World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:4, No:4, 2010

Extensions to Some AOSE Methodologies

Louay M. Jeroudaih[#1], Mohamed S. Hajji[#2]

Software Engineering department, Faculty of IT, University of Damascus-Syria
[1]louy.jer@gmail.com [2]mshaj@scs-net.org

***Abstract*—** This paper looks into areas not covered by prominent Agent-Oriented Software Engineering (AOSE) methodologies. Extensive paper review led to the identification of two issues, first most of these methodologies almost neglect semantic web and ontology. Second, as expected, each one has its strength and weakness and may focus on some phases of the development lifecycle but not all of the phases. The work presented here builds extensions to a highly regarded AOSE methodology (MaSE) in order to cover the areas that this methodology does not concentrate on. The extensions include introducing an ontology stage for semantic representation and integrating early requirement specification from a methodology which mainly focuses on that. The integration involved developing transformation rules (with the necessary handling of non-matching notions) between the two sets of representations and building the software which automates the transformation. The application of this integration on a case study is also presented in the paper. The main flow of MaSE stages was changed to smoothly accommodate the new additions.

***Keywords*—**Agents, Intelligent Agents, Software Engineering (SE), UML, AUML, and Design Patterns.

Open Science Index, Computer and Information Engineering Vol:4, No:4, 2010 publications.waset.org/5825.pdf

I. Introduction

AGENT-Oriented Software Engineering (AOSE) is one of the modern approaches in software engineering field. It focuses on agents' domain and tries to represent agents and its related concepts in high-level abstractions in order to describe the software system clearly [1]. Agent-oriented approach presents a new means of analyzing, designing and developing complex competitive and cooperative software systems. It tries to improve current practices in software engineering and to extend the range of applications these methodologies can handle [2]. This paper gives an overview of recent researches on Agent-Oriented Software Engineering and proposes enhancements to MaSE, a leading methodology in this field. The improvements concentrate mainly on MaSE weaknesses in requirement specification by including a special stage for requirements and building transformation rules to transfer them to the standard MaSE diagrams. In addition, the paper describes an advanced stage of work in progress for incorporating ontology concepts into the MaSE diagrams.

In the second section, the main well established AOSE methodologies are reviewed. The third section discusses how new concepts will be integrated into MaSE methodology. The main transformation rules are presented in the forth section, while a good size case study is handled is the fifth section. Conclusions and future works are left for the final section.

II. Quick Review of Well Established AOSE Approaches

Many agent-oriented methodologies were developed to benefit from agents features/capabilities inside the software lifecycle. Their general aim was to build a consistent process for developing software or to enhance some particular parts of the current development lifecycle. So the efficiency and performance can be improved. There is a number of agent-oriented derived methodologies like: Gaia, MaSE, MAS-CommonKADS, MASSIVE, MESSAGE, PASSI, Tropos and Prometheus [3][4]. The following subsections overview Gaia, Tropos and MaSE methodologies which are in the leading pack:

A. Gaia

Wooldridge, Jennings and Kinny [1][5] presented Gaia methodology as a general methodology to support both the micro-level (agent structure) and macro-level (agent society and organization structure). This methodology suffers a problem that agent abilities are static. The main motivation behind Gaia was representing the autonomous, problem-solving nature of agents, the ways of performing interactions

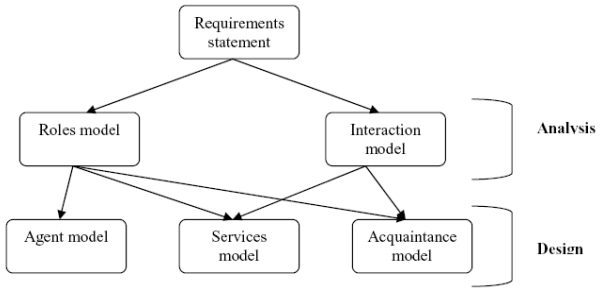

Fig. 1. Gaia Stages

and creating organizations [1].

