# TOWARDS A UNIFIED AGENT ORIENTED SOFTWARE ENGINEERING METHODOLOGY

**A DOCTOR OF PHILOSOPHY THESIS**

**In**

**THE DEPARTMENT OF MODELING AND DESIGN OF ENGINEERING SYSTEMS**

**(Main Fields of Study:Software Engineering)**

**Atilim University**

**by**

**ABDALLA REEM**

**JUNE 2018**

# TOWARDS A UNIFIED AGENT ORIENTED SOFTWARE ENGINEERING METHODOLOGY

**A THESIS SUBMITTED TO**

**THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES**

**OF**

**ATILIM UNIVERSITY**

**BY**

**REEM ABDALLA**

**IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF**

**DOCTOR OF PHILOSOPHY**

**IN**

**THE DEPARTMENT OF MODELING AND DESIGN OF ENGINEERING SYSTEMS**

**(Main Fields of Study: Software Engineering)**

**JUNE 2018**

Approval of the Graduate School of Natural and Applied Sciences, Atilim University.

_____

Prof. .Dr. Ali Kara

Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Doctor of Philosophy.

_____

Assoc. Prof. Dr. Ender Keskinkılıç

Head of Department

This is to certify that we have read the thesis "Towards A Unified Agent Oriented Software Engineering Methodology" submitted by "ReemAbdalla" and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Doctor of Philosophy.

_____

Prof. Dr. Alok Mishra

Supervisor

Examining Committee Members

Prof. Dr. Alok Mishra _____

Associate Prof. Dr.Reza Hassanpour _____

Dr. Öğr.Üyesi Meltem Eryilmaz _____

Prof.Dr.Erdoğan Doğdu _____

Dr. Öğr.Üyesi Çiğdem Turhan _____

Date: 22/6/2018

I declare and guarantee that all data, knowledge and information in this document has been obtained, processed and presented in accordance with academic rules and ethical conduct. Based on these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name: REEM ABDALLA

Signature

## ABSTRACT

### TOWARDS A UNIFIED AGENT ORIENTED SOFTWARE ENGINEERING METHODOLOGY

Abdalla, Reem

Ph.D., Modeling and Design of Engineering Systems Department (Software Engineering)

Supervisor: Prof.Dr. Alok Mishra

June 2018, 153 pages

The computer science and software engineering society have become one of the most progressive and significant domains within science as they have managed to stand firmly in the information technology since the 1990s. Throughout recent years agent technologies have been promoted rapidly and a rising number of processes, frameworks, and notions recommended in related scientific research.

In turn, agent technology has many implementations and uses in problem-solving in a variety of areas, including shopping simulations, surveillance, system diagnosis, and remedial procedures. Over the last years, in particular, there has been an increasing number of agent approaches proposed along with an ever-growing interest in their various implementations. These approaches are evolutionary and specifically designed for agents and their particular properties. Yet, a comprehensive and full-fledged agent approach for developing related projects is still absent despite the presence of numerous agent-oriented methodologies. One of the moves towards

compensating for this issue is to compile the models of various available methodologies, ones that are comparable to the evolution of the Unified Modeling Language in the domain of object-oriented analysis and design. As these to have become defacto standards in software development.

In line with this purpose, the present thesis attempts to comprehend the relationship among seven main agent-oriented methodologies. More specifically, we intend to assess and compare these seven approaches by conducting a feature analysis through examining the advantages and disadvantages of each competing process, structural analysis, a case study, and meta-model evaluation methods. This effort is made to address the important characteristics of agent approaches.

Since the main objective of this thesis is to take a step forward in forming a unified agent-oriented software engineering methodology, we will include the positive features and avoid the undesired ones within seven distinguished agent methodologies.

Keywords: Agent methodologies,Structural analysis,Case study,Meta modeling

# ÖZ

## TÜMLEŞİK AJANA DAYALI BİR YAZILIM MÜHENDİSLİĞİ METODOLOJİSİNE DOĞRU

Abdalla, Reem

Ph.D., Mühendislik Sistemlerinin Modellenmesi ve Tasarımı Bölümü

(Yazilim Mühendisliği)

Akademik Danışman: Prof.Dr. Alok Mishra

Haziran 2018, 153 sayfa

Bilgisayar bilimi ve yazılım mühendisliği topluluğu, 1990'lı yıllardan beri bilgi teknolojisi alanında sağlam bir yer edindikleri ve bunu devam ettirmeye başardıkları için bilimde en ilerici ve en önemli alanlardan biri haline gelmiştir. Ajan teknolojileri son yıllarda hızla gelişmiş olup ilgili bilimsel araştırmalarda tavsiye edilen işlem, çalışma alanı ve kavramların sayısı artış göstermiştir.

Dolayısıyla ajan teknolojisi, alışveriş simülasyonları, gözetim, sistem hata bulucuları ve onarım prosedürleri dâhil olmak üzere çeşitli alanlarda problem çözme konusunda birçok uygulamaya ve kullanıma sahiptir. Özellikle son yıllarda sürekli büyüyen ilgi ve alaka ile birlikte çeşitli uygulamalarda ajan yaklaşımları artış göstermiştir. Bu yaklaşımlar evrimseldir ve özellikle ajanlar ve onların belirli özellikleri için tasarlanmıştır. Bununla birlikte, çok sayıda ajan odaklı metodolojilerin varlığına rağmen ilgili projelerin geliştirilmesi adına kapsamlı ve tam teşekküllü bir ajan yaklaşımı hâlihazırda mevcut değildir. Bu konuyu dengelemeye yönelik hamlelerden birisi, nesneye dayalı analiz ve tasarım alanında Tümleşik Modelleme Dili'nin evrimi ile karşılaştırılabilir çeşitli mevcut metodolojilere ait modelleri derlemektir.

Bu amaç doğrultusunda işbu tez, yedi temel ajana dayalı metodolojiler arasındaki ilişkiyi kavramaya çalışmaktadır. Her bir yarışan/çakışan sürecin avantaj ve dezavantajlarını, yapısal analizi, vaka çalışması ve meta-model değerlendirme yöntemlerini inceleyip bir özellik analizi yaparak bu yedi yaklaşımı daha belirgin bir biçimde değerlendirmeyi ve karşılaştırmayı hedeflemekteyiz. Söz konusu çalışma ajan yaklaşımlarının önemli özelliklerini ele almak üzere yapılmıştır.

