International Journal of Technical Research and Applications e-ISSN: 2320-8163,
www.ijtra.com Volume 3, Issue 2 (Mar-Apr 2015), PP. 71-75

DEVELOPMENT OF A MULTIAGENT BASED METHODOLOGY FOR COMPLEX SYSTEMS

Reshma Kazmi, Brijesh Pandey, Namarata dhandha
Goel Institute of Technology and Management

Abstract- Multiagent Based Methodologies have become an important subject of research in advance Software Engineering. Several methodologies have been proposed as, a theoretical approach, to facilitate and support the development of complex distributed systems. An important question when facing the construction of Agent Applications is deciding which methodology to follow. Trying to answer this question, a framework with several criteria is applied in this paper for the comparative analysis of existing multiagent system methodologies. The results of the comparative over two of them, conclude that those methodologies have not reached a sufficient maturity level to be used by the software industry. The framework has also proved its utility for the evaluation of any kind of Multiagent Based Software Engineering Methodology.

I. INTRODUCTION

In our research, we view Multiagent Software Engineering as a further abstraction of the object-oriented paradigm where agents are a specialization of objects. Instead of simple objects, with methods that can be invoked by other objects, agents coordinate with each other via conversations and act proactively to accomplish individual and system-wide goals. Interestingly, this viewpoint sidesteps the issues regarding what is or is not an agent. We view agents merely as a convenient abstraction, which may or may not possess intelligence. In this way, we handle intelligent and non-intelligent system components equally within the same framework. In addition, since we view agents as specializations of objects, we build on existing object-oriented techniques and apply them to the specification and design of multiagent systems.

The primary focus of MaSE is to help a designer take an initial set of requirements and analyze, design, and implement a working multiagent system. This methodology is the foundation for the Air Force Institute of Technology's (AFIT) agentTool development system, which also serves as a validation platform and a proof of concept. The agentTool system is a graphically-based, fully interactive software engineering tool for the MaSE methodology. agentTool supports the analysis and design in each of the seven MaSE steps. The agentTool system also supports automatic verification of inter-agent communications and code generation for multiple multiagent system frameworks. The MaSE methodology, as well as agentTool, is independent of any particular agent architecture, programming language, or communication framework. The focus of our work is on building heterogeneous multiagent systems. We can implement a multiagent system designed in MaSE in several different ways from the same design.

Designing and building high quality industrial-strength software is difficult. Indeed, it has been claimed that such development projects are among the most complex construction tasks undertaken by humans. Against this background, a wide range of software engineering paradigms have been devised (e.g., procedural programming, structured programming, declarative programming, object-oriented programming, design patterns, application frameworks and component-ware). Each successive development either claims to make the engineering process easier or to extend the complexity of applications that can feasibly be built. Although there is some evidence to support these claims, researchers continually strive for more efficient and powerful software engineering techniques, especially as solutions for ever more demanding applications are required.

This paper will argue that analyzing, designing and implementing software as a collection of interacting, autonomous agents (i.e., as a *multi-agent system*) represents a promising point of departure for software engineering. While there is some debate about exactly what constitutes an autonomous agent and what constitutes interaction, this work seeks to abstract away from particular dogmatic standpoints. Instead, we focus on those characteristics for which there is some consensus. From this standpoint, the paper's central hypothesis will be advanced: for certain classes of problem (that will be defined), adopting a multi-agent approach to system development affords software engineers a number of significant advantages over contemporary methods. Note that we are not suggesting that multi-agent systems are a silver bullet there is no evidence to suggest they will represent an order of magnitude improvement in software engineering produc2tivity. However, we believe that for certain classes of application, an agent-oriented approach can significantly improve the software development process.

Seeking to demonstrate the efficacy of the agent-oriented approach, the most compelling form of analysis would be to quantitatively show how adopting such techniques had improved, according to some standard set of software metrics, the development process in a range of projects. However, such data is simply not available (as it is still not for more established methods such as object-orientation). However, there are compelling arguments for believing that an agent-oriented approach will be of benefit for engineering certain complex software systems. These arguments have evolved from a decade of experience in using agent technology to construct large-scale, realworld applications in a wide variety of industrial and commercial domains.

The contribution of this paper is twofold. Firstly, despite multi-agent systems being touted as a technology that will have a major impact on future generation software ("pervasive in every market by the year 2000" and "the new revolution in software"), there has been no systematic evaluation of *why this may be the case*. Thus, although there are an increasing number of deployed agent applications, nobody has analysed

precisely what makes the paradigm so effective. This is clearly a major gap in knowledge, which this paper seeks to address. Secondly, there has been comparatively little work on viewing multi-agent systems as a software engineering. This shortcoming is rectified by recasting the essential components of agent systems into more traditional software engineering concepts, and by examining the impact on the software engineering life-cycle of adopting an agent-oriented approach.

