Formal Specification Method for Gaia Methodology

Laith Obidat
Software Engineering Department
Isra University
Amman, Jordan

ABSTRACT
the software development life cycle process is in continuity and expansion. So that may occur and leads to complex software systems which need to have technologies to realize and understand of large-scale and complex software systems. One of these methods helps to achieve this goal which called agent technology that is compatible and deals with complex software systems characterized by a high degree of distribution. So, a collection of software agents collaborates or competes with each other to achieve single or collective task led to multi-agent. In other hands, the correctness of specifications in the first phase of software development life cycle it is the main factor that led to successful project. The problem is that the specifications are written by informal method may occur misunderstanding and ambiguity. In this research, I will give a solution by doing formal specifications method for GAIA methodology which is one of the methods used in multi-agent systems. This step will enhance the specification based-on Object Constraints Language (OCL).

Keywords
Agent-Oriented Methodology, Multi-Agent System, GAIA Multi-Agent Method, Formal Method, Formal Specification

1. INTRODUCTION
In recent years, agent-based systems have become more popular in academic and industry environment. The agent-oriented model it is an extension to the Object-Oriented (OO) model [1]. Both deal with the principle of encapsulation and information hidden as well as recognize the importance of interaction. But agents differ from objects in many issues such as when we developing Multi-Agent Systems (MAS), the classical analysis and design approaches are poorly. Moreover, objects are negative when interacts with external factors, as well as the interactions between agents are characterized by independence whereas object interactions are class dependent [2].

In another side Agent-Oriented Software Engineering (AOSE) "is one of the most recent contributions in the field of software engineering to face the complexity of information communication technology such as agent-based system for monitoring and diagnosing faults in nuclear power plants" [3]. AOSE provide a variety of conceptual frameworks, notations, techniques and hence provide a platform supports the generalization, dynamic, and autonomous which helps Introducing robustness and easy to use software methodologies to meet challenges and achieve the goals [4].

Multi-Agent Systems (MAS) are autonomous systems that interacting together or interact with themselves called the agent. The agent has some of the characteristics that have characterized as follows: autonomy– it works without the direct involvement of human or others; social ability– agents interact with other agents through agent communication language and set rules called an interaction protocol [5].

Multi-agent systems are a new subpart of computer science has been studied only start since about 1980 and it has been recognized widely since about the mid-1990s. In addition, after the 1990s, the acquisition of world's attention has begun, because of the belief that the agents are a suitable software approach in huge open distributed systems- such as the internet [6].

In this paper, the Gaia methodology used for the development of multi-agent systems is proposed to help an analyst to apply a systematic method from a statement of requirements to the design in detailed. Gaia methodology that has two levels abstract and concrete; abstract level is used during analysis to conceptualize system such as role, permission, responsibility and so on. Concrete level is used within design phase. Gaia methodology doesn't explicitly deal with requirement capturing, implementation issues and does not provide constructs for the formal verification [7, 8], so we will propose a method to improve the Gaia methodology by providing formal specification method to get a better practices and results.

The agent and multi-agent systems are being described as a new approach in software engineering for the research field to face the complexity of information communication technology. Specification one of the factors which impact on analysis and design phases to reach and achieve best reliable and accurate practices in Gaia methodology. Then we will need to provide a formal specification method for GAIA methodology to achieve this goal that leads to enhancing in Gaia methodology and improve inputs into analysis phase, which positively reflected in the analysis and design phases.

This paper is arranged as follows: the second section includes brief relevant background information over the topics of Gaia methodology and object constraints language and presents related works. Then, third section presents proposed solution and the proof. But last section presents discussions and conclusion with future work.

2. BACKGROUND
In this section, we will show you some of the concepts that you must understand in order to reach a conceptual perception about some of the topics that will lead to understanding the integrated environment for this paper are as follows.

2.1 Definition of Multi-Agent System
A Multi-Agent System (MAS) is a computerized system consists of multiple autonomous agents, who can interact using an interaction protocol and perform actions within a common environment to solve difficult complex problems [9]. Another definition of a Multi-Agent System (MAS), systems are interrelated through processes work together in parallel with the synchronization between them to solve complex problems to achieve and perform a specific goal [8].

