22] http://www.jdom.org/

[23] Laddad R. AspectJ in action. Manning Publications; 2009.

[24] https://sourceforge.net/projects/jadex/files/jadex/2.4/