II. PROPOSED METHODOLOGY

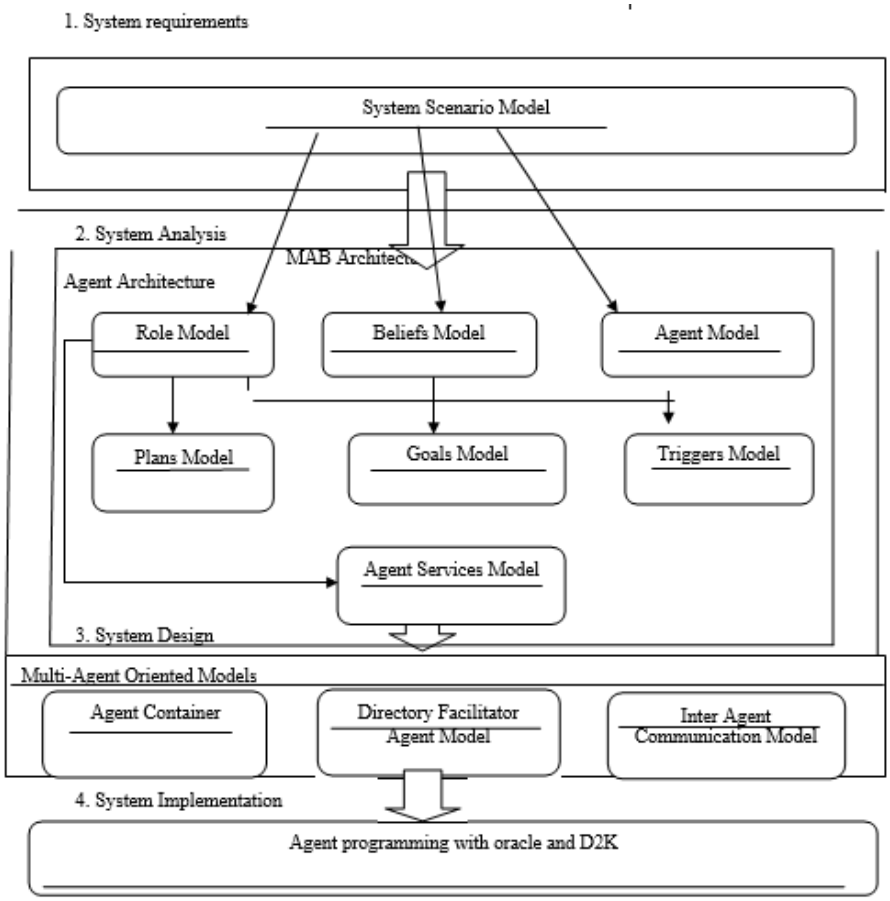

Figure 1 MAB Methodology

Plans Model: What are the plans and targets that have been set that should be used to achieve the goal of the agent. The main plan is to save the forest area and if required then make suitable arrangements for replantation of the trees along the project areas or blank spaces.

Goal Model: Goals which have to be achieved by the agents while working within the system. i.e, less damage to the forest area and more development.

Triggers Model: This model marks the events and change of beliefs that occur in the system. This may represent the different point of view of visualizing particular things which act as a trigger in the project.

Agent Service Model: Here the responsibilities and commitments of each agent is described and marks that it is fulfilled by all agents within the system for the smooth functioning of the system. i.e. the different agents work together for the development of the nation keeping in mind that the forest are to be saved and alternatives to be worked out before clearance is accorded.

Agent Container: The different behavior of various agentsvwithin the system design. Different role of agent in different areas. Different roles and duties of a particular officer.

Directory Facilitator Model: The default services of the agents within the system are defined. This defines the roles of the agents in the ideal stage. Default roles and duties of a particular officer.

Inter Agent Communication Model: Different communication protocols between the system and agents. this model takes care of the inter communication between different agents within a particular system. Or inter communication of the agents within the system and from externals taking into consideration the communication protocols.

System Requirements Phase

The system requirements phase guides towards the detection of the system components and their high level behavior .System components are like (objects, roles , resources etc) The system requirements phase is concerned with the description of the system scenario using well known techniques. The known techniques are such as Use-Cases Diagrams (UCDs), and UML use case diagrams.The system requirements phase is composed of the following model

In this model the scenarios of the system as a whole are described .The description includes components that the system is composed of and the tasks that have to be performed

by each component in the system. How these components interact with each other and with the external environment is also illustrated. To fulfill this description tasks, some specific well-known techniques have been such as User-Case maps(UCMs) and Use-Case diagrams(UCDs).