2.2 Agent Characteristics

Agent in Multi-Agent System (MAS) has many of the properties are as follows [9, 10]:

- Autonomy: the agent is not controlled directly by human or others.
- Local views: the agent does not have a complete public view of the system.
- Decentralization: No agent has complete control over the system.
- Reactivity: agents perceive their environment and respond in an appropriate time to changes that occur in it.
- Pro-activity: agents have the ability to present goal-directed behavior by taking the initiative.

2.3 Agent Environment Characteristics

They are many properties for agent environment as follows [9]:

- Accessible vs. Inaccessible.
- Static vs. dynamic.
- Open vs. closed.

2.4 Gaia Methodology

Gaia Methodology is the first complete methodology suggested to show the process of developing a Multi-Agent System (MAS). The scope of the methodology incorporates the analysis and design stages and rejects both gatherings of specification and implementation. It is applied after gathering and specified of the requirements and applicable to a range of multi-agent systems [11].

It was released many versions that concerning with the Gaia methodology in order to improve and those versions: Gaia v2, ROADMAP and extending Gaia with AUML. In general, the Gaia process comprises in constructing a series of models, as shown in Figure 1. These models are aimed to describe both macro-level (societal) and micro-level (agent) aspects of systems [12].

These models are distributed into the analysis and design phases are as follows: in the analysis phase constructs both role model and interaction model. These two models represent the abstract level, and used as input to design phase, but in design phase that is called concrete level is consist of three models: an agent model, a service model, and an acquaintance model are defined to build a full design specification of the Multi-Agent System (MAS) to be used for implementation phase that is not supported by Gaia methodology [12, 13].

2.5 Object Constraint Language (OCL)

We need language to help in the specification, A UML diagram, such as class diagram is not enough to describe all relevant information about a specification. Also, we need to specify and describe extra constraints don't capable and applicable using UML diagram such as constraints that described only by natural language. Object Constraints Language is used to specify the constraints on object-oriented systems. OCL is a standardized and it is used to converts the semi-formal specification to formal specification without side effects and no control flow; this means that the model cannot change during the validation process. In addition, it is supporting object concepts. OCL is not a programming language but it has a formal mathematical semantics and depends on set theory and predicate logic [14].

2.6 Combining OCL with UML

UML models will be weak and inaccurate without OCL expressions. Also, without UML models, the OCL expressions are not connected with diagram elements. But, when combined the UML model and the constraints, it achieves the fully specify the model. We can connect OCL expression with a UML model using the basic types such as String, Integer, Real, and Boolean, also, we can link between OCL expression and class from the UML model and their attributes [15].

3. RELATED WORK

In this section, we will study the related work in the area of merged the object constraint language (OCL) with class diagrams and validate it; also, we will study integrated Gaia methodology with UML/OCL class diagrams.

The study in [16], it depicts the importance of integrating both the OCL with UML models to get a precise description of some aspects of software models. In addition to this research, introduces recommendation to use the OCL because the OCL is supported by tools, accepted language, consistency and expressivity.

The research in [17], it was intended to introduce a formal specification through analyzing functional and non-functional properties in stepwise enhancement process from abstract specification level to concrete specification level to achieve to formal verification for these specifications. Where it was merged Gaia methodology with finite state process (FSP) and as a result, it has been providing the formal specification of their system and validate these specifications by analyzer called Labeled Transition System Analyzer (LTSA) which was proposed and modeled by Magee and Kramer, as a result, it has been verified both safety and liveness property in Gaia methodology.

The research in [18], the main goal of this research is to present a method for fully automatic, decidable and expressive verification of UML/OCL class diagrams. In this method is used constraint programming approach as formalism, so that was developed a systematic procedure for the transformation of UML class diagram annotated with OCL constraints through a constraint satisfaction problem (CSP), where it was pre-definition set of the correct properties about the UML/OCL diagram, such as satisfiable of the model, liveliness of a class and redundancy of a constraint. Also checked the result using a graphical front-end tool called UMLtoCSP which is developed to improve the usability of them verification method. Using this tool (UML to CSP) begins by introducing a UML class diagram in an XMI and a text file contains the OCL constraints as an input. On the opposite side, the output of the (UMLtoCSP) tool is a UML object diagram that leads to prove the property. The (UMLtoCSP) tool is easy to use because the user does not need to know about prolog or CSPs. So the input and output notations are usable to designer. The whole verification stage is fully automated and hidden from the user, so the following the hidden formal methods is to enhance the usability of the (UMLtoCSP) tool and its results. The preliminary results of this approach offered through the workshop and have taken those preliminary results to add a richer description of the method and tool, an enhance UML/OCL to CSP mapping strategy, an evaluation of the problem complexity and efficiency results with more detailed comparison with relevant approaches. The limitations of this approach that is used the (UMLtoCSP) tool is the lack of the translation from UML/OCL into CSPs because it does not provide support for

all the features described in the complex standards for UML and OCL, such as multiple inheritances and recursive OCL queries. Also, the encoding in the CSP can reduce the performance of the verification process.