İşbu tezin temel hedefi tümleşik ajana dayalı bir yazılım mühendisliği metodolojisini bir adım ileriye taşımak olduğundan olumlu özellikleri dahil edecek ve bilinen yedi ajan metodolojisinde istenmeyen özellikleri ele almayacağız.

Anahtar kelimeler: Ajan metodolojileri, Yapısal analiz, Vaka Çalışması, Meta modelleme

To My Dear Parents,

## ACKNOWLEDGEMENTS

Above all, without the grace of Allah, I could not have attained anything in my life,the ,all praises to Allah for the strengths and his blessing in completing this thesis.

I would especially like to thank my advisor, Prof.Dr. Alok Mishra, for his generous time and commitment. Throughout my thesis work he encouraged me to develop independent thinking and research skills. Prof.Dr. Mishra has been a great source of encouragement and inspiration to me. Without his support this dissertation would not have been written. I offer sincere thanks to my thesis jury members Associate Prof. Dr. Reza Hassanpour and Dr. Öğr.Üyesi Meltem Eryilmaz for their guidance in evaluation and review of my study.

I am very grateful to my dear husband Mahmoud, my lovely kids Moneeb, Awab and Mohammed and my brothers for their encouragement, patience, and understanding throughout my study journey.

**TABLE OF CONTENTS**

## LIST OF TABLES

## LIST OF FIGURES

## LIST OF ABBREVIATIONS

IAC - Inter-Agent Control

MAS - Multi-Agent System

IP - Interaction Protocol

UCS - Use Case Scenarios

UC - Use Case

AOSE - Agent Oriented Software Engineering

RUP - Rational Unified Process

UML - Unified Modling Language

MDE - Model-Driven Engineering

AUML - Agent Unified Modling Language

FIPA - Foundation for Intelligent Physical Agents

BDI - Belief-Desire-Intention

OO - Object Oriented

AO - Agent Oriented

KE - Knowledge Engineering

OMG - Object Management Group's

MaSE - Multi-Agent Systems Engineering

NIMSAD - Normative Information Model-based System Analysis and Design

DESMET - Determining an Evaluation Methodology for Software Methods Tools

GA - Guardian Angle

GAPCHIS - Guardian Angle Patient-Centered Health Information System

OMACS - Organization Model for Adaptive Computational Systems

AMAS - Adaptive Multi-Agent System

NCS - Non Cooperative Situations

AMOLA - Agent Modeling Language

EMF - Eclipse Modeling Framework

GRL - Goal-Oriented Requirement Language

# CHAPTER 1

## INTRODUCTION

### 1.1 Overview

In the 1970s, structured programming first emerged as an approach to software development. Later in the 1980s, Object Oriented (OO) languages came into the spotlight with certain novel applications such as data encapsulation, inheritage, and polymorphism. These technologies were followed by several paradigms as component-ware, design patterns, and application frameworks which expanded object-oriented software engineering even further.

Although the OO approach has become one of the most popular modeling methods and has achieved a large degree of maturity, today systems in use worldwide are constantly changing. To be able to respond to such ever-growing change in different aspects such as organizational frameworks, business practices, market drives, and other forms of organized initiatives as well as the related and necessary educational shifts, consistent frameworks, models, methods, and tools are to be developed to support these requirements.

In recent years, more and more methodologies intended for software engineering have started to emerge (Iglesias et al, 1992) thanks to efforts by researchers. In general, the purpose of designing and utilizing software development approaches is to shift the evolution from a specific practice to a well-arranged engineering activity that yields

high-quality software within restricted resources and despite constraints in terms of finance and time.

With this in the background, researchers continue to discover and bring about new and powerful software engineering techniques for more efficient operations and to produce solutions to the complexities in applications. With the increasing importance of such systems in the industry, our need for employing agent technologies to promote commercial and industrial software systems is rising rapidly as well and moving the industry from a form of product for a user to a manner of delegation, where it can also involve the user in order to make decisions and take actions accordingly. Since agent-oriented software engineering (AOSE) is vastly acknowledged as researchers expected, it is also equally important to develop mature tools and methodologies to keep up with the efforts.

There are a number of agent-based projects already available in fields such as the football player, e-commerce, business process management, air traffic control, etc. (Luck.et al, 2003). These projects were produced without the backing of agent-oriented methodologies. However, the existing methodologies may not be mature enough to be used in developing these systems(Nicholas. et al, 1998).

To present more elastic systems for complicated applications, a vast number of agent approaches have been proposed in the software engineering area, alongside other novel properties such as independence and proactiveness. The main purpose for these methodologies is to deliver the required processes, models, mechanisms and tools so that agent-based systems can be developed in a more formal and organized manner. All in all, it is not easy to define an exact and concrete method for developing any project. Also, there are no ways to determine what features and disadvantages each approach may be likely to possess. This situation is similar to the development of methodologies and concepts of objects carried out before the Unified Molding Language (UML) came in to existence.

We view that the time has come to work towards a unified AOSE methodology. There are two causes that drive us to this conclusion: First, the overspreading of different methods has brought developers to the point where they have to choose one

methodology from all AOSE approaches. Second, while a few of such methods may be mature and described in elaborate detail just a few years before, at present there are numerous other methods that can be beneficial to developers as they have been revised through extensive use and also enjoy support tools.

Currently, there is a necessity to develop a unified AOSE process and the area is sufficiently well-developed to allow for an acceptable integrated approach to be designed. Before any other discussions, however, it is critical to realize what is meant by the term "methodology". According to Dam & Winikoff (2013), a methodology includes what the developers desire to provide back up in the development and design of agent-based systems. In particular, a methodology is far more comprehensive and does not serve as a notion; it also conveys a clear picture of what is to be designed in order to utilize the notations, a thorough process, and detailed guidelines concerning the activities of that process.

In this respect, the supporting tool is highly desired but not universally considered as a complete segment to the approach. The main purpose for these methodologies is to deliver the required processes, models, mechanisms and tools so that agent-based systems can be developed in a more formal and organized manner.

The major objective of this work is to propose an agent-oriented software methodology that reduces the shortcomings of agent methodologies according to a comparative analysis of the seven main AOSE methodologies: Gaia, Tropos, PASSI, O-MaSE, ADELFE, ASME, and Prometheus, in order to distinguish each one’s strengths, weaknesses, commonalities and variations. A comparative analysis of these competing methodologies will be provided in order to present well-based and properly calculated ideas for the paradigms and activities that constitute a unified methodology. This is expected to help understand the connection among these different methodologies, including features, drawbacks, and range of applicability.
In all, the purpose of this thesis is to move one step closer to a standard approach by suggesting a joint process comprising a group of models.

