EX-MaSE: AN EXTENDED AND EXPRESSIVE MULTIAGENT SYSTEM ENGINEERING METHODOLOGY

Ahmad Abdollahzadeh Barfouroush
ahmad@ce.aut.ac.ir

Shiva Vafadar
vafadar@aut.ac.ir

Intelligent Systems Laboratory
Computer Engineering Faculty
Amirkabir University of Technology
Tehran- Iran

ABSTRACT. In this paper, we introduce a new methodology in agent oriented software engineering which is an extension of MaSE methodology. We explain our improvement process and evaluation criteria for evaluating MaSE methodology. We report weaknesses of MaSE methodology in different phases of system life cycle based on using MaSE to implement a case study in electronic commerce. In order to bridge the gaps in original methodology, we introduce new methodology named E-x-MaSE, which has “Environment Modeling”, “Knowledge Modeling” and “Agent-Object Modeling” steps in addition to the steps of the MaSE methodology. For evaluating our new methodology, we use another case study in electronic commerce, which have implemented by E-x-MaSE. We will explain why by using these improvements E-x-MaSE is more expressive, refinable, accessible and extendible than the original methodology. Also reusability and maintainability of the developed systems using the E-x-MaSE methodology will be improved.

KEYWORDS: Software engineering, Agent base methodologies, Electronic commerce

1. INTRODUCTION

Now agent-based solutions are used in some industrial, commercial, medical, networking and educational application domains. These software solutions are mainly complex, open and distributed. Agent oriented software engineering is a powerful way of approaching large scale software engineering problems and developing agent-based systems [1]. There are some agent oriented methods for analysis and design of agent systems. An overview of such methods is presented in [2,3]. The early versions of these methods have many weaknesses in analysis and design of agent-based systems. Therefore, researchers are extending and improving these methods for agent-based systems so that they can better guide developers to implement agent-based systems.

One of the most developed agent oriented methodologies is Gaia [4]. It has been improved to model complex open systems [5] and to model aspects of Internet applications [6]. The Multiagent System Engineering (MaSE) methodology is another one which takes an initial system specification and produces a set of design documents in a graphically based style [7]. This methodology has been improved by researchers in order to cover and model some aspects of multiagent systems such as mobility [8], ontology [9] and organizational rules [10]. Although MaSE has been improved in many aspects by its developers in recent years it still has some weaknesses which should be discovered and improved.

We use an improvement process for extending MaSE methodology. Figure 1 shows the improvement process.

FIGURE 1. OUR IMPROVEMENT PROCESS

In our previous works, according to this process, first we defined a case study named electronic book shop (E-shop). During analysis and design of our case study by MaSE and implementing E-shop application, we discovered some implementation requirements which methodology does not provide them. For bridging these gaps, we proposed some extensions for MaSE methodology [11].
Following our previous work for improving MaSE methodology, in this paper we continue and mature discovered weaknesses in the MaSE methodology and introduce E∗-MaSE as a new methodology to bridge these gaps. In the E∗-MaSE we have added an “Environment Modeling” step to specify environment and properties. We also have added “Knowledge Modeling” step in order to model knowledge of each single agent in the analysis phase of the methodology. Also, we have proposed an "Agent–Object" model to decrease design and implementation complexity.

In order to evaluate proposed extensions, we will continue improvement process. In this way, we have used a more complex case study named CASBA and analyzed and designed it by E∗-MaSE. Based on this experiment and other evaluation methods, we will evaluate E∗-MaSE based on software engineering criteria for multiagent systems and agent properties as well.

The remainder of this paper introduces E∗-MaSE as an extension to MaSE and the results that we have obtained by evaluation E∗-MaSE methodology. In section 2, we describe in detail the identified weaknesses in the MaSE. In section 3, we introduce new models and steps in E∗-MaSE methodology in comparison with MaSE methodology in order to bridge the gaps and remove weaknesses of the MaSE methodology. In section 4, we will evaluate E∗-MaSE by using some evaluation techniques and criteria. In section 5, we will present our conclusions and further works.