Gaia divided the process of designing software into two main stages (Figure 1): analysis and design. The Analysis stage includes two internal steps, which are building Role Model and Interaction Model in order to build the conceptual models of the target system. While Design stage includes two internal steps, which are building Agent Model, Services Model, and Acquaintance Mode in order to transform the abstract constructs to well-defined entities that can directly be mapped to the implementation code [4].

Another version of this methodology was built and called Gaia v.2 included three additional models that are Environmental Model, Organizational Rules and

Open Science Index, Computer and Information Engineering Vol:4, No:4, 2010 publications.waset.org/5825.pdf

Organizational Structures. After that ROADMAP methodology was created which is considered as an extended version of Gaia. It focused on dynamic role hierarchy, additional models for agent's environment, and the agent knowledge concepts [8].

B. Tropos

Tropos methodology was developed by a group of researchers from various universities in Canada and Italy for agent-based system development with strong focus on early requirements analysis where the domain stakeholders are deeply analyzed [4].

Tropos divided the process of designing software into four main stages: Early Requirements Analysis stage focuses on the intentions of the system's stakeholders that modeled as goals. Late Requirements Analysis stage focuses on building the Strategic Dependency Model between the system's components. Architectural Design stage builds a preliminary model of system structure that describes how system components will work together. Detailed Design offers additional detail for each architectural component of a system [8][6].

C. MaSE

Wood and DeLoach [1][9] presented Multi-agent Systems Engineering Methodology (MaSE) methodology as a general purpose technique that supports application domain and automatic code creation through the MaSE tools. The goal of MaSE is to provide us with a complete methodology in order to lead the designer from the initial system specification to the implemented system [1][9][7].

MaSE divided the process of designing software into two stages (Figure 2): The Analysis phase consists of three steps: Capturing Goals that includes the Goal Hierarchy model, Applying Use Cases that includes the Use Case and Sequence Diagrams, and Refining Roles that includes the Role Model and Concurrent Tasks Model. The Design phase consists of four steps: Creating Agent Classes that includes the Agent Model, Constructing Conversations that includes the Conversations Model, Assembling Agent Classes that includes the Agent Architecture Model, and System Design that includes the Development Diagram [4][9][10][7].

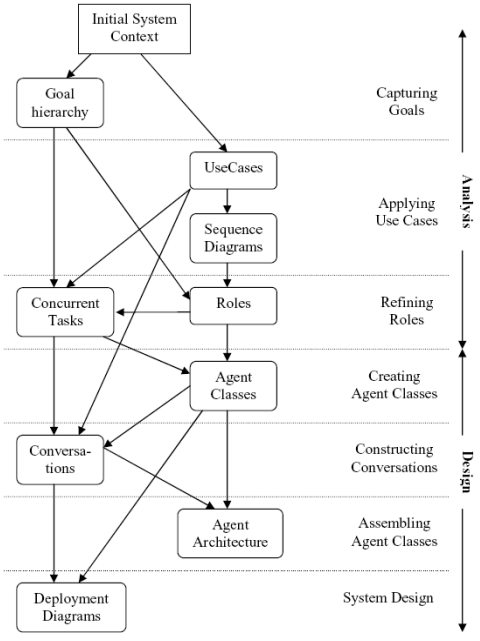

Fig. 2. MaSE stages

III. Extending MaSE

The choice of working on MaSE methodology and improving its features is based on two main reasons. First, it is one of the most common mature general purposes methodologies [1]. Second, its structure and diagrams are similar to those used in the main stream software engineering methodologies, and it uses (or extend) standard and well-formed diagrams. The introduced modifications to the MaSE methodology led to a change in its original flow. The new modified flow is shown in figure 3 (notice the bold boxes and lines and compare with figure 2). As mentioned earlier, the extensions include integrating early requirements handling and incorporating ontology concepts.

A. Integrating early requirements specification into MaSE

MaSE methodology doesn't concentrate on the system requirements, and it deals with the requirements as a separated phase from the software analysis process, while other methodologies (like Tropos) may not have MaSE's flexibility and modeling power but they focuses deeply on the early requirements definitions [8]. The first part of this work integrates early requirements stage into MaSE's process of designing software as a new stage. This will divide the process of designing software in MaSE into three stages instead of two (figure 3).