We do not intent to present a definitive standard methodology, but to propose a preliminary and draft framework, which we hope will drive further discussion in the

field and that, in the end, this argumentation and cooperation may lead to an actual integrated AOSE approach.

### 1.2 Selecting Methodologies

As mentioned earlier, there are numerous methodologies for developing agent-based systems available in the literature. For this reason, we will adopt a multi-phase selection method (Law,1988). The primary step minimizes the set of approaches by utilizing the standard criteria that follow:

**Documentation:** The document of methodology must be clear and elaborate in characteristic, not just a presentation at a conference but cited and included in books, journal papers, or other well-known technical reports.

**Maturity:** The methodology chosen has to have extensive use and improvement over time, involving use by other non-authors and their colleagues and students.

**Tool support:** The approach has to be preferably have backing tools rather than not. Only chosen methodologies are to be applied to develop a particular project (Dam & Winikoff, 2004), for which reason the presence of tool support is an effective attribute**.**

In further detail, the following are the methodologies selected for the present work:
Gaia (Zambonelli et al., 2005) which is particularly designed for the development of agent-based systems; Tropos (Giorgini et al., 2005) which is a detailed agent-oriented software engineering methodology and which supports a broad domain of software life-cycle processes; PASSI (Cossentino, 2005) for planning and the evolution of multi-agent communities to integrate design models; O-MaSE (DeLoach & Garcia-Ojeda, 2014) which is a customizable agent-oriented approach based on compatible and well-established ideas strengthened by additional applications in industrial development environments; ADELFE (Bernon et al, 2002) which is a multi-agent-oriented approach convenient for adaptive multi-agent systems; ASEME (Spanoudakis & Moraitis, 2011) which suggests a modular agent design approach

and presents the notions of control within the agent; and, finally, Prometheus (Padgham & Winikoff, 2005) that presents models and activities to construct smart agents using objectives, beliefs, plans and events.

These approaches support all of the standards set earlier: all of their documentations have appeared in journal papers and have support tools (except for Gaia). In term of maturity, O-MaSE, Prometheus, Tropos, Gaia, PASSI stand out as meeting the criteria, but the other two, ADELFE and ASEME, are bit weaker on the maturity criteria – to be more precise and detailed.

### 1.3 Aim and Objectives

The objective of this research is to make an attempt toward a standardized approach. A collection of artifacts and ideas is made from different key AOSE techniques. Yet, it is crucial to comprehend the relationship among them including the strengths, weaknesses and applicability of each methodology.

After analyzing and evaluating the selected methodologies, we can present our proposed methodology based on the results obtained through the application of methods for the evaluation of selected agent methodologies. The essential stages for this work can be detailed as follows:

- Firstly, examine relevant journal papers with greater depth to gain a better understanding of the exact nature of these competing approaches, including philosophies, objectives and features.

- Secondly, highlight the resemblance and distinguishing variances of the competing methodologies in terms of paradigms, mechanisms, and tools using structural analysis.

- Thirdly, carry out a case study by applying all the available methodologies on the same model.

- Fourthly, demonstrate common concepts by conducting an analysis of these Methodologies, comparing the meta-models related with each chosen approach and combining them into one meta-model.

- Fifthly and finally, propose proceedings and models based on the previous steps for a unified AOSE approach to meet the aim of the thesis. Figure 1 summarizes the essential steps for this thesis.

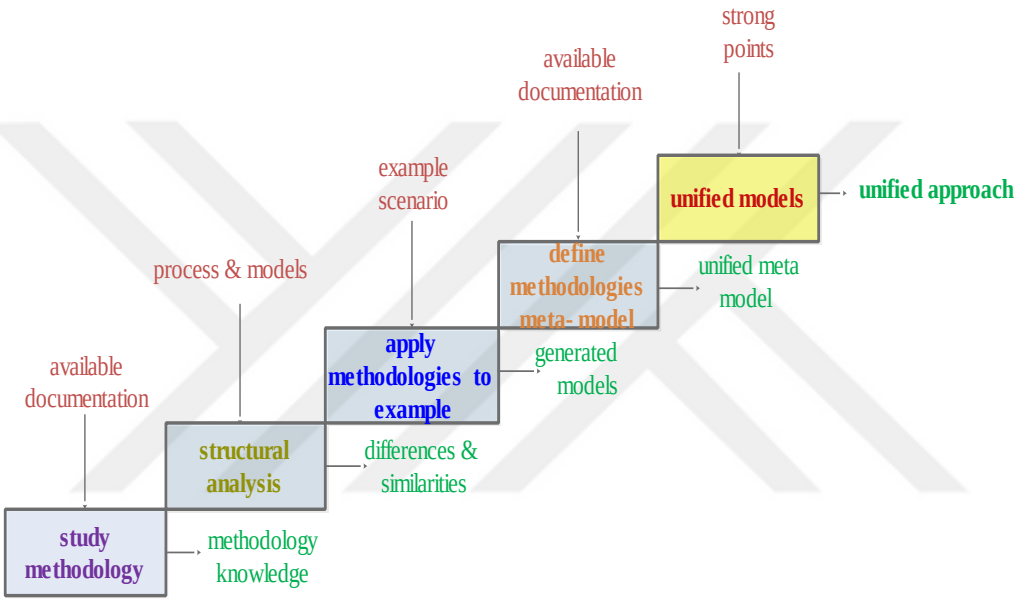


Figure 1. Sequence of research steps

### 1.4 Outline of Thesis

The work in this thesis is structured as follows:

Chapter 1: "Introduction" offers an introduction to the research and the work phases.

Chapter 2: "Litrature Review" provides a theoretical background for the research and the evaluation techniques applied to the methodologies.

Chapter 3: "Evalution Techniques part-1" describes the method (structal analysis) of assessment followed in this thesis involving a display of this technique for comparing software engineering approaches and a depiction of the criteria for the selection and application of methodologies.

Chapter 4: " Evaluation Techniques part-2" describes the methods (case study and meta models) of assessment followed in this thesis involving a display of current techniques for comparing software engineering approaches and a depiction of the criteria for the selection and application of methodologies.

Chapter 5: " Unifided AO Software Engineering Methodolgy" is the discussion and analyses of the outcomes of the valuation as well as a presentation of a proposal to integrate the advantages of each agent-oriented software engineering approaches.

