Journal of Theoretical and Applied Information Technology
10th February 2013. Vol. 48 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

GENERAL AND SPECIAL-PURPOSE METHODOLOGIES FOR AGENT ORIENTED SOFTWARE ENGINEERING

[1]MOHAMMAD SHKOUKANI, [2]RAWAN ABU LAIL

[1]Asstt Prof., Department of Computer Information Systems, Applied Science University, Jordan

[2]Asstt. Prof., Department of Computer Information Systems, Philadelphia University, Jordan

E-mail: [1]m.shkokani@asu.edu.jo, [2]r_abulail@philadelphia.edu.jo

ABSTRACT

This paper provides a summary of software engineering process and its importance in open system industry. It describes the agent oriented software engineering development lifecycle. It also focuses on orientation of multi agent systems and on some representative agent oriented software engineering methodologies such as Gaia, ROADMAP, Tropos, and MaSE which are general purpose methodologies. Then it describes some special purpose methodologies such as ADELFE and SADDE. It also presents the phases for each methodology with its strengths and weaknesses. Finally it proposes the development of a new model that combines the features of two of the existing methodologies which are Gaia and Tropos by concentrating on their strengths and avoiding their weaknesses.

Keywords: *Agent Oriented Software Engineering Methodologies, Multi Agent Systems, Software Engineering Process.*

1. INTRODUCTION

There are many differences between software development and product development such as software produces an intangible product but product development and other engineering disciplines produce tangible products. Software engineering addresses all aspects of software production, it concerned with all phases of System Development Life Cycle (SDLC) from system requirement elicitation through maintenance, so it can be defined as the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software [22]. It depends on many processes including stakeholders with their different point of view, tools, and methods that are used to produce an automated solution [9].

When a large set of agents interact in a heterogeneous environment, many problems will appear such as management and coordination will be more difficult, probability of exceptional situations will increase, increases security holes, and unexpected global effects. So a successful open agent-based application will require software engineering methodologies [3].

Many methods and approaches have been suggested for developing agent-based systems, but none of these methods have been accepted as a standard, since there is a gap between agent oriented methods and the modeling needs of agent-based systems. Another problem of agent oriented software engineering methodologies that there is no agreement on how to identify roles in the analysis phase and how to identifies agent types in the design phase [1], [20].

2. MOTIVATION

Software development can be considered as a smart activity which needs high skills of planning, analysis, design, coding, testing and evaluation. These activities integrate different tools, methods, approaches, and methodologies; the coordination between these activities requires knowledge based reasoning, diagnosing, and adopting which is supported by the agent paradigm. Software is so important and it is present in every aspect in our life, pushing us to the world of distributed, context-aware computing systems. Multi-agent systems (MASs) are primary technology to model and

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

develop context-aware computing systems, because MAS consists of large number of cooperating entities that consider their context in performing their tasks. So context can be defined as any information about the objects, circumstances, or conditions by which an agent is surrounded that is related to the interaction between computer environment and agent. Furthermore software will become more intelligent and adaptive in the future and should have the ability to integrate with smart applications that have not been designed to work together [10].

The traditional software engineering approaches like structured approach or even object oriented approach offer limited support for the development of intelligent systems, electronic commerce, and enterprise resource planning. These new systems, in turn, call for new concepts, tools and techniques for engineering and managing software, for these reasons agent oriented software engineering development is gaining popularity over traditional development techniques [1], [2], [4].

3. A SURVEY ON AGENT ORIENTED SOFTWARE ENGINEERING

Agents and multi-agent system (MAS) have emerged as a powerful technology to face the complexity of new software systems. AOSE gives to the developer all the flexibility and the expressive power of agents and it helps with the software lifecycle management in an attempt to improve the quality of the resultant software products [10]. Agent-based computing promotes designing and developing applications in terms of autonomous software agents, the main enhancement that results from autonomy is that the agents are become proactive rather than reactive. Agents can achieve their objectives more flexible by interacting with each other in terms of high-levels protocols and languages. Agent concepts are natural to describe intelligent adaptive systems which are able to seek for optimal solutions for their design objectives; they are simply computer systems that are capable of autonomous in some environment in order to meet their design objectives [4], [8], [11].