To achieve that, the following process was followed:

- Research was conducted to find the most advanced environments of MaSE.
- AgentTool3 was selected. It is a special tool developed by the Multi-agent & Cooperative Robotics Laboratory [11]. This tool offers highly rich interfaces with extra features for supporting MaSE diagrams [10] [12].
- Research was conducted to find the most advanced environments that support early requirements diagrams. Concepts similar to Tropos's early requirements diagram were favoured for modelling the final system's requirements.
- SI* tool [13], which is a special environment that supports Tropos methodology was selected. Using this tool allows the creation of the early requirements diagram which defines system scope and all objects inside it (agents, roles, goals…). Besides, we will be able to define how all of those objects connects/interacts with each other (section 4).
- Finally, transformation rules from SI* representation to MaSE (AgentTool3) representation were developed, and a special software was built to handle all objects, relationships and concepts in the early requirements diagram, and then, accordingly, generate preliminary Agents, Goals and Roles diagrams as required in MaSE environment. This work can help the designers to build concrete MaSE models consistent with each other. The main steps we conducted were:
  - Checking the XML format of SI* tool's early requirements diagram, and analyzing all relationships between its objects.

- Checking the XML format of AgentTool's Agents, Goals and Roles diagrams, and analyzing all relationships between their objects.
- Creating all needed transformation rules (summarized in section 4) to transfer all objects, relationships and concepts from the early requirements diagram to the Agents, Goals and Roles diagrams.

The process mentioned above allows designers to generate preliminary Agents, Goals and Roles diagrams based on the early requirements diagram and use them directly. Additionally, the tool generates a note file for all relationships that did not map into any of the Agents, Goals or Roles diagrams. These notes provide the information which designers might require to modify the Agents, Goals or Roles diagrams if they needed to do that.

This work makes modifications on the software designing process of MaSE by adding a special phase for requirements and adding Preliminary Roles inside the Refining Roles step. This step allows designers, if they like, to check the generated diagrams in order to update them in case any additional information needs to be presented in them.

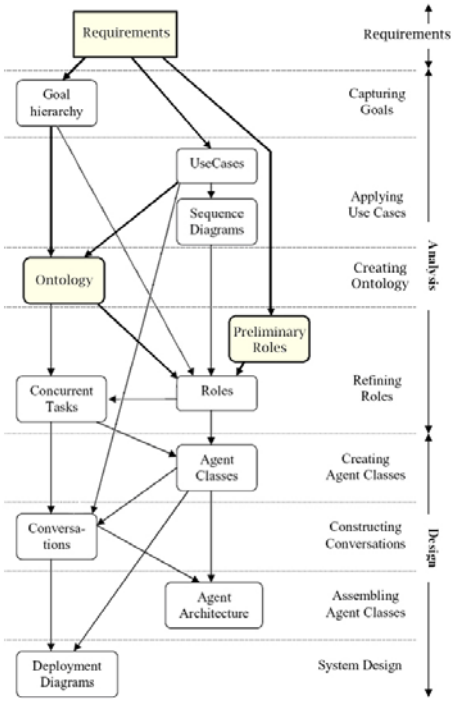

Fig. 3. The extended MaSE stages

B. Integrating ontology concepts into MaSE

Unfortunately, most existing AOSE methodologies (including MaSE) do not pay enough attention to information domain specification for multi-agent systems and for the agents inside the system. The interaction between agents in the system usually occurs through communications which is usually defined using the standard agent communication protocols. Systems usually require passing all kinds of information between agents in the form of parameters. However, without specifying the information domain of the system, designers will not be able to specify exactly what types of information need to be passed as parameters and will not be able to describe the information flow between the system's agents [14].

In order to use ontologies which describe the domain of information inside a system in MaSE, this work suggests adding a new step which is named Creating Ontology, inside the Analysis stage. This step allows designers to construct the final system's ontology that can be used inside the other steps. This step should occur after the Applying Use Cases step in order to specify exactly what types of parameters are needed to be passed between agents, goals, tasks …etc.