2. MaSE WEAKNESSES

We have evaluated the MaSE methodology according to our improvement process. We identified the following weaknesses during analysis, design and implementation of the previous case studies [11, 12] and completed it by investigating specific requirements of multiagent systems.

1. Existing gap between analysis and design phases: One of the most important weaknesses of the MaSE methodology is the existing gap between analysis and design phases. For example, the "Assembling Agent Classes" step in which agent architecture is constructed is an isolated step that does not have enough connection with other analysis and design steps. In this step, the designer needs information that is not provided in the earlier steps. Therefore, he/she should go back to first step and analyses the role requirements to gather necessary information for decision making about the appropriate agent architecture.

2. Lack of knowledge modeling in the methodology: Knowledge is an essential component of an agent. An agent has a knowledge component and a mechanism for operating on it or drawing inferences from its knowledge. In the MaSE the knowledge of an agent is represented at the agent level design in "Assembling Agent Classes" step. Designer uses each agent knowledge type to select agent architecture and then agent components. Since MaSE is a communication oriented methodology, its focus is on the interactions between agents. Since the previous extension to MaSE for ontology modeling [9] focuses on data model of the system and not on the mechanism for operating on or drawing inferences from it, modeling the internal knowledge of agents such as rules or plans is ignored and is not addressed in this step of the methodology.

3. Lack of environment modeling in the methodology: Environment is an important factor for agents. Almost in all definitions, an agent is an entity in an environment and reacts to changes in the environment [13]. On the other hand, the environment of an agent influences on the mechanism of acting and deciding by an agent. Therefore decision about the architecture of an agent is depending on the environment of the agent and subsequently multiagent system. In the MaSE methodology, environment of the system and the roles in the system does not consider in the methodology steps.

4. Mapping all roles to agent abstraction: In the "Creating Agent Classes" design step, all refined roles are mapped to the corresponding agents. Although some of them are independent roles, they do not have desired properties of agents such as autonomy and reactivity. We believe that they are just objects in a multiagent system and in such cases we should decompose such entities to objects.

5. Weak documentation: In the MaSE, documentation of many aspects of an agent-based system is implicit and the scope of the roles is not explicit. Since a good documentation is needed for maintenance of a system, the methodology should guide software engineer to produce necessary documents and artifacts for better maintenance of the developed system. Explicit documentation for roles helps the methodology to achieve this goal.

6. Lack of clear basic definitions and guidelines: MaSE does not have explicit definition for goal, role, task and activity. For example, overlap between goals and tasks definitions causes conflict between them. Consequently, the analyst should go back to the first analysis step and refine the distinguished goals. These uncertainties in definitions increase time and cost of performing analysis activities.

7. Problem in modeling interactions: MaSE uses UML sequence diagram for modeling role interactions so it can not model some specific characteristics of agents such as concurrent threads of interactions among roles.

We have proposed some models and steps to remove some of the above weaknesses in the MaSE. We will describe our improvements in the E∗-MaSE methodology.
3. Ex-MaSE METHODOLOGY

To improve and extend the MaSE methodology, we have added the “Environment Modeling” and the "Knowledge Modeling" steps in the analysis phase and we have extended the "Agent Modeling" in "Creating Agent Classes" step to "Agent-Object Modeling". We also refined “Applying Use Cases” step in order to justify it for agent interactions. In the following subsections we will describe new steps in the Ex-MaSE methodology. Figure 2 shows the Ex-MaSE steps and models.

3.1. ENVIRONMENT MODELING

MaSE methodology does not consider environment of the multiagent system. Environment of the system highly affect on the design of each role and subsequently multiagent system. Therefore a good methodology for multiagent systems should consider environment modeling in analysis phase. We have added “Environment Modeling” step to the analysis phase of original methodology. Purpose in this step is identifying environment entities of the system and properties for each role and entire multiagent system.

There are two steps in environment modeling, “Capturing Environment Entities” and “Specifying Environment Properties”. In first step, which is “Capturing Environment Entities”, all the entities of the environment that influence on multiagent system, are determined. These entities could be users, physical resources such as sensors and responders and external data resources ranging from HTML files to databases. In this step a context view of the multiagent system is constructed in an environment diagram. Figure 3 illustrates an example of the environment diagram for CASBA system.