An agent also called an intelligent agent, the words intelligent and agent describe some of the agent characteristic such as intelligent that is used because the software can have certain types of intelligent behavior which is the selection of actions based on knowledge, and the term agent tells the purpose of the software [6], [7]. The Agent oriented approach promises the ability to develop flexible systems with complex and sophisticated behavior by combining highly modular components [2], [5], [7].

3.1 General Purpose Agent Oriented Software Engineering Methodologies

There are many agent-oriented software methodologies that have been proposed such as GAIA, ROADMAP, MaSE and TROPOS. Gaia was the first complete methodology proposed for developing MAS from analysis to design. Gaia has two versions; the first version of it emphasizes the necessity to identify proper agent oriented abstractions, also it includes the analysis and design and excludes both requirement specification and implementation, it is applied after the requirements are collected and identified. In general, Gaia models are aimed at describing both the macro and the micro aspects of a MAS, as shown in figure 1 the analysis phase includes the role model and interaction model which are then used as input to the design phase which involves three type of models; agent model, service model, and an acquaintance model which are defined to compose a complete design of MAS [14].

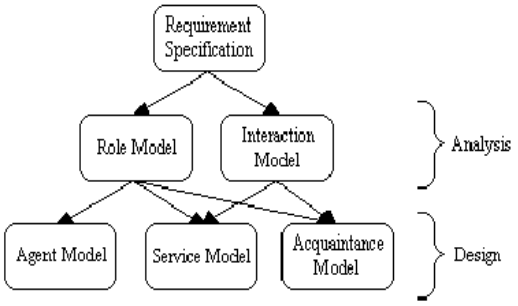

Fig. 1. The Gaia models.

The first version of Gaia has many limitations such as, it is appropriate only for the analysis and design of closed MAS but unfortunately agents in many MAS can belong to different stakeholders so it is unsuitable for open agent systems, another limitation is the notations that are used by Gaia in MAS which are not suitable to the real word systems especially the complex ones and do not even follow the standards that are accepted by software engineers [10], [14], [15].

The second version of Gaia is the official extension of Gaia which had been proposed to overcome the limitations which were exist in the first version of Gaia, thus Gaia version two is more oriented to design and develop complex systems in open environments [14], [15].

Journal of Theoretical and Applied Information Technology
10th February 2013. Vol. 48 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

The ROADMAP methodology is another extension to Gaia but it should be noted that ROADMAP was proposed before Gaia version two, so ROADMAP was an attempt to extend the first version of Gaia as a way to address its limitations by adding the followings: a dynamic role hierarchy in order to deal with open agent systems, additional models to describe the environment more explicitly, and the agent knowledge which so important in intelligent systems and that is not used in both versions of Gaia. Another advantage of ROADMAP over Gaia is the using of standard notations.

In ROADMAP, the system is viewed as an organization of agents consisting of role hierarchy which is the specification of the system and agent hierarchy which is the implementation of the system. As shown in figure 2 the environmental model and the knowledge model contain reusable high level domain information. The use case model, interaction model, role model, agent model, and acquaintance model application specific. The protocol model and service models describes reusable low level software components [14], [15].

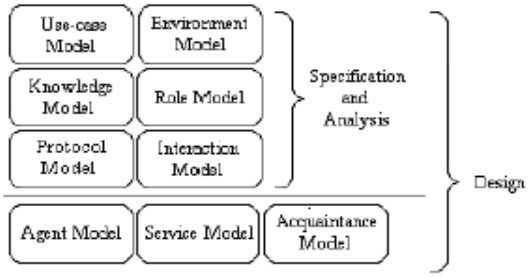

Fig. 2. Structure of ROADMAP models.