To build the ontology the following process is advocated:

- Defining the Ontology's Scope: in this step, the designer must specify what the main concepts of ontology are and what objects will use them inside the system.
- Checking Old Diagrams: in this step, the designer must check all objects in all previous diagrams (Goal Hierarchy، Use Cases، Sequence Diagram) in order to find all types needed to be defined in the ontology so all objects in that diagram can be described.
- Defining Levels: based on the final system scope, the designer can define exact levels of detail to describe all objects inside the system.
- Describing Preliminary Ontology: in this step, the designer will be able to define the main types in the ontology, and for each type he/she can specify exactly what features need to be presented.
- Searching for an Old Ontology: in this step, the designer must search for any old ontology that can meet all or some of the system's types. This will help designers integrating new systems with other existing systems.
- Refining Ontology: in this step, the designer must check the built ontology to ensure that it is totally sound and meets final system's requirements.

To support this process, a special software tool is underdevelopment (figure 4). This tool allows designers to define all types and relationships between them in order to identify the hierarchy of all needed types in the ontology. On the other hand, the tool allows designers to define types in detail including their features and axioms.

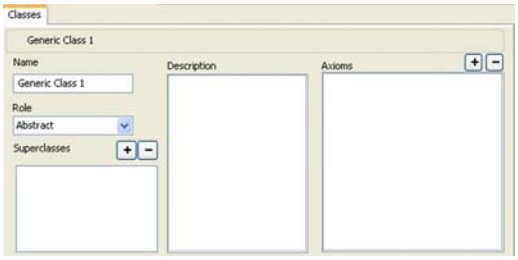

Fig. 4. The Ontology support tool

IV. Transformation Rules

The transformation rules which were developed in this work are thorough and lengthy. This why it's difficult to include them all in this paper.

Tables I, II and III present samples of the most significant of these rules which were used for transferring objects/relationships from the early requirements diagram in

Open Science Index, Computer and Information Engineering Vol:4, No:4, 2010 publications.waset.org/5825.pdf

the SI* platform to the Agents, Goals and Roles diagram in the AgentTool3.

TABLE I
MAIN TRANSFORMATION RULES IN THE AGENTS DIAGRAM

Name	Instead of	Transformation type	Description
Agent	Agent	Totally and direct.	Each Agent in the SI* platform will be replaced with an Agent in the agents diagram.
Capability	Resource	Totally but not direct.	Each Resource in the SI* platform will be replaced with a Capability in the agents diagram.

TABLE III
MAIN TRANSFORMATION RULES IN THE GOALS DIAGRAM

Name	Instead of	Transformation type	Description
AND	Composition	Partially and not direct.	Each 'Composition' relation have "And" feature in the SI* platform will be replaced with a 'AND' relation in the goals diagram.
OR	Composition	Partially and not direct.	Each 'Composition' relation have "Or" feature in the SI* platform will be replaced with a 'OR' relation in the goals diagram.

TABLE IIIII
MAIN TRANSFORMATION RULES IN THE ROLES DIAGRAM

Name	Instead of	Transformation type	Description
Service	Task	Totally but not direct.	Each Task in the SI* platform will be replaced with a Service in the roles diagram.
Provides	Provide (only from Agents to Tasks)	Partially and not direct.	Each 'Provide' relation from Agents to Tasks in the SI* platform will be replaced with an 'Provides' relation in the roles diagram.

Notes:

- "Totally" transformation means that the component/relation in the SI* platform will be copied totally to one of the AgentTool3 diagrams at least.
- "Partially" transformation means that the component/relation in the SI* platform will be copied to one of the AgentTool3 diagrams at least but based on some conditions in the transformation rule.
- "Direct" transformation means that the component/relation in the SI* platform will be copied as is to one of the AgentTool3 diagrams at least.
- "Not Direct" transformation means that the component/relation in the SI* platform will be copied as another component/relation to one of the AgentTool3 diagrams at least.

V. CASE STUDY

The conference management system has become fairly common in AOSE articles [8]. This system is used here as a case study to demonstrate how the suggested transformation rules, and the tool which was developed in this work can be used to generate all needed MaSE diagrams efficiently starting from a clear early requirements diagram.