Chapter 6: " Discussion and Conclusions" offers the summary of the research and propositions for future work.

# CHAPTER 2

# LITRATURE REVIEW

In this chapter, we will provide a background of studies on the issue subject to the thesis, and start with a general introduction to software engineering paradigms in section 2.1. A short overview on AO software engineering that includes agent concept, software, application, and development methodologies is given in section 2.2, followed by an overview of agent AOSE methodologies and their classifications in section 2.3. In section 2.4, we discuss and introduce a theoretical background for the research and the related work. Finally, a detail description will appear in section 2.5 of the existing techniques used for comparison and assessment of methodologies.

## 2.1 Software Engineering Technology

Software engineering processes were designed to facility and constitute the system lifecycle development; these technologies have become an important part of information systems in the last few decades, but building reliable, maintainable, and cost-effective software systems that satisfy stakeholders needs is not an easy task and these systems continually grow in size and complexity. In many situations, small teams of developers can no longer release new products on schedule due to the increasing number of features. Also, software development organizations usually use third-party libraries, components and services in their systems. More pressure isfaced

from clients to provide customized and easy-to-use systems that can be maintained quickly and cheaply. To minimize these difficulties, software engineering researches and studies focus on the methods and techniques that assist in the production of high-quality and robust systems within a suitable period of time and at a lesser cost. Several definitions have been offered for software engineering; for instance, IEEE defines software engineering as to apply a systematic and quantifiable methodology for program expansion, operation and maintenance; any engineering application on software (IEEE, 1993).

There are software engineering models designed as methodologies to devise quality programs, creating a vast array of models (e.g., procedural programming, arranged programming, demonstrative programming, object-oriented programming, styling manner, application framing & ingredient-ware, and many others). In fact, proposed software engineering models were generally proposed to facilitate the engineering procedure to produce software or to increase the complexity of applications to be structured. Because software systems are becoming more complex, researchers are constantly exploring more robust software engineering patterns (Vliet, 2000).

## 2.2 An Overview on Agent Technology

The purpose of this section is to give an overview of the common notions of agents and a comprehensive determination for agents in section 2.2.1. In section 2.2.2, we discuss AOSE and, later, agent applications in section 2.2.3.

### 2.2.1 Agent Concepts

In the preceding section, the concept of agent in general was discussed without addressing the details of the issues of technology. Perhaps, the most important issue to be discussed is ‘what an agent is’. In fact, because researchers and developers define agent from their own perspective, there is no agreed-upon definition yet, and it is not easy to find one unified definition despite the vast number of such definitions. Here, some of them are listed.

As to the Oxford dictionary, the agent is the one who have the authority to act for another or in the place of it. According to BradShaw (1997), an agent is a software entity, operating continuously and independently in a particular environment, which is often populated by other factors and operations. Quoting from Wooldridge & Jennings (1995), the agent is considered to be a computer system located in some environment, where the agent can take independent procedures in this surroundings to achieve the objectives. of the design. Maes (1995) in a general sense, states that software is an independent entity that senses and acts alone in the surrounding environment, and by doing so recognize a set of objects or tasks which is was designed for.

Corresponding of other software engineering paradigms, the agent required for continuity and independence must be able to carry out activities in a flexible manner and without reliance on humans. Also, agents that exist in the same environment must be able to cooperate. Agents that specialize in many characteristics, such as interaction, are able to feel and act in the environment and are socially able to cooperate. An agent can be categorized using various views (Brinkkemper, 1993) (Wooldridge & Jennings, 1995). The artificial intelligence community distinguishes two types of agents, weak and powerful, and the artificial intelligence community separates and classifies the related factors by degree of the ability for problem-solving (e.g., interactive agents and intentional agents). Agents are characterized by the following characteristics (Wooldridge & Jennings, 1995).

-Autonomy: that is, agents can work and make their own decisions by themselves
Reactivity :agents can recognize their environment and offer rapid response to changes to achieve the goals.

-Pro-activeness: agents are capable of pursuing their interests over the years and showcase purpose-directed conduct by taking the show on the way to gain their layout aims.

-Social ability: agents can interact with other agents so as to accept their objects.

-Situatedness: agents join within a specific environment. Sensors are used to study the environment and influence it with their own effects.

Powerful agency can be defined by the above as a weak agency but the agency sometimes attributes other advantages, such as movement, that is the agent's ability to get about around an electronic network; and honesty, that is agents do not provide wrong data and do not have discordant objects, Therefore, each agent constantly tries to do what it is being asked of. In all, the purpose of all agent characteristics is to provide flexible and strong entities. Flexibility is enabling agents to debate with other agents, and to benefit from any and all chances and changes, and to adapt to the environment. The capability of rehearsal, choice of action, interaction and following up goals combined will improve the agent' robustness.

These characteristics are very important in complex dynamic environments and, with the emergence of distributed information systems involving the Internet, agents are receiving more attention and becoming more popular; in the same way, software systems become more complex. Indeed, the concepts of autonomy, elasticity, and pro-activeness do not exist in conventional OO processes (Odell, 2002). Therefore, we need to have a technology that offers strong entities and has high intelligent characteristics to simplify distributed computing.

### 2.2.2 Agent-Oriented Software Engineering

The prime aims of AOSE is to build methodologies and tools that allow for the expansion and repair of agent-based software. It is also a point that the software should be adaptable, simple to-use, versatile and of high quality (Erol, et al., 2000). Jennings and Wooldridge have written many papers on AOSE in which they provide “intellectual justification” (Jennings, 2000) (Jennings &Wooldridge, 1999) for the reliability of the agent-oriented techniques. Their support, be that as it may, originates from a subjective investigation of how well the procedure tends to the rules that permit programming building methods to manage complex issues suggested by Booch: abstraction, decomposition, and hierarchy (Jennings &Wooldridge, 1999) (Booch, 1994). They leave “Understand where agent solutions are appropriate "as a

pending issue. Researchers have in front of them numerous challenges and have many goals to achieve.

It is possible that the most important one is to demonstrate that this technology offers a suitable treatment of the complexity of software systems. At present, most software products and models distributed are anticipated to react with ingredients and to take advantage of the tasks that available on the web. Yet, current technologies (e.g. OO technologies) fail to provide solutions for dealing with complex software systems. Additionally, specialists endeavor to develop proper instruments and reflections to speak to the new ideas presented by this new innovation. Some have been attempting to extend the UML (Unified Modeling Language) with new forms to make it appropriate for agents, while others take agent theory as their perspective and produce notations specific to agents.