FIGURE 3: ENVIRONMENT DIAGRAM

In “Capturing Environment” step the analyst should document environment entities for each role in “Role Schema” as well. Figure 4 demonstrates an example of “Role Schema “ for auctioneer role. Slots of this schema are required in design phase for constructing I-O interface component in “Assembling Agent Classes” step.

FIGURE 4: Role Schema

In the second step, properties of the environment are determined from each role point of view. Properties of the environment are an important factor in deciding about architecture of an agent. Different environment types require different agent types to deal with them effectively. One environment may view differently by different agents in a multiagent system. Therefore for each role in the system, properties of the system should be specified.

In this step analyst should fill in an environment schema and specify the entities and factors that cause properties of the environment such as accessibility or non accessibility, deterministic or non deterministic, episodic or non episodic, dynamic or static, discrete or continuous.
Figure 5 illustrates an example of the environment schema for CASBA system. By using this schema complexity factors for each role is determined.

<table>
<thead>
<tr>
<th>Properties</th>
<th>Roles</th>
<th>Administrator</th>
<th>Auctioneer</th>
<th>Buyer</th>
<th>Seller</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accessibility</td>
<td>Buyer and Sellers information is not accessible all the time.</td>
<td>Admin send required information</td>
<td>Just Use send information</td>
<td>History of Bidding</td>
<td>All the Environment</td>
</tr>
<tr>
<td>Non Accessibility</td>
<td>Number of Sellers and Buyers</td>
<td>Information for Decision Making are deterministic</td>
<td>Environment changes all the time by other buyers biddings</td>
<td>All the Environment</td>
<td></td>
</tr>
<tr>
<td>Deterministic</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Non Deterministic</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Episodic</td>
<td>View environment episodically.</td>
<td>Decide episodically</td>
<td>Environment is no episodic but is viewed episodically</td>
<td>View episodically</td>
<td></td>
</tr>
<tr>
<td>Non Episodic</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Static</td>
<td>New buyers can add all the time.</td>
<td>After running auction is static</td>
<td>Just see users</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dynamic</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Discrete</td>
<td>View by the admin in discrete time</td>
<td>Decision making is discrete</td>
<td>Discrete actions</td>
<td>Discrete</td>
<td></td>
</tr>
<tr>
<td>Continues</td>
<td>Registration of sellers and buyers</td>
<td>Receiving Buyers bids</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

"Assembling Agent Classes" step have some common knowledge components which can be categorized as rules, facts and plans. The rules are used to interpret the stimulus and generate a response if appropriate. Analyst should model each rules according to the schema that is shown in Figure 6.

In general, a plan consists of one or more steps, each of which contains a number of attributes. Also for documenting each predefined plan used by a role, we propose a “Plan Schema” which has been shown in Figure 7. This schema can be used for documenting dynamic and static plans of a role. For each fact in the system, analyst should specify a name, resource which provides the fact, and update conditions. The facts contain any predefined information the agent may need as well as any new information derived by the agent.

3.2. KNOWLEDGE MODELING

We have proposed a “Knowledge Modeling” step for modeling knowledge of the roles in the analysis phase. By adding this step to the analysis phase, we model knowledge level aspects of system in the analysis phase. Knowledge component is necessary for decision making about the appropriate agent architecture and therefore should be provided in the analysis phase.

By using "Use Case Model" and "Environment Model" and “Environment Schema”, the system analyst can identify required knowledge for each role to achieve its associated goals and to handle complexity caused by each environment type. The architectural styles that are used in

"Assembling Agent Classes" step have some common knowledge components which can be categorized as rules, facts and plans. The rules are used to interpret the stimulus and generate a response if appropriate. Analyst should model each rules according to the schema that is shown in Figure 6.