The conference management system is responsible for managing various sized international conferences and the flow of evaluation for research papers and it requires coordination of several individuals and groups (authors, reviewers, decision makers, review etc.).

The conference management system is an organization (agents' organization according to AOSE terminology) and it will work with the authors' organization that used to be considered as a resource to it, because authors used to write their papers and send them to the conference management organization. The organizations' members will get the papers and store them in the DB. After that they send them to evaluators who will check the papers and evaluate them. Then, papers will be forwarded to a decision maker in order to check, review and accept or reject them. Then the final decision will be forwarded to notifiers who will inform the authors about the final decision.

First, the early requirements diagram needs to be drawn and the following issues need to be considered:

1) The agents of this system are the Authors, Members and Chairmen (decision makers).

2) The roles of this system are the Papers' Managers, Notifiers, Reviewers and Decision Makers.

3) Authors will write a paper and send it to the conference management organization by using a suitable sending tool.

4) Papers' Managers will receive the papers and store them in the conference's archiving system. Then they will send them to the Reviewers in order to evaluate them. After that Reviewers will send them to one of the Decision Makers which make the final decision about each paper. Decision Makers will then send them to the Notifiers who will send the final results to the Authors.

5) The Chairmen agent inherits from the Members agent.

The final requirements diagram is presented in figure 5.

Open Science Index, Computer and Information Engineering Vol:4, No:4, 2010 publications.waset.org/5825.pdf

Open Science Index, Computer and Information Engineering Vol:4, No:4, 2010 publications.waset.org/5825.pdf