Another related approach is the USE tool [19]; this tool is more focused on validation process than verification process. USE allows to check of formal properties and allows us to review the consisting of UML models and constraints. Also, USE allows generating snapshots automatically, but it permits to generate finite snapshots. In this tool (USE) the user identifies a list of properties for the instances and their numbers. As a result, the USE tool is generating and validates them. Finally, the limitation of this approach is the USE tool used for validation only because it isn't focused on verification.

Another related approach is the Higher-Order Logic (HOL-OCL) [20]; this system is developed to introduce formal and interactive proof environment for UML and OCL specifications which is involved into Isabelle/HOL. HOL-OCL is depends on the store for UML/OCL models that are called Su4sml and Isabelle/HOL. UML/OCL specification is developed by ArgoUml tool. Then, they import them into HOL-OCL by Su4sml repository. The core theorems needed for verification and formal semantics that related with the OCL. The integrated of all parts in this approach will provide a proof environment for specification that based on UML class models with OCL constraints. Finally, the limitation of this system is undecided until now.

Another related work in [21]; adopts the reasoning process on UML class diagram based on the CASE tools are available, that provide tools to create and modify multiple UML diagrams easily. These tools have the efficiency which allows discovering related formal properties. Some of these tools such as Rational Rose, Together, Poseidom and ArgoUml, are used to support the designer with a GUI which has a rich user-friendly graphical environment for accessing to several UML class diagrams. But in this study, the knowledge representation and reasoning procedures are developed by Description Logics (DLs). Description logics are used to represent of knowledge in classes and relationship between them. The contributions resulting from this study are included to prove the reasoning on class diagrams is EXPTIME-hard by presenting a polynomial reduction from reasoning in description logics. Another contribution is establishing EXPTIME-membership of reasoning on class diagrams without using OCL constraints. The last rich contribution provides polynomial encoding of class diagrams in the description logic that called ALCQI. Finally, the limitation of this study is not support OCL.

Another related work in [22]; it adopts the model-driven development (MDD) approach to present a new automatic method for verifying UML models that extended with OCL constraints to provide correctness then by operation contracts. In this study, the different in this study from the most others studies have focused on the verification of both dynamic aspects and more focus on static aspects, but in the most other studies that are only focused on the static aspects. Also, in this study the automatic translation process is done into constraint satisfaction problem (CSP) by (UML to CSP) tool. Successive steps in this study are as follows: declarative operations in OCL, list of correctness properties based on pre- and post-condition, some of these properties such as applicability, redundant precondition, weak excitability, strong excitability, correctness preserving, and immutability. Finally, the last step is verifying operation with constraint programming.