The Use-Case maps(UCMs) techniques is a precise structure notation .It describe the general behavior of the system in the form of scenarios without reffering to any implementation details. UCMs include adequate information in a summarized form. It has two advantages.

1) It enables developers to understand and conceptualize the behavior of the system as a whole.

2) It gives an explicit concept about how the system operates as a whole.

In the system scenario model, User-Case diagrams will also exploited. User-Case diagrams are an UML notation which describes the behavior of the system from the user point of view. It is through this notation that the roles in the system can be recognized .Recognition of roles within a system is a very helpful during the analysis and design phases as well as for understanding the system's requirements.

Analysis Phase

The analysis phase is concerned with the description of agent architecture and AAS. It is divided into two parts.The first part deals with agent architecture. The second part deals with AAS architectures.

The next section provides a detailed description of both architectures. The agent architecture step describes the internal structure of agents in the system. On the other hand AAS architecture step describes the relationship between the agents, the conversation and exchanged messages and agent services.This description is important in order to facilitate two main functions:

1) To enable negotiation and cooperation between agents.

2) To establish commitments and agreements that the agents should adhere to in order to provide the services to other agents in the system.

Agent Architecture

This step is concerned with the description of agent's internal structures. It describes the following:

1) Roles that the agent should play or perform in the system.

2) Agents that should exist in the system.

3) Goals and plans that each agent should have within.

4) Triggers that each role should be aware of as being events that take place in the system.

5) Roles Model

In this model the roles that an agent will play in the system will be stated. The important attributes of each role such as responsibilities, permissions, perceptions; obligations and constraints will be described in details.

Responsibilities are the activities that the role is responsible to perform. Permissions are the authorities related to numbers and types of resourses that will be exploited by agents in the system. Obligations are requirements that should be available to enable the role to start its functionality and carry out its responsibilities and activities. Constraints are restrictions and boundaries that the role must not infringe through executing its tasks.

Agent Model

In the agent model the internal description of agents within the system are illustrated .The internal structure of an agent is also described. In this methodology each agent possesses a goal or more, which it desires to realize .In addition an agent will also possesses beliefs that it depends on to achieve its goals. These beliefs can be considered as preconditions to initiate the achievement of goals for the system agents.

Agent Goals Model

In the agent goals model the goals that the agent desires to achieve will be identified.

Each goal and its priorities will be identified. Each goal will be initiated according to a specific priority. The plans which are prepared by the agent to satisfy the desired goal will also be identified. This model also contains preconditions and post conditions to initiate the process of achieving goals that the agent desires to realize.

Plans Model

In the plans model the plans that have to be performed or have to be followed by an agent during achieving a specific goal are recognized. In other words, every goal has to be achieved through a specific plan or more. Plans may execute in a sequential manner or according to the priority of each plan or in parallel manner. This model describes the plan as a set of tasks executed by the agent. Also includes the names of interaction protocols that take place between agents in the system.

Beliefs Model

The agent knowledge is considered as one of the most important parts of MAB model methodology. It stores relevant facts about the agent and its environment. Agent knowledge may be taken to explicitly represent the agent's beliefs about its environment or even about itself or about its environment or even about itself or about other agents. The beliefs model in MAB model methodology is carried out by following the scenario of UCMs. This is followed by the transfer of those scenarios into beliefs according to a specific goal or a specific plan or both.

Agent Triggers Model

This model describes the mechanism of how the agent perceives its environment through percepts and how to act on it through actions. Percepts are information coming from the environment which has an effect on the behavior of agents. According to that information the role performs some actions as a reaction. Actions are the mechanism through which an agent effects its environment.

There are events in AAS which occur during the system runtime. There may also be change in agent's beliefs. These events and changes in beliefs will trigger some affected agents to take some actions or reactions as a response to those events or changed in beliefs. These events and these changes in beliefs are called triggers. For if these triggers occur in the system then obviously there must be a party responsible for generating them. Triggers can be generated by an agent, an object or some by other party. Any party in a system generating a trigger is known as a source. There is also another party in the system which is the party that benefits from the operation. This is called the beneficiary. The beneficiary represents the agent who will react to this trigger. All expected

events and changes in beliefs in the system can be recognized through this model.

III. MAB ARCHITECTURE

This step is concerned with the description of AAS structure. It describes the relationships between the agents in the system, the conversation and exchanged messages. All this enable negotiation and cooperation between agents, the commitments and agreements that agents should adhere to in order to provide the services to other agents in the system.