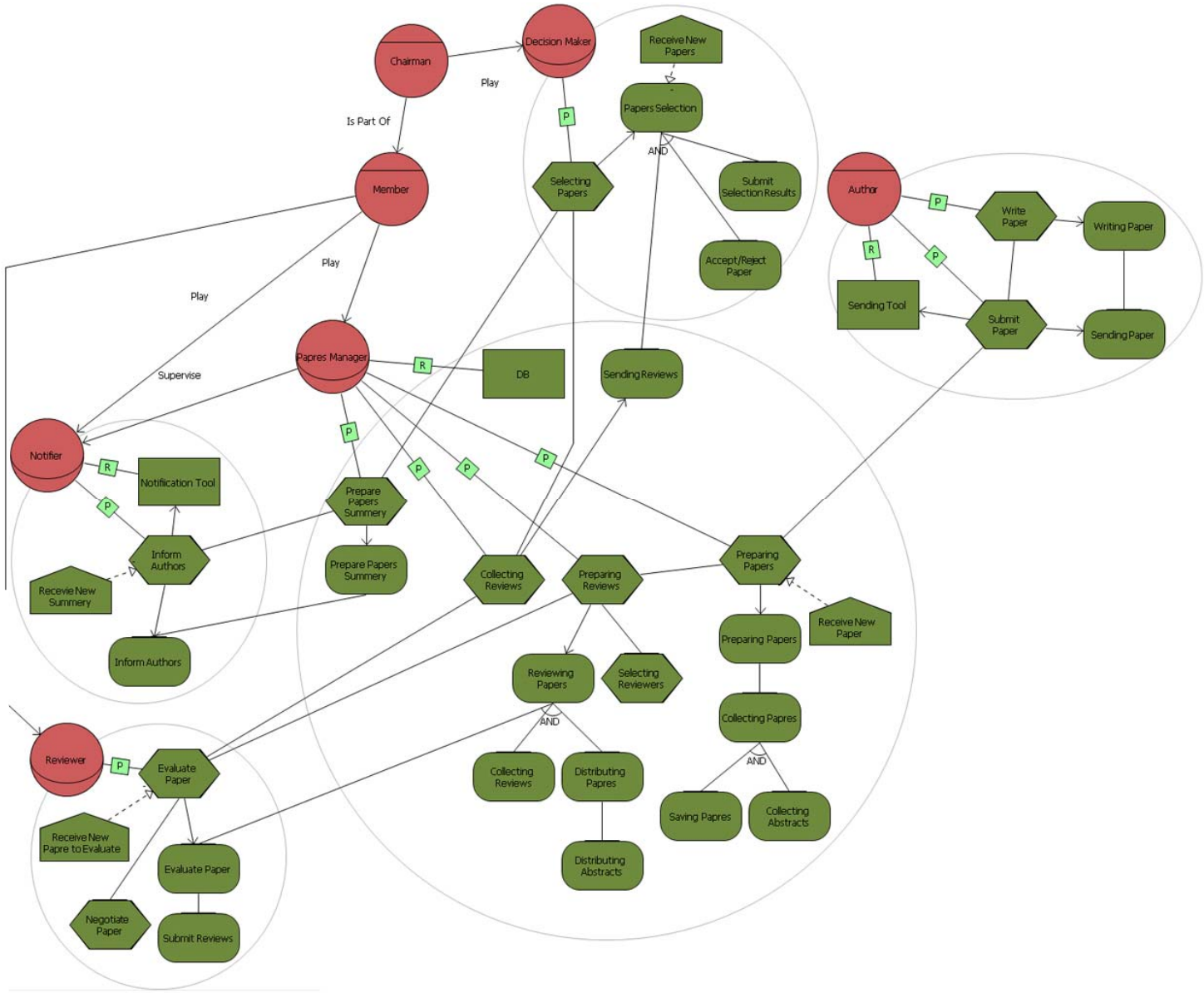

Fig. 5. The early requirements diagram

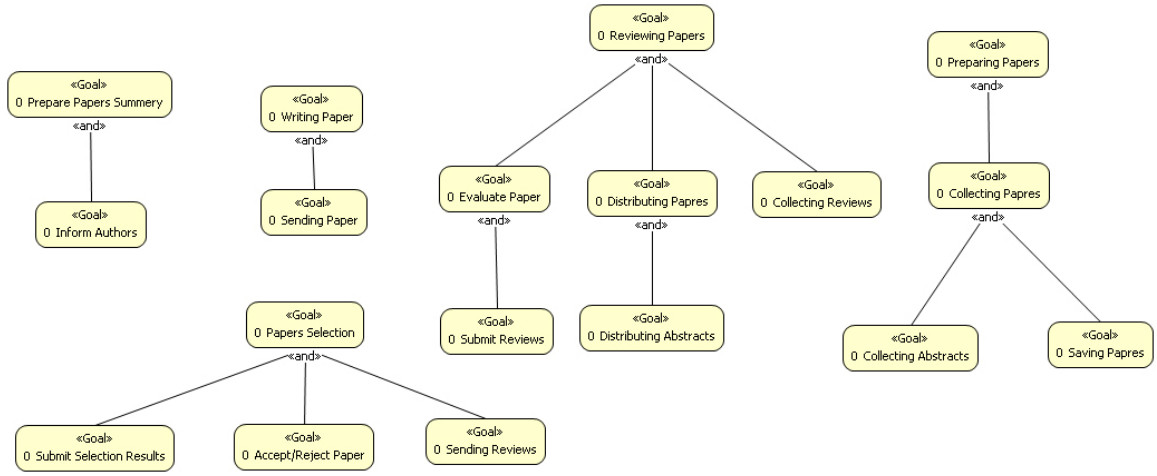

Fig. 6. The Goals diagram

After drawing the early requirements diagram, the tool developed in this work can be used to generate Agents, Goals and Roles diagrams. The result is shown in figures 6, 7 and 8. Figure 6 presents the generated Goals diagram, while Figure 7 describes the generated Agents diagram, and Figure 8 describes the generated Roles diagram.