Various agents take operator hypothesis as their point of view and deliver documentations particular to specialists, whose solutions and innovation are to be later transferred from research to industry. This requires the creation of methodologies to assist developers to efficiently analyze and style agent-based systems (The National Science Foundation, 2005). Also, they are trying to standardize the work of various current approaches, similar to the expansion of UML in the OO area.

### 2.2.3 Agent Applications

Applications of agent technologies exist in many fields, including those currently and potentially artificial and commercial ones in AI distributed by multiple available agent systems (Nicholas, et al., 1998). According to (Luck, et al., 2003), instances of these applications are:

- Electronic trading, where purchaser agents and vendor agents sell and purchase merchandise on behalf of their users.

- Real time observing and more administration for telecommunication networks, the place operators need aid responsible, i.e., transmission.

- Modeling and improving internal transport systems, within cities, national or global, where agents are represent, for example, transport vehicles in general, which carry goods or customers.
- Dealing with data in an environment, such as the Internet, where multiple agents will be in charge, for example, of filtering and collecting information.

- Refinement of air traffic, where agents take charge of suitably interpreting information arising at various sensor stations.

- Scheduling meetings automatically, where agents perform on behalf of users to arrange meeting specifics such as place, time, and protocol.

- Improving industrial production operations, such as shop-listing or import administration, where these factors represent various working groups or institutions.

- Distribution of business operations among companies or within the internal system of the company, representing the agents of various divisions or parts involved in these operations in various stages and in different scales.

- Reconstructing of data and the control-flow manner within a wide array of naturalistic, technical, and hybrid institutions, where agents act as the element accountable for these manners.

- Achieving the social aspects of intelligence and matching complex social phenomena, such as the evolution of roles, norms and organizational structures as agents deal with the role of members within all natural communities.

### 2.3 Agent Development Methodologies

Developing different approaches remains a main field of research in software engineering and within any paradigm, one cannot depict a certain technology without the development methodologies associated with it. For this, the main purpose for these developments is to facilitate the process of improving software lifecycle and promote the status and quality of the product.

By means of the methods, models, notations and tools that are supported in these methodologies, developers can analyze, design and implement the target systems. In this effort, of course OO, AO can be considered as an important step in software engineering. The AOSE methodologies offer very novel notions as pro-activeness and autonomy (see section 2.2.1) to present more elastic and sturdy systems for complicated applications. Thus, a number of AOSE approaches, models and tools have been proposed.

The main aim of constructing these methodologies is to make for an agent's elastic, independent conduct and to establish relations among agents with the necessary complexity prevailing in the system's structure. These goals are accomplished with the help of the approaches, system goal paradigms, roles and conducts that the agents offer, and the connections protocol among the agents.

### 2.3.1 Categories of Current Agent Development Methodologies

The rising number of AO approaches and models are all related to one another and each has a different source.

Over the last years, the examination and analysis of these methodologies have led to MAS (or multi-agent systems) development (Iglesias,et al., 1999) (Tveit, 2001) (Weiss, 2002). These approaches are categorized based on the source on which they are based, as either agent-based, OO-based or knowledge engineering (KE)-based. Table 1 shows the AO methodologies' classification.

**-Object Oriented- Based Methodologies**

This classification for AO methodologies is not specifically designed for agent-based projects, but it is an extension of the OO methodologies, which are among the most popular and mature technologies. Also, there are similarities that exist between the agent and the object. The methodologies which belong to this type are founded on the Object Management Group's (OMG), and UML, which has a wide range of applications and represents the most important molding approach in the OO paradigm.

There is a significant number of agent methodologies that extend to UML so as to meet the demands of agents as well as to realize software methodologies according to OO, such as ODAC (Gervais, 2003), MaSE (DeLoach et al., 2005), MASSIVE (Lind ,1999), DESIRE (Brazier, et al., 1997), AAII (Kinny & George, 1996), AOMEM

Table 1. Agent oriented methodologies classification

| **Methodology** | **Year** | **Classify** |
|---|---|---|
| MASB | 1996 | Object methodologies |
| CoMoMAS | 1996 | Knowledge Engineering |
| AOAD | 1996 | Object methodologies |
| Cassiopeia | 1996 | Agent methodologies |
| AOMEM | 1996 | Object methodologies |
| AAII | 1996 | Object methodologies |
| commonKADS | 1998 | Knowledge Engineering |
| MaSE | 1999 | Object methodologies |
| DESIRE | 2000 | Object methodologies |
| ADEPT | 2000 | Knowledge Engineering |
| AOR | 2000 | Object methodologies |
| Gaia | 2000 | Agent methodologies |
| AUML | 2000 | Object methodologies |
| OPEN agents | 2000 | Object methodologies |
| Tropos | 2001 | Agent methodologies |
| Styx | 2001 | Agent methodologies |
| MASSIVE | 2001 | Object methodologie***s*** |
| SODA | 2001 | Agent methodologies |
| MESSAGE | 2001 | Object methodologie***s*** |
| PASSI | 2002 | Object methodologies |
| ROADMAP | 2002 | Agent methodologies |
| Prometheus | 2003 | Agent methodologies |
| ODAC | 2003 | Object methodologie***s*** |
| SONIA | 2004 | Agent methodologies |
| ASEME | 2011 | Agent methodologies |
| GORMAS | 2008 | Agent methodologies |
| OperA | 2009 | Object methodologies |
| DSML4MAS | 2008 | Object methodologies |
| ForMAAD | 2010 | Agent methodologies |
| O-MaSE | 2014 | Agent Methodologies |

(Kendall,et al., 1995), AOAD (Burmeister, 1996), MASB (Moulin & Cloutier , 1994), AUML (Odell, et al., 2000), OPEN agents (Debenham, et al, 2002), MESSAGE (Caire, & Leal, 2001), and PASSI (Cossentino, 2005).

Some methodologies, such as ODAC and AOAD, recognize the agents through analysis. Often times, all approaches, namely ODAC, MASB, DESIRE, AAII, AOMEM and AOAD recognize inter-agent sides, but only MASSIVE and AOAD support the social issues. Only DESIRE performs the ingredient-driven bottom-up agent determination operation. Only MASSIVE, addresses the surroundings feature (Alonso, et al., 2003). In this way, complex objects are processed, but agents have high levels of abstraction of objects and there are those that cannot be considered as objects. Despite similarities in some proprieties between the agent and object, a major difference prevails in their internal structure; objects encapsulate methods and attributes and agents encapsulate aims, plans, beliefs, and commitments.