Tropos is a novel agent-oriented software development methodology that is based on two key features: First the notions of agent and related mentalistic notions are used in all software development phases from the early analysis phase to the implementation phase. Second it emphasizes on the phase that precedes the requirement specification that is early requirement analysis [17], [18], [19]. Tropos's software development life cycle consists of four phases: first phase is early requirement analysis which is concerned with understanding the problem, in this phase the intension of stakeholders are focused and modeled as goals, through the analysis the functional and non-functional requirements can be derived from the these goals [10], [24].

Second phase is late requirement analysis where the final system is described within its operational environment, so all functional and non-functional requirements are described. Third phase is architectural design where the final system's architecture is defined and describes how system's components interact and work together. Fourth phase is detailed design where the components' behavior is defined; it determines how the goals assigned to each actor. Although Tropos has many advantages over Gaia especially in its first two phases which are emphasizing on requirements, it has some limitations such as it still lacks tools that supported transition between four phases, another limitation is that it has not been used for a development of full-fledged MAS [25], [26], [27].

MaSE was originally designed to develop general-purpose MAS; it is a full life cycle methodology for analysis, design, and development. MaSE uses a number of models that is derived from UML models such as use case diagrams, sequence diagrams, and class diagrams. MaSE methodology is a specialization of many software engineering methodologies [21]. MaSE methodology consists of two main phases each one has many steps. The analysis phase provides a set of tasks and roles, which shows how the system meets its overall goals. The analysis phase consists of three steps; first step is capturing goals whose purpose is the transformation of system specification into set of structured system goals. Second step is applying use cases whose purpose is translating goals into tasks and roles, these use case are derived from system requirement, after that the use cases converted into sequence diagram. Third step is Refining rule whose purpose is to transform sequence diagrams into roles and their tasks.

The second phase of MaSE methodology is design phase which consists of four steps. First step creating agent classes, classes are created from roles that are identified in the analysis phase and identify conversation. Second step is constructing conversation, in the previous step the designer just identified the conversation but in this step the goal is define the detail of those conversations. Third step is assembling agent classes, which is defined internal agent architectures. Fourth step is system, in this step the final system is identified by using deployment diagrams [13], [16]. MaSE provides many advantages for building MAS such as providing a high level and top-down approach in many cooperative robot applications, one strength of MaSE is the ability to track changes during the whole process so every object created during analysis and design phase can be traced forward or backward [20], [21].

3.2 Special Purpose Agent Oriented Software

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Engineering Methodologies

Unlike general purpose methodology that is applicable to a wide range of MAS, a special purpose methodology is a methodology which is dedicated to a certain field of applications such as web based applications, telecommunication applications, and electronic business applications. Applications belonging to a same field share the same features and characteristics which are taken into account in the methodology and consequently are predominant to choose the methodology. A special purpose methodology improves the software development life cycle by facilitating developers' work and shortens time of development. Many differences could be identified between general purpose and special purpose methodologies but the main differences can be noticed in the last phases such as design, implementation, and deployment [10].

There are many special purpose methodologies; the first one is ADELFE which is dedicated to software that is characterized by system's need to environment adaptation and openness and the need of the system adaptation to an environment. It guarantees that the applications are developed according to the Adaptive Multi Agent Systems (AMAS) theory. The AMAS theory provides a solution to sophisticated systems which cannot be solved by using traditional algorithms, since these systems are open and complex [28].

The ADELFE methodology consists of six phases. First phase is preliminary requirements whose purpose is to define the proposed system; it concerns the system and its environment that will be deployed in. It also defines the functional and non-functional requirements. Second phase is final requirements whose aim is to transform requirements into a use case diagram and model system's environment. Third phase is analysis; this phase has to develop an understanding of the proposed system, its components' structure, and to know if AMAS theory is required or not. This phase includes the following steps: domain analysis, adequacy of the AMAS theory, agent identification, and adequacy of the AMAS theory at the local level. Fourth phase is design which aims to develop models for non-functional requirements and the solution domain, this phase includes two steps which are: study of interaction languages and agent design. Fifth stage is implementation, and the final phase will be the test [23], [28]. One of the advantages of ADELFE is providing an interactive tool that helps designer when following the process established in the method which does not exist in classical object oriented or in agent oriented methods [10].