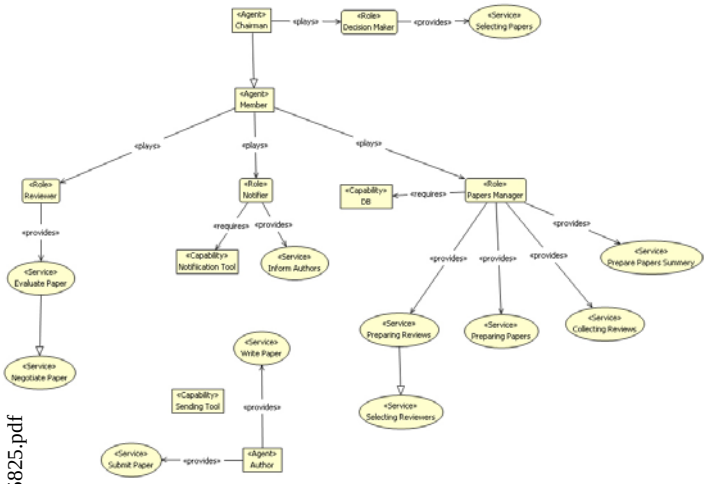

Figure 7. The Agents diagram

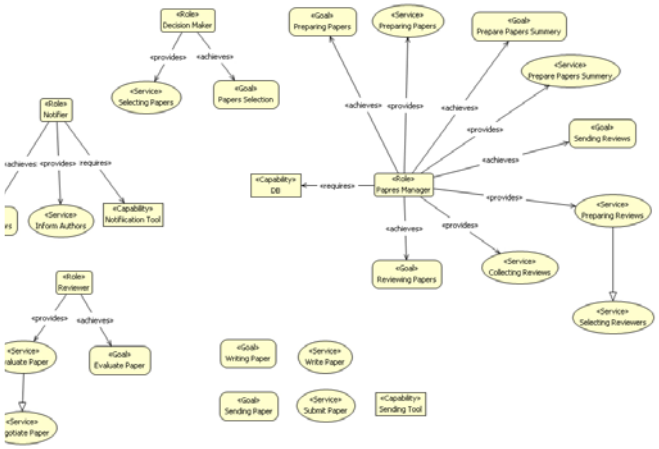

Fig. 8. The Roles diagram

In addition, the tool generates a note file which includes notes about all components/relationships that didn't appear in any of MaSE diagrams. Figure 9 shows a sample of that file.

VI. Conclusions and Future Work

This paper proposes some modifications to the MaSE methodology by integrating early requirement specification and ontology concepts into the standard flow of a well established methodology. The work also proposes a list of transformation rules that should be used in the transformation tool in order to transfer all information in the early requirement diagram to the Agents, Goals and Roles diagrams of MaSE. In addition, a work was conducted on integrating ontology concepts on the MaSE flow in order to clearly define the types of all used objects in all diagrams.

Several open points still remain, especially in the future usage of the MaSE (especially after integrating early requirements and ontology concepts in it) which may concentrate on:

- Symantec web: in order to create a great network of intelligent agents that interacts together to do many sophisticated tasks.
- Developing open systems: that will allow developers to create any kind of agents and allow them to coordinate with other agents in order to do some special tasks of the system. But before that, agents must be checked and confirmed that they satisfied all needed working rules of the system.

```
Event: Name(Receive New Paper)↓
Event: Name(Receive New Papers)↓
Event: Name(Recevie New Summery)↓
Event: Name(Receive New Papre to Evaluate)↓
Supervise Relation: Source(Papres Manager)[Role], Target(Notifier)[Role]↓
Impact Relation: Source(Receive New Paper)[Event], Target(Preparing Papers)[Task]↓
Impact Relation: Source(Recevie New Summery)[Event], Target(Inform Authors)[Task]↓
Impact Relation: Source(Receive New Papre to Evaluate)[Event], Target(Evaluate Paper)[Task]↓
Impact Relation: Source(Receive New Papers)[Event], Target(Papers Selection)[Goal]↓
Means Ends Relation: Source(Write Paper)[Task], Target(Writing Paper)[Goal]↓
Means Ends Relation: Source(Submit Paper)[Task], Target(Sending Paper)[Goal]↓
Means Ends Relation: Source(Submit Paper)[Task], Target(Sending Tool)[Resource]↓
Means Ends Relation: Source(Preparing Papers)[Task], Target(Preparing Papers)[Goal]↓
Means Ends Relation: Source(Selecting Papers)[Task], Target(Papers Selection)[Goal]↓
Means Ends Relation: Source(Preparing Reviews)[Task], Target(Reviewing Papers)[Goal]↓
Means Ends Relation: Source(Evaluate Paper)[Task], Target(Evaluate Paper)[Goal]↓
Means Ends Relation: Source(Collecting Reviews)[Task], Target(Sending Reviews)[Goal]↓
Means Ends Relation: Source(Prepare Papers Summery)[Task], Target(Prepare Papers Summery)[Goal]↓
Means Ends Relation: Source(Inform Authors)[Task], Target(Inform Authors)[Goal]↓
Means Ends Relation: Source(Inform Authors)[Task], Target(Notifiication Tool)[Resource]↓
Custom Relation (Ordering Tasks): Source(Write Paper)[Task], Target(Submit Paper)[Task]↓
Custom Relation (Ordering Tasks): Source(Submit Paper)[Task], Target(Preparing Papers)[Task]↓
Custom Relation (Ordering Tasks): Source(Preparing Reviews)[Task], Target(Evaluate Paper)[Task]↓
Custom Relation (Ordering Tasks): Source(Evaluate Paper)[Task], Target(Collecting Reviews)[Task]↓
Custom Relation (Ordering Tasks): Source(Preparing Papers)[Task], Target(Preparing Reviews)[Task]↓
Custom Relation (Ordering Tasks): Source(Collecting Reviews)[Task], Target(Selecting Papers)[Task]↓
Custom Relation (Ordering Tasks): Source(Selecting Papers)[Task], Target(Prepare Papers Summery)[Task]↓
Custom Relation (Ordering Tasks): Source(Prepare Papers Summery)[Task], Target(Inform Authors)[Task]↓
Request Relation: Source(Auther)[Agent], Target(Sending Tool)[Resource]↓
```

Fig. 9. The note file

References

[1] Amund Tveit. A Survey Of Agent-Oriented Software Engineering, NTNU Computer Science Graduate Student Conference, Trondheim, Norway - May 2001.
[2] Nicholas R. Jennings and Michael Wooldridge. Agent-Oriented Software Engineering. 2001 AAAI/MIT Press.
[3] Brian Henderson-Sellers and Ian Gorton, Agent-Based Software Development Methodologies, Workshop on Agent-oriented Methodologies at OOPSLA 2002, Seattle, USA, November 2002.
[4] Khanh Hoa Dam. Evaluating And Comparing Agent-Oriented Software Engineering Methodologies, RMIT University – Melbourne, Australia, Master thesis, September 2003.
[5] Franco Zambonelli Nicholas, R. Jenningsy and Michael Wooldridge, Developing Multiagent Systems: The Gaia Methodology 2003.
[6] http://www.troposproject.org/ last Jan 2010.
[7] http://macr.cis.ksu.edu/ last Jan 2010.
[8] Federico Bergenti, Marie-Pierre Gleizes, and Franco Zambonelli, METHODOLOGIES AND SOFTWARE ENGINEERING FOR AGENT SYSTEMS, 2004.
[9] Mark F. Wood and Scott A. DeLoach, An Overview of the Multiagent Systems Engineering Methodology, First International Workshop, AOSE 2000, Limerick, Ireland.
[10] Scott A. DeLoach, Analysis and Design using MaSE and agentTool, 12th Midwest Artificial Intelligence and Cognitive Science Conference (MAICS 2001) Miami University, Oxford, Ohio.
[11] http://agenttool.cis.ksu.edu/, last update 01 October 2009.
[12] Juan C. Garcia-Ojeda, Scott A. DeLoach and Robby, agentTool III: From Process Definition to Code Generation, Kansas State University 2009.
[13] http://sesa.dit.unitn.it/sistar_tool/home.php?7, last update 2009.
[14] Jonathan DiLeo, Timothy Jacobs, Scott DeLoach, Integrating Ontologies into Multiagent Systems Engineering.

Open Science Index, Computer and Information Engineering Vol:4, No:4, 2010 publications.waset.org/5825.pdf