Agent Interaction Model

An agent interaction model contains a description of the interactions between agents in the system where inter-communication among agents is performed. Therefore, interaction diagrams were adapted from object oriented design techniques, and allocated

To agents instead of objects.

The interaction diagrams from the system scenario model will be developed by capturing the lines that connect agents inside a use-case maps diagrams and transform them to conversations within the system. This model represents the primary step of Inter-agent communication model.

Agent Relationship Model

Agent relationship model is a set of system agents connected together to satisfy and pursue a common goal. This model consists of all system agents with the relationships, dependencies and authorities between system's agents being clearly described. The constraints and restrictions that a system must not encroach will also be described.

Agents Services Model

Agent services model provides a standard mean of inter-operating between different agent in the system. This model is intended to provide a common description of an agent services. The model is intended to provide a common description of an agent services. The model is also intended to define the place of the agent services within an agent system. This guides the agent community to those services easily. A services is realized by an agent and is used by another agent. Agent services are captured by means of the messages exchanged between requester agents and provider agents. The main goal of the agent services model is to facilitate access to sevices that are offered by each agent. Also, it organizes the cooperation between agents through constructing formal agreement. An agreement maintains agent's rights by providing them the ability to obtain those services in time.

Design Phase

The design phase is concerned with the detailed representation of the models developed in the previous phases and transforming them into patterns. These patterns are useful for actually implementing the multi-agent system. This phase captures key activities including agent structural design, development strategy and System design specifications. The design phase has three steps:

1) Creating agent container.

2) ConstructingInter-agent communications.

3) Creating Directory facilitator.

4) Agent Container Model

The agent container will describe the overall system organization which is composed of agent classes and the conversations between them. The agent's behavior is defined in terms of a container representing agent's roles in the system and the conversations in which they participate.

Inter –Agent Communication Mode

This model defines in details interactions among agenta in the system where communication between agents is established. To perform this communication between agents, agreed and accepted protocols have to be defined. these protocols are related with exchanged knowledge between agents in the system. Therefore we will exploit Agent Communication specifications.

Directory Facilitator Model

Directory facilitator model is responsible for providing the equivalent of a yellow pages directory service to ther agents in the system. Agents may register their services at the directory facilitator (DF) or query the DF to find out what services are offered by other agents. An agent is responsible to provide information related to service e. g. service type, service name etc. Furthermore, an agent can also deregister or modify its service details. Any agent can interact with DF to make its services public and to identify agents that provide a particular service.

Implementation Phase

Case Study: Project Clearance System

A brief description of how the project clearance system works which represent the case study to test and evaluate the new methodology.

The case study "Project Clearance System" has been chosen because it is simple and straight forward. It can be used to illustrate the type of reflective reasoning required by agents involved in a distributed collaborative environment. It entails a distributed design process, where several participants needed to interact with each other. It encompasses and highlights a number of underlying and interconnected agent concepts.

Project Clearance System

Project Clearance System was designed for Ministry of Environment & Forest, Central Region. In the regional office there are two sections one is the Environmental section which monitors the projects that have been accorded Environment Clearance and the second section i.e, is the forest is there accord forest clearance under FC Act 1980 to the projects submitted by the user agencies.

The services provided by the forest department

1) Different types of clearance to the user agencies for the constructions of various projects.

2) The user agency can submit the proposal online and being processed at different levels it is submitted to the Ministry for final action.

3) Projects once submitted is then evaluated, any further queries are required that is asked from the concerned authority.

4) If documents evaluated are up to the mark and that project is required for that place it is accorded clearance and the user agencies can start with the projects.

5) The projects submitted by the user agency are intimated with the decision of the Ministry with a set time frame required for the processing of the project.

6) Project Clearance System has also many bad experiences with the user agencies such as improper information, fake information, hiding the facts. In this phase, we deal with a part

of Project Clearance System behavior, triggered by the following kinds of events:

- Accepting projects for the clearance of new and existing User agencies on various locations.
- According clearance on the set rules and conditions.
- Providing the best services to the user agencies, by providing them with the progress of the objects submitted for clearance through their site and written communication.
- Handling the user agencies more thoroughly with the whom they found errors in.

The case study is considered to be applied with both UCMs and UCDs.

Implementation phase

The implementation (or construction) phase is the point in the development process. When we actually start to construct the solution. This is the time to start writing the program code. If the methodology process was followed so far, then a group of model would have been contructed. They will provide a lot of guidance for the implementation phase. The models have a complete set of design specification showing how the agent system and its components should be structured and organized. The next step would be to start handling out the various design specifications and start to build the implementation code step.