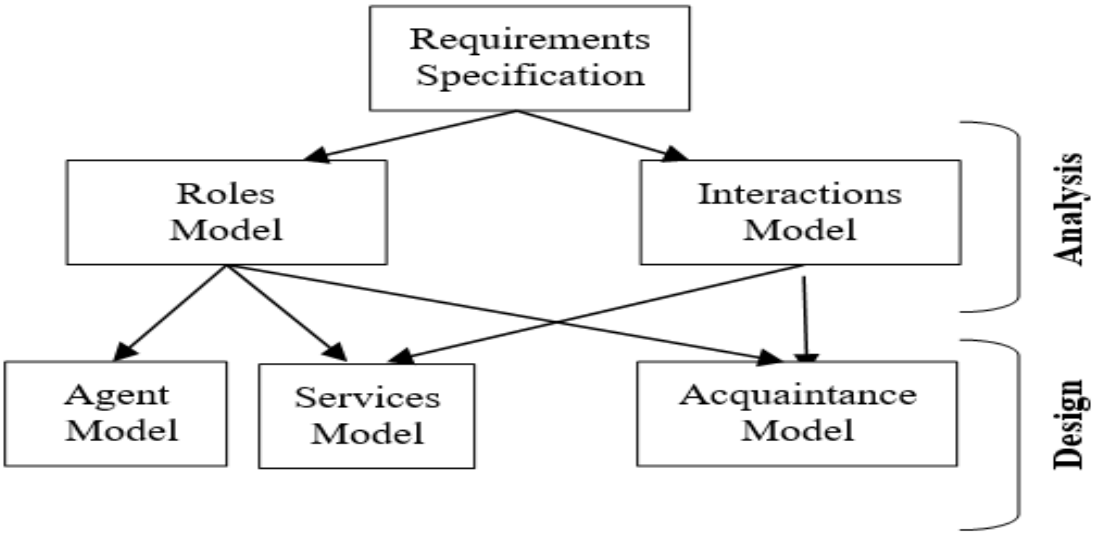

Fig 1: Models in Gaia Methodology

4. PROPOSED SOLUTION AND THE PROOF

The heading of a section should be in Times New Roman 12-point bold in all-capitals flush left with an additional 6-points of white space above the section head. Sections and subsequent sub- sections should be numbered and flush left. For a section head and a subsection head together (such as Section 3 and subsection 3.1), use no additional space above the subsection head.

4.1 Proposed Solution

As noted in studies related to Gaia methodology, the lack contained in this methodology didn't deal with the system requirements because it's scope of application only includes both the analysis and design phases. Also, it ignores both the

implementation and the system requirements phases (see Figure 1: Model of Gaia) was necessary to propose a solution that leads to improved Gaia methodology. After reviewing the many aspects that can improve the Gaia methodology, whether in the implementation phase, the system requirements phase, within the two phases of analysis and design which are contained in the Gaia methodology or otherwise. We have reached a solution to regarding of the system requirements that was ignored by Gaia methodology, leading to the improvement of Gaia methodology by providing a formal specification. After searching of the part of the formal specification for the system, we have reached to two ways to provide a formal specification of the system are: The Z language and Object Constraints Languages (OCL), and we have chosen the OCL, rather than the Z language because of the features offered by OCL previously mentioned (see section 2.1.6). In addition, the limitation exists in the Z language. After doing an analytic study about both options Z language and OCL, we can say that the OCL has syntax more familiar to the users. That leads to the description OCL more readable. In addition to the OCL, does not require prior knowledge of mathematics. Thus, we agree on the choice of the OCL to be used in the description of the models instead of the Z language. The OCL is supported by tools but the Z language not supported by tools [26]. After choosing the OCL, we chose the unified modeling language models (Class Diagrams) to be combined with the OCL to introduce a formal specification of the system into verification task. After obtaining the formal specifications we combine these specifications as input to the analysis phase in the Gaia and evaluate the combine process by JADE framework (see Figure 2).

4.2 Proved the Proposed Solution: E-Travel System Case Study

As shown in the solution proposed (see Figure 2), which begins in the introduction of system specifications, and then, get into the modeling environment by providing class diagrams of the system using the CASE tools that deal and address strongly the class diagrams, in addition to support for the Object Constraints Language (OCL) which help top make a formal specification [14]. After studying the characteristics and features for several CASE Tools, which support the Unified Modeling Language (UML), and supporting the Object Constraints Language (OCL). We have come to the selection ArgoUml tool and the reason has gone back to some of the characteristics which are characterized by ArgoUml

tool. after choosing ArgoUml tool, we have combined both the object language constraints with class diagrams of the system in order to help us to make a formal specification of the system to be presented in the form of inputs in the Gaia methodology. As a result, this method will save the time and effort for both the analyst and designer. In addition, it achieves a better practice in the Gaia methodology and relying on well-known idea. The process that provides a good and a formal system specification are going to get the best results at all stages that follow the stage of submission of the specification [15].