They have various relationships and communication patterns. Where agents' behavior must be proactive, objects should not follow the same behavior. According to these differences, agent systems are often more complex than OO systems and, hence, OO techniques often fail to obtains the complexity of agent systems. (Alonso, et al., 2003)

**-Knowledge Engineering-Based Methodologies**

This type of methodology is an extension of the knowledge Engineering (KE) techniques and specializes in determining and gaining knowledge as utilized by agents. Knowledge Engineering has a lot of styles and forms, such as the libraries of ontology as well as problem libraries that can be useful for agent-oriented methodologies. Also, since the operations are handled by many methods (Glaser, 1997), this methodology is convenient for developing agent knowledge. CommonKADS (Iglesias,et al., 1998) methodology, MAS-commonKADS (Glaser, 1996), and CoMoMAS (Glaser, 1997) are knowledge engineering-based methodologies.

As mentioned in the other methods earlier, some problems can occur in these methodologies as they do not employ a general methods. For instance, MAS-

CommonKADS defines factors through analysis, after the top-down operations in the role (identifying actors). Both concentrate on aspects within the agent and the common client; yet, they do not take into account social cases or the surroundings.

**-Agent- Based Methodologies**

Because this group of methodologies is established on the substance notions of agent and the MAS, it is said to be better than the two previous ones since it also produces social-level abstractions suitable for the agent and the MAS. The most representative methodologies in this category are: Tropos (Giorgini et al., 2005) (Giunchiglia, et al., 2002), Styx (Bush, et al, 2001), Prometheus (Padgham & Winikoff, 2002), Gaia (Wooldridge, 2000) (Zambonelli et al., 2005), SODA (Omicini, 2001), Cassiopeia (Collinot, et al., 1996), ROADMAP (Juan, et al., 2002), O-MaSE (DeLoach & Garcia-Ojeda, 2014) and SONIA (Frutos, 2003).

All of these approaches define agents in terms of social roles, while Tropos and Prometheus do so in terms of the actors' roles. There are many propositions used by methodologies for the agent paradigm, such as those applied in Prometheus and Cassiopeia or the analysis stages in Tropos, Gaia, SODA, and Styx.

In this respect, three aspects sides should be borne in mind when it comes to designing MAS: intra-agent structure, inter-agent structure and social structure. Most agent approaches provide the intra-agent and inter-agent aspects (for example, Tropos, Gaia, Prometheus, Styx) but only SODA and cassiopeia consider the social structure. To address the issue of the environment is important, and SODA is the only method that does so.

There are more than twenty AO methodologies (see table 1) and, due to constraints such as time and resources to study all of them, this number is downsized so that we can evaluate them in detail. For this purpose, we select seven agent-based methodologies - Gaia, Tropos, PASSI, O-Mase, ADELFE, ASEME and Prometheus - to make sense of their advantages, drawbacks, and similarities and differences. In addition, we compare the meta-model for the selected methodologies. There are certain reasons leading to the selection of these seven methodologies as all of them

are designed specifically for the styling and analysis of agent-based systems and to cover a wider range of program expansions than other methodologies in AO software projects (see section 1.2).

### 2.4 Related work

Given the main purpose as to apply AOSE methodology for constructing powerful, industrial-strength Multi-Agent Systems (MAS) as well as the current trends, the best way to develop agent programs is by obtaining further analysis and understanding of the proposed method. There are many attempts to evaluate and compare agent-oriented methodologies undertaken by many researchers in this field. (Dam, & Winikoff, 2013) states that it is time to work towards a new generation of AOSE software architecture to give us an ultimate and unified methodology.

They propose an operation and style for AOSE methodology through comparisons of main agents among seven methodologies - Gaia, Tropos, MaSE, ADEM, INGENIAS, PASSI and Prometheus - in order to grasp the connections among them. The main objective are, in particular: (a) to assess the strengths and weaknesses of each methodology; and (b) to identify resemblance and variations between them in respect of needed or beneficial operations and models in leading the expansion of agent-based systems. After this comparition analysis, they suggested a next-generation methodology.

Fabiano Dalpiaz, Ambra Molesini, Mariachiara Puviani (2007) clarify that many methodologies have been suggested for the expansion of agent-based projects in the literature, and the main notions on which they depend differ from those adopted when implementing the system. This conceptual gap often creates contradictions among features and the actual enforcement of the process. They use a meta model-based method intended to fill this gap related to four relevant AOSE methodologies -GAIA, Tropos, SODA and PASSI. In this way, they provide a clear and basic definition of common characteristics and the main components used in MASs, leading to an integrated meta-model that combines common concepts of the participants' methodologies.

For every methodology to be combined in the meta-model, the authors elicited the requirements, determined a set of operations, thoroughly compared the concepts related to the various operation, and finally constructed the meta-model (Dalpiaz, et al., 2007). There are multiple frameworks to compare proposed methodologies to the agent. In the work by Sukhvir et al. (2012), there is a discussion regarding various agent-based systems classified in different application areas, and an evaluation of five various AO methodologies - GAIA, MaSE, MESSAGE, Prometheus and Tropos - through conducting a feature analysis and capturing the advantages and drawbacks of each methodology to integrate their strong points to develop new extensions. This work was achieved through a comparative study based on a features-analysis method.

Dam and Winikoff (2004) provide an analysis study aimed to capture the relations among five prominent AO approaches - Gaia, MaSE, MESSAGE, Prometheus and Tropos - in particular, through an analysis of their features by assessing the strengths and drawbacks within an attribute-based structure. Included in this framework are the criteria as concepts, modeling language, operation and pragmatics. The comparison can take a wider scope with a small experimental evaluation of methodologies which makes this comparison more important as it includes inputs from methodological rulers using a questionnaire.

A structural analysis is performed where main common factors and clear differences are identified for the five agent methodologies in terms of paradigms, processes and programs. Also, several initial proposals are presented so as to integrate these AO approaches by combining their advantages and minimizing their drawbacks (Dam & Winikoff, 2004).

In addition to these studies, Sabas et al. (2002) offer a multi-dimensional framework to treat five main important aspects of an AO approach to cover general software engineering issues and agent concepts in the process of evaluating a methodology. The results are described in the form of a two-dimensional group, including titles for standards and approaches. A different method using a goal-question-metric (GQM) is suggested by Gernuzzi & Rossi (2002), who applied feature-based evaluation mechanisms and presented metrics and quantitative assessments to define the

important factors for evaluation methodologies. A characteristics tree is formed based on the objectives of the GQM questions to determine the compared criteria. Every node of the tree represents specific characteristics of agent-based systems.