In general, a plan consists of one or more steps, each of which contains a number of attributes. Also for documenting each predefined plan used by a role, we propose a “Plan Schema” which has been shown in Figure 7. This schema can be used for documenting dynamic and static plans of a role. For each fact in the system, analyst should specify a name, resource which provides the fact, and update conditions. The facts contain any predefined information the agent may need as well as any new information derived by the agent.

3.3. AGENT-OBJECT MODELING

In the MaSE, all refined roles in analysis phase are
mapped to the corresponding agent class in design phase. In this way every entity in a system is modeled as an agent. But there are some roles in the systems which are simpler than an agent. A sound principle in system development states that one should always attempt to develop the simplest solution for any given problem [14]. We believe that a good agent oriented methodology would encourage developers to achieve the correct decomposition of entities into either agents or objects. A role that has a deterministic response to messages and does not have the desired characteristics of an agent is essentially a simple object in an agent-based system. Because objects are simpler computational entities than software agents, in such cases it is more efficient to use object abstraction. If one follows this guideline in design of an agent oriented analysis and design method, the resulted method can help the system analyst to choose effective decomposition in design phase. In this way, individual computational entities in the system could be modeled and implemented based on either software agents or objects. Thus, we have refined "Agent Model" to "Agent-Object Model". In this model designer can map a role to either an agent or an object. If a role does not use any rule and fact or does not act based on any predefined plan which provided in the proposed "Knowledge Modeling" step, it is not an agent and can be designed as an object.

4. EVALUATING THE E_{\text{x}}-MaSE

There are some techniques for evaluating an agent based methodology such as comparison with other methodologies, evaluation based on software engineering criteria, using case studies and comparison based on existed versus needed constraints [15]. In this paper in order to evaluate E_{\text{x}}-MaSE we have used a mixture of these techniques. In this way, we have compared E_{\text{x}}-MaSE with some other methodologies based on their ability to model special characteristics of agents in different steps of the methodology. In order to evaluate software engineering criteria, we have focused on our experiments on analysis, design and implementation of case studies by using these methodologies. In this comparison, we have considered MaSE and E_{\text{x}}-MaSE which we have used them in case studies. In the following subsections, we will describe evaluation of E_{\text{x}}-MaSE and its advantage in detail.

4.1. EVALUATING MASE BASE ON AGENT CHARACTERISTICS

One of the techniques to evaluate a methodology is the examination of agent based system characteristics [15]. For evaluating E_{\text{x}}-MaSE, we have selected some of the agent based system characteristics and compared E_{\text{x}}-MaSE with them base on their ability to model these characteristics of agent based systems. The result of this comparison is shown in Table 1. We have selected agent base systems characteristics from [14] and have sorted them according to the most usage of them in agent definitions by different researchers [13]. The methodologies have selected among different categories. MAS-commonKADS is an extension of knowledge engineering methodologies. Gaia is one of the most important methodologies for multiagent systems. MaSE is the original methodology in our research. MESSAGE is an methodology which try to gather different features of other methodologies in one and it uses widely in communication field [16]. ROADMAP is an extension to Gaia for modeling open systems. We also considered different extension on MaSE methodology in recent years in the comparison. E_{\text{x}}-MaSE is our new methodology.

<table>
<thead>
<tr>
<th>Methodology</th>
<th>Gaia</th>
<th>MaSE</th>
<th>MESSAGE</th>
<th>MAS-commonKADS</th>
<th>ROADMAP</th>
<th>ROADMAP</th>
<th>Extentions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Autonomy</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Environment</td>
<td>-</td>
<td>-</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Proactiveness</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Knowledge Modeling</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Goal Directed</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>-</td>
<td>*</td>
<td>*</td>
<td>+</td>
</tr>
<tr>
<td>Mobility</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Ontology</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>ACL Modeling</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Agent Architecture</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>

According to the Table 1, in the E_{\text{x}}-MaSE we have added modeling ability of knowledge and environment of the multiagent system to the MaSE. As shown in Table 1, in the E_{\text{x}}-MaSE methodology most characteristics of agent based systems can be modeled.