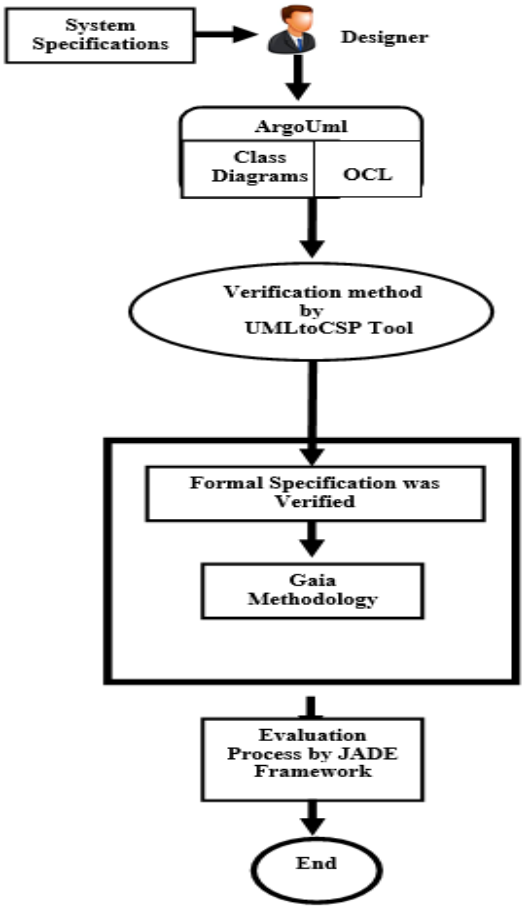

Fig 2: Overall picture of the process in our solution

4.3 E-Travel System as Case Study

When applying the E-travel system case study, that contains Personal Travel Assistant agent PTA Agent and Coordinator Agent which representing a real scenario of the tourist zone. Mentioned scenario is one where someone wants to find out information such as price and availability of the different elements related to travel, such as hotels and transportation. The real scenario was derived from E-Travel case study. From the beginning when the user orders something. For instance, the user needs to look for place to stay. The confined in New York from 20/03/2014 to 25/03/2014, he should write in the search places and submits it. Then, we explain what is going on the multi-agent applications, later the user presses the "Search" button. Then again, the user has compassed his order, this order will be arrived and reserved through Personal Travel Assistant Agent to require and organize it. The connection within the user and the PTA is done by jade Gateway class. A servlet response to user's input, that is crossed through Gateway Agent among Black Board object. The Gateway Agent read out the recipient and message tenor and later on sends and exports the message to the PTA.

4.4 Using UMLtoCSP Tool

After obtaining class diagrams, the system extended with the Object Constraints Language (OCL). It is essential that the process of verifying the class diagrams. After reviewing several ways that was proven earlier, we had chosen verification method that has been used in [28], which depends on the constraint programming through use and adaption the UMLtoCSP tool of power.

4.5 Applying JADE Framework

After obtaining formal specifications have been verified to be merged with Gaia methodology to improve them, evaluate and study the new changes in the workflow of operations and communications between agents and to clarify aspects of our optimization and discussed them. The evaluation process will be through the use of the Java Agent Development Framework (JADE). We concern with the sniffer tool in the JADE framework that views the ACL messages which forwarded through agents. Next paragraph explains the points of operations and steps.

The sniffer Agent declares and provides us to exam and observes at real period and time, how the agents connect through each other to organize and determine the complication as it's explained in figure 3.

At figure 3 that explained, During the TravelAgent arrives an order message, it looks for an agent offering the "coordinator" service through an order messages and receives the reply like report connected action. In addition, it forwards the order to coordinator Agent who is going to order the Agent located or situated in required categories. The moment that it finds, the coordinator is going to forward orders to agent that waiting to their affirmations. When all of them are accepts, it's going to forward a conformation message for the TravelAgent saying that end.

After execute sniffer agent tool by add classes into JADE framework and validate the sequence diagrams, the JADE is still not support OCL perfectly since it when we do the sniffer things still the OCL does not appears, so we do it manually to show that OCL can go through the sequence diagrams (see figure 4 add OCL into sniffer manually).

After becoming extremely take formal specification that was validated it through the object constraint language, we have to note that the performance of the Gaia methodology became better than before in the case of its application within the same environment with the absence of a formal specification.

5. CONCLUSION

In recent years, agent-based systems have become more popular in academic and industry environment. AOSE provide a variety of conceptual frameworks, notations, techniques and hence provide a platform supports the generalization, dynamic, and autonomous which helps introducing robust and easy to use software methodologies to meet challenges and achieve the goals. We have chosen the Gaia methodology to apply our solution for several reasons as follows: because Gaia methodology was used extensively worldwide.