In another work, Shehory & Sturm (2001) propose traditional software engineering and agent-based system characteristics to compare a number of AOSE methodologies using a set of criteria as anomalies for evaluation purposes. In every domain, the identical standards are checked and text data offers notes on every point. The definitive assessment is on a gradually-increasing order, from good (+), to satisfactory (*), or not-supporting (NS). A comprehensive and multidimensional framework was introduced by Tran et al. (2003) to assess and compare MAS expansion methodologies developed from various and existing evaluation frameworks. In another paper, O'Malley & DeLoach (2001) suggested several standards for assessment approaches and allowing organizations to use AOSE methodologies or other existing software engineering methodologies.

### 2.5 Evaluation of Software Engineering Methodologies

In this section, we discuss and list the current existing techniques used for comparison and evaluation of software engineering methodologies.

#### 2.5.1 Evaluation of Object-Oriented Methodologies

As mentioned in section (2.4.1), the OO technology is the most popular and mature technology currently in use and an important experiment in the computer sciences. Certain similarities exist between agents and objects in terms of properties. In addition, there is an extension from the OO methodology into the agent-oriented methodology and OO methodologies represent an important part of software engineering paradigms and are counted as the foundation of AO processes. In order to benefit from research on OO methodologies relating to AOSE evaluation, it is necessary to review the major ways to evaluate and compare OO software engineering methodologies.

There are numerous assessment standards in this field beneficial to our objective; that is, to unify AO methodologies. In contrast to agent-oriented methodologies, there has not been much work on OO methodologies as they faced a hurdle similar to AOSE as it is currently taking place, and users have had difficulty finding a way to fully meet their needs. In this period of time, certain methods and concepts began to emerge. After introducing a series of OO methods and performing numerous comparative studies on them, in the mid-1990s, effort was made to unify the modeling languages which continued successfully until the late 1990s, resulting in the formation of UML, which can be used by all developers as a unified OO approach.

Accordingly, the same steps must be taken by the AO community in an attempt to explore engineering methods and, then, to use them in AO design. The use of an AO approach agrees to make a system out of an agent-oriented orientation.

Researchers in OO software engineering produced a large number of such methodologies for use in the expansion of OO applications along with systematic evaluation methods applied for evaluating, comparing and understanding these methodologies. A set of criteria has been defined to evaluate OO methodologies and, as an important step in the field, there has been a large number of analyses and comparative studies performed in this respect. Some of these works include Cribbs et al. (1992), Dumke & Foltin (1998), and Walker (1992), who used different types of evaluation methods such as feature analysis, meta-modeling, outcome analysis and quantitative evaluation, all of which will be discussed in more detail in the next section.

While some of these works have been done fully systematically, others focus on multiple stages of the operation. For example, in Hong's work, (1993), the entire operations of software expansion, modeling language and tool backing of the approaches have been evaluated. Also, Surya, et al. (1988) target the comparison of requirements specification techniques, while Frank (1993) deals with OO design and analysis.

### 2.5.2 An Overview of Agent Methodologies Evaluation

Deciding which approach is better to use in designing and implementing a project is not a plain task, as mentioned previously, and there is a large number of AOSE methodologies with different methods and tools developed during the past years. The complexity of today's life is increasing day after day and software products have become an important part of it as they continue to play an increasing role in different sectors. For this reason, software engineering must develop according to the current requirements as conventional software engineering available with earlier concepts and systems involve only routine automating processes, maintaining data in databases, or as reacting and interacting tools.

Today's emerging agent-oriented paradigm moves software to a proactive and clear autonomy and sociality, leading software engineering from manipulations by users toward delegation decision-making, and taking actions. Because there is no unified approach that can be utilized to develop and carry out agent applications as UML in the area of OO analysis and design, AOSE methodologies can be used to evaluate AO approaches and assist the developer in selecting among the available AOSE methodologies. Yet, despite the presence of diverse AO methodologies throughout the past years, there is no organized system of assessment to be applied to these methodologies.

### 2.5.3 Evaluation Techniques

In this part, we introduce the assessment techniques applied for evaluating other software engineering paradigms as OO as well as other agent-based methodologies. By means of the evaluation of these methodologies, developers can decide what the appropriate methodology is so as to design and implement a particular system, and produce high-standard software programs, Furthermore, this will obviously assist researchers in detecting the resemblance and variation points among these methodologies.

So far, there have been many major initiatives in this area over the past decades. Law (1988) provides a scientific approach used to compare methodologies, particularly

with respect to the way in which styling methodologies and requirements are assessed. Also, Jayaratna (1994) has offered a methodology to devise a framework for evaluating this area of expertise from a more comprehensive perspective. In a separate work, DESMET system, which began in 1990 and started to post specifics in 1996 was proposed by Kitchenham (1996), (Kruchten, 2000). It is based on elements involved in the expansion of a combined framework for evaluation processes, models and mechanisms in software engineering. DESMET has always been a basic tool for all areas of specific and quantitative assessment.

There have been countless classifications suggested by different researchers to categorize the ways for evaluating software engineering methodologies and approaches In (Dam, 2003), the main approaches to evaluating a certain methodology have been classified into four main groups: feature-based assessment, which is discussed in detail in the next section; quantitative assessment, which evaluates the methodology based on several standard outcomes from its utilize. These products can be applied in software implementations or alterations within the development processes; the third is the NIMSAD framework is based on methods that understand and evaluate methodologies in a more detailed and rather theoretical manner (Jayaratna, 1994).

Finally, there are other assessment ways as suggested in the literature and somehow independent of the three previous groups. For example, Law (1988) suggested that experts could employ a set of criteria to evaluate and make direct comparisons of methodologies rather than to study all comparative methodologies. Accordingly, the main evaluation and comparison techniques to be applied to software engineering methodologies are as follows:

**1.Feature-Based Comparison (Feature Analysis)**

This method of is generally considered as a qualitative method and is more popular than the other approaches as the comparison is based mainly on a number of traits and characteristics to be examined alongside the establishment of an assessment framework that can be represented in a set of characteristics, attributes, features or merits (Law, 1988). This type of evaluation technique is often used to highlight the

ease and usability of application and is employed in a simple manner, having been already and successfully implemented in many cases, and the application can be implemented by a small number of evaluators. However, the feature analysis applied is subjective and depends on the experience, background, and knowledge of the evaluator.
Researchers have produced numerous papers on this evaluation techniques and the different derivations of its features.