The second special purpose methodology is called Social Agents Design Driven by Equations (SADDE) this methodology based on three main factors. First, a certain approach to the design of MAS by using Equation Based Models (EBM) where equations model all the behavior of agent society abstracting behavior from interaction between individual agents. Second, use electronic institutions as a way to constraint the interactions among individual agents in order to engineer the emergence. Third, using evolutionary computation techniques to find out what agent structure produces agent behavior that is specified in EBM. Based on these main factors the SADDE methodology had been proposed. SADDE methodology consists of four phases. First phase is EBM, in this phase a set of equation must be identified related to the agents in order to identify and model desired global properties of the MAS, this phase includes social interaction analysis which concerns with agent's sort such as what sort of interactions the agent must have, what sort of transactions they will have. Second phase is Electronic Institution Model (EIM), which aims to restrict interactions between agents, in this phase an individual behavioral analysis which is semi-automatic is used to determine if all aspects of agent's behavior are determined or there are some aspects which are not completely determined. Third phase is Agent Base Model (ABM) which aims to decide what the most appropriate decision models to use are. This phase includes experiments design where the experiments should be set to explore all possibilities and to see if the EBM is making the right prognosis or not. Fourth phase is Multi-agent system which is the final phase in SADDE methodology; it focuses on design of experiment for interaction among several agents. It includes experiment analysis (ABM redesign) that compares the predicted values of the global variables by EBM and the actual values of agent variables and their averages [10], [12].

4. THE PROPOSED MODEL

One of the problems of using AOSE is a formal identification and characterization of agent roles in the analysis phase and a formal determination of the agent type in the design phase. In our work we will combine two of the existing AOSE methodologies, which are Gaia and Tropos, by concentrating on their strengths and avoiding their weaknesses, for developing a new model helping in solving the above problems. In our proposed model we will combine the early

Journal of Theoretical and Applied Information Technology
10th February 2013. Vol. 48 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

requirement phase from the Tropos methodology with the analysis phase in the Gaia methodology.

In our proposed model we will use the i* notations that is used in Tropos methodology to analyze the requirements to find the functional and non-functional requirements as the first step in the Gaia methodology.

In i* notations actors are represented as circles; dependums – goals, softgoals, tasks and recourses – are respectively represented as ovals, clouds, hexagons and rectangles ; and dependencies have the form depender → dependum→dependee [25], [26]. We can summarize the i* notations as shown in figure 3.

*Fig. 3. I*Notation*

Finally in the proposed model as shown in figure 4 we will use the strategic dependency model from tropos as the resource for the requirement phase in Gaia as follows :

- Resource dependency →permission
- Softgoals and goals dependencies → responsibilities
- Task dependencies → Protocol
- There is no task dependency →activity

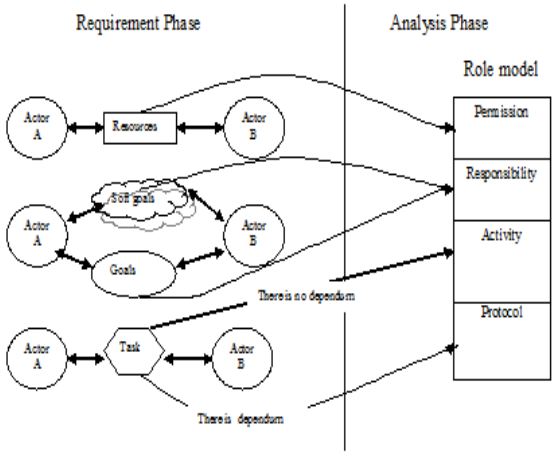

Fig. 4. The Proposed Model

5. CONCLUSION

Software development for enterprise systems has been very difficult because computing architecture have gone from centralized to distributed to fully open especially with the expansion of e-business, thus traditional software development methodologies are not suitable for such these systems.