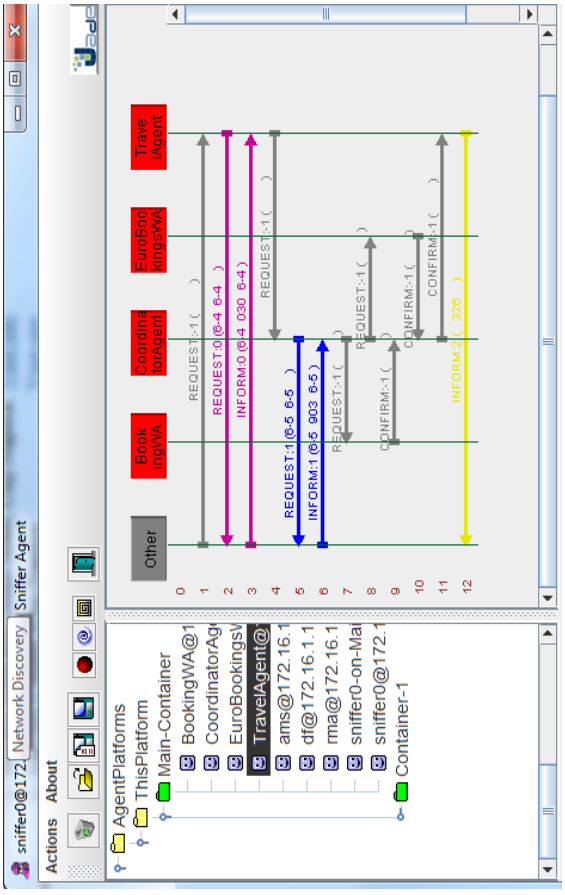

Fig 3: Communication between agents by sniffer

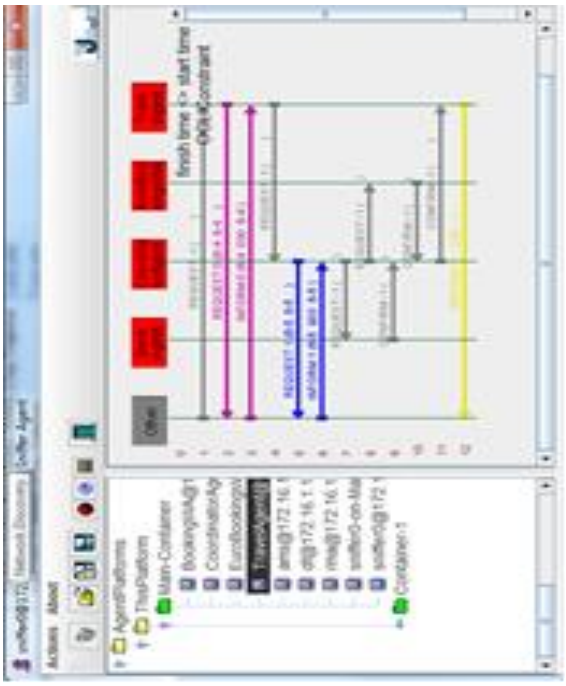

Fig 4: Add OCL constraints into Sniffer manually

In addition to the Gaia methodology, it applied to many multi-agents' systems. And finally, it has been released many versions of the methodology, such as Gaia V2, ROADMAP and AUML.

After reviewing the many aspects that can improve the Gaia

methodology, whether in the implementation phase, the system requirements phase or within the two phases of analysis and design, which are contained in the Gaia methodology or otherwise. We have reached a solution to regarding of the system requirements that was ignored by Gaia methodology, leading to the improvement of Gaia methodology by providing a formal specification. After searching of the part of the formal specification for the system, we have reached the way to provide a formal specification of the system by Object Constraints Languages (OCL). After choosing the OCL, we chose the unified modeling language models (Class Diagrams) to be combined with the OCL to introduce a formal specification of the system into verification task. After obtaining the formal specifications we combine these specifications as input to the analysis phase in the Gaia and evaluate the combine process by JADE framework.

In this paper, we have introduced a solution for representing a case study from E-travel system. Into our solution, we introduced formal specification method for Gaia methodology which helps into develops agent-based systems in Gaia methodology by using class diagrams that extends with OCL constraints. The proposed solution improves the system through a better performance and management.