4.2. EVALUATING E_{\text{x}}-MASE BASE ON SOFTWARE ENGINEERING CRITERIA

We have evaluated E_{\text{x}}-MaSE base on software engineering viewpoint. In this way, we have selected some criteria for evaluating the quality of the modeling technique in agent based methodologies [5,14,17,18]. We have evaluated MaSE and E_{\text{x}}-MaSE base on these criteria according to their modeling technique and our experiments to analysis and design of the E-shop and
CASBA case studies by these methodologies. Since evaluating methodologies based on software engineering criteria needs experiments on analysis and design of the case studies, we just included MaSE and E_x-MaSE in this comparison.

Preciseness means the semantic of modeling technique must be ambiguous in order to avoid misinterpretation of models by those who use them. In this methodology based on available documents, in most steps modeling techniques are expressive except that tasks and goals definitions. Based on our studies in this methodology, there are not enough guidelines in the methodology in order to distinguish between tasks and goals of the system. In E_x-MaSE we didn't concentrate on this weakness of the original methodology. Therefore in both of them expressiveness of the methodology is satisfied to some extent.

Accessibility means a modeling technique should be comprehensible to both experts and novices. We have used MaSE as a novice developer. In this way except assembling agent classes step there was no problem in using different steps of the methodology. Therefore we believe that accessibility in the MaSE methodology satisfies to some extent. In the E_x-MaSE we have added knowledge model and environment model in order to bridge this gap in the MaSE methodology. By adding these models to analysis phase accessibility of the E_x-MaSE is improved. But because we have added new models to the methodology, it should be tested by other developers as novices in the E_x-MaSE as well.

Expressiveness is another criterion in the evaluation of the methodology. Based on this criterion a modeling technique should be able to represent structure of the system, knowledge encapsulation, data flow, control flow in the system and interaction with external systems [18], the resource constraints (such as time, CPU and memory) and the system physical architecture[15]. We have increased expressiveness of our methodology by adding knowledge modeling step in order to model knowledge encapsulation of the systems. On the other hand using Agent-Object model the designer could consider system constraints such as CPU and memory by mapping roles to either agent or object. Since objects are simpler entities than agents, by designing some simple roles as objects, the system needs less memory and CPU. Therefore expressiveness in the E_x-MaSE is improved in comparison with original methodology.

Modularity means a modeling technique should be expressible in stages. That is when new specification requirements are needed there is not need to modify previous parts and these maybe used as part of the new specification. MaSE and E_x-MaSE are modular. These methodologies use building blocks such are roles and tasks in analysis phase and agents in design. A new role in the system can map to new agent type or consider as a part of existing role. Therefore, adding a new role just influence on one part of the system.

Complexity management is another criterion for evaluating agent based systems. In order to achieve this criterion, a modeling technique should be expressed and then examined at various levels of detail. MaSE presents a hierarchy view of goals in the system. Also roles and their tasks are in different level of abstractions. In E_x-MaSE we have two different level of abstraction of environment factors in environment modeling step. A context view of the system and a schema for each role is modeled. By using these models, analyst could analysis environment of the system which is one of the most important factors for the complexity of a multiagent system. In knowledge modeling step according to the environment factors for each role, required knowledge to handle these complexities presents. According to these reasons we believe that E_x-MaSE is stronger in complexity management.

Executability means either a prototyping capacity or a simulation capacity should be associated with at least some aspects of modeling technique. That is a modeling technique should have related tools that allow computation for sample inputs. In MaSE and E_x-MaSE there are not such tools. Therefore this issue is not deal within these methodologies.

Refinability means a modeling technique should provide a clear path for refining a model through gradual stages to reach an implementation. In the MaSE we have discovered that smooth transition between different steps of the methodology has been lost, especially at “Assembling Agent Classes” step. By adding knowledge modeling step in E_x-MaSE we have bridged these gaps and provided smooth transition between different stages of the methodology from analysis to implementation. Therefore E_x-MaSE is more refifiable than original methodology.

Analyzability means a methodology or an associated tool is available to check internal consistency or implementation of models or identify aspects that seem to be unclear. In MaSE, agentTool has validation and verification ability. Therefore analyzability is provided in MaSE. In E_x-MaSE we did not deal with agentTool. To achieve analyzability in E_x-MaSE consistency and coverage should be considered at environment modeling step and knowledge modeling step in analysis phase and relevant agents in design phase.