So this paper has presented the importance of multi-agent systems and has described the main general purpose agent oriented software engineering methodologies, the paper has summarized the characteristics of original version of the Gaia methodology and has presented it extensions which are ROADMAP and Gaia version two that have been proposed in order to overcome its limitations. An overview has been presented for both Tropos and MaSE methodologies.

A general purpose methodology applicable to a wide range of MAS however sometimes there is many applications that needs special purpose methodology, so some special purpose methodologies for agent oriented software engineering has been discussed such as ADELFE which is based on rational unified process and uses unified modeling language notations, another special purpose methodology was SADDE which is dedicated to electronic institutions applications.

In conclusion, each methodology has its own advantages and limitations, so it could be suitable and very appropriate for one application but not for all applications.

Finally a new model has been proposed that combines the features of two of the existing methodologies which are Gaia and Tropos by concentrating on their strengths and avoiding their weaknesses, helping to formally identify and characterize roles in the analysis phase and determination of agent types which are recognized as open problems in actual active researches.

In the future work authors will make simulation for the proposed model to proof its strength over the existing methodologies.

6. ACKNOWLEDGMENTS

The authors are grateful to the Applied Science Private University, Amman, Jordan, for the financial support granted to cover the publication fee of this research article.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

REFERENCES:

[1] A. Sturm, D. Dori, and O. Shehory," Single-model method for specifying multi-agent systems", *ACM Press*, 2003, pp. 121-128.

[2] A. Garcia, R. Choren, C. Lucena, A. Romanovsky, H. Giese, D. Weyns, T. Holvoet, and P. Giorgini, , "Software Engineering for Large-Scale Multi-Agent Systems", *workshop report, ACM SIGSOFT Software Engineering Notes*, vol. 30, issue 4, Jul. 2005, pp. 1-8.

[3] R. Choren, A. Garcia, C.s Lucena, M. Griss, D. Kung, N.y Minsky, and A. Romanovsky, "Software Engineering for Large-Scale Multi-agent Systems" SELMAS, *ACM Press , International Conference on Software Engineering Proceedings of the 26th International Conference on Software Engineering*, 2004, pp 752-753.

[4] L. Sterling, and T. Juan ,"The software engineering of agent-based intelligent adaptive systems", *ACM Press , International Conference on Software Engineering Proceedings of the 27th international conference on Software engineering*, 2005, pp. 704-705.

[5] P. Massonet, Y. Deville, and C. Nève, "AOSE methodology to agent implementation", *ACM Press , International Conference on Autonomous Agents Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1*, 2002, pp. 27-34.

[6] G. Engels, W. Schäfer, R.t Balzer, and V. Gruhn, "Process-centered software engineering environments: academic and industrial perspectives", *ACM Press , International Conference on Software Engineering Proceedings of the 23rd International Conference on Software Engineering*, 2001, pp. 671-673.

[7] A. Tvei., "A survey of agent-oriented Software Engineering", www.csgsc.org, 2001

[8] H. Knublauch, "Extreme programming of multi-agent systems", *ACM Press , International Conference on Autonomous Agents Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 2*, 2002, pp. 704-711.

[9] C. Toffolon, and S. Dakhli, "A framework for studying the coordination process in software engineering", *ACM Press, Symposium on Applied Computing Proceedings of the 2000 ACM symposium on Applied computing* – Vol. 2, 2000, pp. 851-857.

[10] F. Bergenti, M. Gleizes, and F. Zambonelli, "Methodologies and software engineering for agent system", Kluwer Academic publishers, 2004.