Oracle & Developer Platform

The oracle Database (commonly reffered to as RDBMS or simply as oracle) is an object-relational database management system (ORDBMS) Produced and marketed by oracle corporation. Oracle Database suite is a suite of development tools realesed by oracle corporation. The principal components were initially Oracle Forms and Oracle Reports .The Developer interface became more similar over time and they were eventually group together as Oracle IDE (integrated Development Environment). Oracle Forms is a software product for creating screens that interact with an Oracle database. It has an IDE including an object navigator, property sheet and code editor that uses PL/SQL. It was originally developed to run server –side in character mode terminal sessions .It was ported to other platforms, including Windows, to function in a client-server environment. Later version were ported to Java where it runs in a J2EE container and can integrate with Java and Web sevices. Oracle Reports is a tool for developing reports against data stored in an Oracle database. Oracle Reports consists of Oracle Reports developer(a component of the Oracle Develop Suite) and Oracle Application Server Reports Services J Developer is a freeware IDE supllied by Oracle Corporation .It offers features for development in Java , XML , SQL and PL/SQL ,HTML, JavaScript, BPEL and PHP. J Developer covers the full development lifecycle from design through coding, debugging, optimization and profiling to deploying. With J Develper, Oracle has aimed to simplify application development in addition to building an advanced coding-environment. Oracle J Developer integrates with the Oracle Application Development Framework(Oacle ADF) . The IDE platform also serves as the basis of another Oracle product, SQL Developer, which Oracle Corporation promotes specifically to PL/SQL-and database-developers.

IV. CONCLUSIONS

In this article, we have described why we perceive agents to be a significant technology for software engineering. We have discussed in detail how the characteristics of certain complex systems appear to indicate the appropriateness of an agent-based solution: as with objects before them, agents represent a natural *abstraction* mechanism with which to decompose and organize complex systems. In addition, we have summarized some of the key issues in the specification, implementation, and verification of agent-based systems, and drawn parallels with similar work from more mainstream computer science. In particular, we have shown how many of the formalisms and techniques developed for specifying, implementing, and verifying agent systems are closely related to those developed for what are known as *reactive* systems in mainstream computing. Finally, we have described some of the pitfalls of agent-based development. Software engineering for agent systems is at an early stage of development, and yet the widespread acceptance of the concept of an agent implies that agents have a significant future is software engineering. If the technology is to be a success, then its software engineering aspects will need to be taken seriously. Probably the most important outstanding issues for agent-based software engineering are: (i) an understanding of the situations in which agent solutions are appropriate; and (ii) principled but *informal* development techniques for agent systems. While some attention has been given to the latter (in the form of analysis and design methodologies for agent systems), almost no attention has been given to the former.

REFERENCES

[1] A. van Lamsweerde, and E. Letier, “Handling Obstacles in Goal-Oriented Requirements Engineering,” IEEE Transactions on Software Engineering vol. 26(10), pp. 978-1005, 2000.

[2] A. Cockburn, "Structuring Use Cases with Goals,” Journal of Object-Oriented Programming, Sep-Oct, 1997 and Nov-Dec, 1997.

[3] S. A. DeLoach and M. Wood, "Developing Multiagent Systems with agentTool," in Y. Lesperance and C. Castelfranchi, editors, Intelligent Agents VII - Proceedings of the 7th International Workshop on Agent Theories, Architectures, and Languages (ATAL'2000).

[4] Springer Lecture Notes in AI, Springer Verlag, Berlin, 2001.

[5] P. K. Harmer, G. B. Lamont, G.B, "An Agent Architecture for a Computer Virus Immune System," in Workshop on Artificial Immune Systems at Genetic and Evolutionary Computation Conference, Las Vegas, Nevada, July 2000.

[6] G. J. Holzmann, “The Model Checker Spin,” IEEE Transactions On Software Engineering, vol. 23(5), pp. 279-295, 1997.

[7] M. Wooldridge (1997) “Agent-based software engineering” IEE Proc. on Software Engineering, 144 (1) 26-37.

[8] H. Barringer, M. Fisher, D. Gabbay, G. Gough, and R. Owens (1989) “Concurrent MetateM: A framework for programming in temporal logic” REX Workshop on Stepwise Refinement of Distributed Systems: Models, Formalisms, Correctness (LNCS Volume 430), 94-129. Springer-Verlag.

[9] J. R. Marden and A. Wierman, “Overcoming limitations of game-theoretic distributed control,” in Proc. 47th IEEE Conf. Decision Control, Dec. 2009, pp. 6466–6471.