Openness means a modeling technique should provide a good basis for modeling agent based systems without coupling them to a specific architecture, infrastructure or programming language. MaSE and E_x-MaSE provide openness because none of them are coupled with specific
infrastructure, architecture or programming language. Scalability means a modeling technique should be applicable for systems which the scale of the system may expand greatly over system’s useful life cycle[17].

Scalability should be promoted to facilitate the growth of the open system to any given complexity. In MaSE and E_r-MaSE we didn’t deal with this property. Therefore it should test in an open system with growth in different stages and with different complexity levels.

Maintainability means the case with which a program can be corrected if an error is encountered, adapted if its environment changes or enhance if the customer desires a change in requirements [17]. On of the ways to achieve this goal is well documents produce during analysis and design phases. A good agent oriented methodology should consider this criterion and guide software engineer to produce a maintainable system. In the MaSE, documentation is based on graphical diagrams and in this methodology a brief document of a role and its scope doesn’t produce during analysis phase. Documentation provides a foundation for successful engineering and more important, guidance for software support. We have added a stage to produce brief documentation of each role in environment modeling step and knowledge mechanisms required by each role in knowledge modeling step but we did not evaluate this criterion in maintenance phase of a case study. To evaluate it exactly, maintenance of an implemented system should be done.

<table>
<thead>
<tr>
<th>Criteria</th>
<th>MaSE</th>
<th>E_r-MaSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precisionness</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>Accessibility</td>
<td>*</td>
<td>?/+</td>
</tr>
<tr>
<td>Expressiveness</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Extendibility</td>
<td>?</td>
<td>?/+</td>
</tr>
<tr>
<td>Complexity Management</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Executability and Testability</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Refinability</td>
<td>*</td>
<td>+</td>
</tr>
<tr>
<td>Openness</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Analysability</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Scalability</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>Scalability</td>
<td>?</td>
<td>?/+</td>
</tr>
<tr>
<td>Reusability</td>
<td>?</td>
<td>?/+</td>
</tr>
</tbody>
</table>

Reusability extent to which a system or some parts of it can be reuse on other systems related to packaging and scope of the function the system performs[17]. We believe that a good methodology should provide required facilities to achieve this goal. Since the scope of the system and its components should be clarified, in E_r-MaSE by documenting environment and knowledge of each role in the system, the designer gather required information for reusability of the system. However we believe this criterion should be tested in a case study that try to use components of other systems.

Table 2 is providing a qualitative summary of our evaluation. According to the table, our new methodology is more expressive, refinable, accessible and extendible than the MaSE methodology. Also reusability and maintainability of the developed systems using the E_r-MaSE methodology will improve.

5. CONCLUSIONS AND FURTHER WORKS

In this paper we introduced E_r-MaSE as a new methodology for agent based systems by extending and improving an existent methodology, MaSE. In this way, we used an improvement process and discovered weaknesses in original methodology. By adding “Environment Modeling” Step, “Knowledge Modeling” Step and “Agent-Object Modeling” Step to the original methodology and refining the “Applying Use Cases” Step we have increased expressiveness, refinability, Complexity management, accessibility and extendibility of the original methodology and reusability and maintainability of the developed systems using the methodology.

The improvement process for improving the methodology could continue its iteration with another case study and evaluation until all agent-based systems abstractions and semantics have been modeled effectively by the methodology and the methodology satisfies all the criteria of a desired one. Using different case studies at different levels of complexities and in a variety of application domains can help us to discover the gaps and weaknesses of the methodology.

In this paper and similar researches, evaluation of the methodology is done qualitative. In order to change this process to an engineering process, these evaluation should be changed to measurement. To measure software engineering criteria in multiagent systems, metrics for measurement should be defined and specified for multiagent systems. To continue our research on agent based methodologies and
agent oriented software engineering, we are working on specifying this metrics in E$_c$-MaSE and agent based methodologies generally.

6. REFERENCES