[11] M. Wooldridg, and N. Jennings, "Intelligent Agents: Theory and Practice", www.citeseer.ist.psu.edu, 1995

[12] C. Sierra, J. Sabater, J. Agustí, and P. Garcia, "Integrating evolutionary computing and the SADDE methodology", ACM Proceedings *of the second international joint conference on Autonomous agents and multiagent systems* 2003, pp. 1116 – 1119.

[13] E.Villaplana, "A proposal for an organizational MAS methodology", ACM Press , *International Conference on Autonomous Agents Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, 2005, pp. 1370-1370.

[14]T. Juan, A. Pearce, and L. Sterling, "ROADMAP: extending the Gaia methodology for complex open systems", *ACM Press, International Conference on Autonomous Agents Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1*, 2002, pp. 3-10.

[15] T. Juan, L. Sterling, M. Martelli, and V. Mascardi, "Customizing AOSE methodologies by reusing AOSE features", ACM Press , *International Conference on Autonomous Agents Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, 2003, pp. 113-120.

[16] P. Ranjan, and A. Misra, "A hybrid model for agent based system requirements analysis", ACM Press, *ACM SIGSOFT Software Engineering Notes*, vol. 31, issue 3, May 2006, pp. 1-7.

[17] F. Giunchiglia, J. Mylopoulos, and A. Perini, "The tropos software development methodology: processes, models and diagrams", *ACM Press, International Conference on Autonomous Agents Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1*, 2002, pp. 35-36.

[18] A. Perini, A. Susi, and F. Giunchiglia, "Coordination specification in multi-agent systems: from requirements to architecture with the Tropos methodology", *ACM Press , ACM International Conference Proceeding Series; Vol. 27 Proceedings of the 14th international conference on Software*

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

engineering and knowledge engineering, 20002, pp. 51-54.

[19] P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, and J. Mylopoulos, "A knowledge level software engineering methodology for agent oriented programming", *ACM Press , International Conference on Autonomous Agents Proceedings of the fifth international conference on Autonomous agents*, 2001, pp. 648-655.

[20] M. Dastani, J. Hulstijn, F. Dignum, J. Jules, and C. Meyer, "Issues in Multiagent System Development", *ACM Press , International Conference on Autonomous Agents Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems* – Vol. 2, 2004, pp. 922, 929.

[21] S. DeLoach, "Multiagent systems engineering of organization-based multiagent systems", *ACM Press , International Conference on Software Engineering Proceedings of the fourth international workshop on Software engineering for large-scale multi-agent systems***,** 2005, pp. 1-7.

[22] B. Bracken, "Progressing from student to professional: the importance and challenges of teaching software engineering", *Journal of Computing Sciences in Colleges, ACM***,** vol. 19, issue 2, Dec. 2003, pp. 358-368.

[23] D. Capera, G. Picard, C. Bernon, and M. Gleizes, **"**Applying ADELFE Methodology to a Mechanism Design Problem", *IEEE International Conference on Autonomous Agents Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems* – Vol. 3**,** 2004, pp. 1508 – 1509.

[24] A. Dardenne, A. lamsweerde, and S.fixckas, "Goal directed requirement acquisitions", Science of computer programming, 1993, pp. 3-50.

[25] J. Brinkkemper, and J. Castro, "Tropos: A Framework for Requirements-Driven Software Development", Information Systems Engineering: State of the Art and Research Themes, Lecture Notes in Computer Science, *Springer-Verlag*, June 2000

[26] A. Fuxman, M. Pistore, J. Mylopoulos, and P. Traverso, "Model Checking Early Requirements Specifications in Tropos".

[27] R. Cervenka, "Modeling Notation Source Tropos, Foundation for Intelligent Physical Agents", 2003.

[28] G. Picard, C. Bernon, and M. Gleizes, "Cooperative Agent Model within ADELFE Framework: An Application to a Timetabling Problem", IEEE International Conference on Autonomous Agents Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems – Vol. 3, 2004, pp. 1506 – 1507.