6. FUTURE WORK

Further research may be required to extend the solution to be applicable to other platforms like mobile platform, because the JADE framework includes the LEAP library that provides the environment to develop the agent for mobile device. Another work is to apply our solution on different case studies.

7. REFERENCES

[1] Srivastava, Praveen Ranjan, et al. "Extension of Object-Oriented Software Testing Techniques to Agent Oriented Software Testing." Journal of Object Technology 7.8 (2008): 155-163.

[2] DeLoach, Scott A. Multiagent systems engineering: a methodology and language for designing agent systems. Air force inst of tech wright-pattersonafb oh dept of electrical and computer engineering, 1999.

[3] Tveit, Amund. "A survey of agent-oriented software engineering." NTNU Computer Science Graduate Student Conference, Norwegian University of Science and technology. 2001.

[4] Genza, N., and E. Mighele. "Review on multi-agent oriented software engineering implementation." International Journal of Computer and Information Technology 2.03 (2013): 511-520.

[5] Dastani, Mehdi, and Jorge J. Gomez-Sanz. "Programming multi-agent systems." The Knowledge Engineering Review 20.02 (2005): 151-164.

[6] Tan, Ming. "Multi-agent reinforcement learning: Independent vs. cooperative agents." Proceedings of the tenth international conference on machine learning. 1993.

[7] Wooldridgey, Michael, and Paolo Ciancarini. "Agent-oriented software engineering: The state of the art." Agent-oriented software engineering. Springer Berlin Heidelberg, 2001.

[8] Akhtar, Nadeem. "Requirements, Formal Verification and Model transformations of an Agent-based System: A CASE STUDY." arXiv preprint arXiv:1501.05120 (2015).

[9] Steiner, Renee, Gary Leask, and Rym Z. Mili. "An architecture for MAS simulation environments." Environments for Multi-Agent Systems II. Springer Berlin Heidelberg, 2005. 50-67.

[10] Panait, Liviu, and Sean Luke. "Cooperative multi-agent learning: The state of the art." Autonomous Agents and Multi-Agent Systems 11.3 (2005): 387-434.

[11] Wooldridge, Michael, Nicholas R. Jennings, and David Kinny. "The Gaia methodology for agent-oriented analysis and design." Autonomous Agents and multi-agent systems 3.3 (2000): 285-312.

[12] Cernuzzi, Luca, et al. "The gaia methodology." Methodologies and Software Engineering for Agent Systems. Springer US, 2004. 69-88.

[13] Iglesias, Carlos A., Mercedes Garijo, and José C. González. "A survey of agent-oriented methodologies." Intelligent Agents V: Agents Theories, Architectures, and Languages. Springer Berlin Heidelberg, 1998. 317-330.

[14] Warmer, Jos B., and Anneke G. Kleppe. "The Object Constraint Language: Precise Modeling WithUml (Addison-Wesley Object Technology Series)." (1998).

[15] Warmer, Jos B., and Anneke G. Kleppe. The object constraint language: getting your models ready for MDA. Addison-Wesley Professional, 2003.

[16] Duarte, R., J. Junior, and A. Mota. "Precise modeling with UML: why OCL?." submitted to the Workshop of Formal Methods. 2003.

[17] Akhtar, Nadeem. "Requirements, Formal Verification and Model transformations of an Agent-based System: A CASE STUDY." arXiv preprint arXiv:1501.05120 (2015).

[18] Cabot, Jordi, Robert Clarisó, and Daniel Riera. "On the verification of UML/OCL class diagrams using constraint programming." Journal of Systems and Software 93 (2014): 1-23.

[19] Gogolla, Martin, Fabian Büttner, and Mark Richters. "USE: A UML-based specification environment for validating UML and OCL." Science of Computer Programming 69.1 (2007): 27-34.

[20] Brucker, Achim D., and Burkhart Wolff. "HOL-OCL: a formal proof environment for UML/OCL." Fundamental Approaches to Software Engineering. Springer Berlin Heidelberg, 2008. 97-100.

[21] Berardi, Daniela, Diego Calvanese, and Giuseppe De Giacomo. "Reasoning on UML class diagrams." Artificial Intelligence 168.1 (2005): 70-118.

[22] Cabot, Jordi, Robert Clarisó, and Daniel Riera. "Verifying UML/OCL operation contracts." Integrated Formal Methods. Springer Berlin Heidelberg, 2009.