There are some methods to assist evaluators to derive the attributes evaluation (Kitchenham, 1996) (Law, 1988). For example, one can think of the methodology of expanding software engineering as a group of paradigms, operations, mechanisms and connections. Then, the methodology assessment is applied to backing for the characteristics of each of these elements. The second method includes utilizing an expert's perspective to identify the main merits of the methodology required for evaluation**.** These advantages can be deduced either by experience or by the notions and rules of software engineering.

We can use both methods in the same study to obtain satisfactory results as the techniques are often applied together to set up criteria, which is an important issue in the evaluation framework and assessing the methodologies and must, hence, be generated more precisely. There are some classifications applied to feature-based methods, such as that proposed by Kitchenham (1996), which categorized them into four main types of approaches: screening-mode approach, case-study approach, formal experiment approach, and survey approach:

**-Screening Mode Approach**

This method depends on only one person to perform, one who is accountable for both producing the evaluation standards and carrying out the methodologies' assessment. Although this method does not need a significant amount of time or cost, the success of the methodologies assessment applied can be affected by the individual's background, experience and degree of precision when examining the methodology documentations. These abilities differ from one person to another. In addition, the estimation of the evaluator subjective it may not be consider of the users and authors

of the approach. Also, the outcomes of the evaluation may not be precise and the expert may not even apply a correct set of criteria to any aspect of the methodology (Dam, 2003).

**-Case-Study Approach**

When we use a case study approach for methodology evaluations, the results can be positive if the user applied it so (Kitchenham, 1996) (Law, 1988). Additionally this approach is beneficial since the evaluation outcomes can be made based on direct comparisons among the competing methodologies. The goal of this approach is to examine the ability of different methodologies to solve a real problem by applying the same real example in several methodologies. In this technique, the evaluator selects the methodologies, produces the assessment standards and chooses the testing system. Then, software developers evaluate the methodology attributes based on their background with this methodology to design and analyze real projects. Similar to the previous method, the evaluation relies on the background of the evaluators in the field as well as their ability and experience in evaluating different features so as to gain a good understanding.

**-Formal Experiment Approach**

Evaluators in the formal experience approach choose the method and create the collection of merits, plan and run experiments, and analyze the outcomes, all of which may call for a large number of evaluators participating in the methodology estimation. They also need to select a proper experimental design, distinguish experimental subjects, and categorize them into various groups of users. Furthermore, experimental materials are trained to use the methodology. Not only does this method produce the most reliable results, it can also reduce the influence of the one-evaluator-only factor; yet, the approach is considered more expensive, when compared with other techniques, and requires more time to complete the evaluation process (Dam,2003).

**-Survey Approach**

The survey approaches is similar to the three groups mentioned earlier, and evaluators need to set up a group of evaluation standards. Then, they style the survey and work on it. In contrast to other evaluation techniques, this approach is not included in real projects' evaluation methodologies and is based on the experience of users to apply some of the methodologies that have been evaluated as several procedures must be followed when selecting the survey approach to evaluate software engineering methodologies, not to mention selecting the type of survey.

Sometimes, questionnaires are used for this purpose on the Internet or in the form of a personal interviews depending on the decision as to who should be questioned as participant in the study. Finally, the surveyors collect and analyze the user's understandings of the selected methodologies according to the design.

The survey approach does not require a large amount of time and effort. Also, it presents the view of a large number of users and their approaches. However, the assessment results completely depend on the experience, background, and the knowledge, of the persons who shares ideas in the questionnaire. In fact, these abilities are different from a person to another and an evaluators cannot determine and control the person's abilities (Dam, 2003). We believe that there are other things that have an effect on the survey results such as the number of persons who participate in the survey; the higher the number, the better the results, especially if these participants have experience and a high degree of understanding as to the selected methodologies.

**2. Meta-Modeling**

The meta-modeling approach is a model of percepts that can be applied to designing and to portray systems. Models that describe system model entities are examples of meta-model components, and the assessments using this technique are similar and feature-based comparisons. The meta-modeling approach for assessing analysis styling methodologies uses a formal approach to compare methodologies and, in general, should be formed based on a range of various traits offered by comparative methodologies (Bernon et al., 2005).

According to the meta-models of the competing approaches, this work depicts the comparison of methodologies based on three main traits: steps of analysis, design, notions and mechanisms offered. This method is effective as it is more precise compared to the feature-based technique. As opposed to OO, the status regarding the AO approach varies greatly because the lack of a specific model to MAS results in each approach to have its own notions and construction, and there is no commonly accepted definition to explain the notions of the agent and the associated meta-model of the MAS.

A framework representation of the notions (agent, role, behavior, ontology, etc.) will compose of the actual system and their authoring relations (Walker, 1992). The assembly building does not look at other points of software engineering principles such as reuse, maintenance, and modification. Plus, meta-modeling methods are intended only for situations when we need to compare a specific number of approaches.

### 3. Metrics Approach

The metrics Approach analyzes the complexity of the different methodologies as it is an analysis of constructs and performs a metrics-based assessment of program expansion approaches for objects previously proposed (Siau & Cao, 2001). The structure is a building block of the methodology; for example, in a UML class, object, link, message, state, and activity are built. This approach is subjective in deciding on which buildings will be calculated and how much. Furthermore, there is not much empirical work on this technique as the metrics approach is required to add experimental work to validate the standards (Siau & Rossi, 1998).

### 4. Ontological Evaluation

The existential assessment approach (Wand & Weber, 1990) proposes the use of ontological concepts to assess methodologies. Using the banjo's ontological model (Hong, et al., 1993), the structures are arranged in the modeling method of the ontological structures, and assessment is complete once planning is carried out one by

one between the method and the ontology. If there is no such customization, the method may include many problems such as building overload, redundancy, or deficit.

The priority of this technique is that the external and objective body determines the structures required, and that assessment is required to determine the structures within the evaluated model. In any case, doubts have been raised as to whether the choice of the Bunge model to describe the etiology of the system is justified (Hong, et al., 1993). Moreover, there are no guidelines for identifying structures within the methodology.

## 5. Empirical Evaluation Techniques

This approach includes surveys, laboratory and field trials and case studies; this techniques usually require the participation of organizations and practitioners, is a very difficult subject and needs a lot of resources (Hong, et al., 1993). Apart from the approaches stated earlier, there are other methods that can be used for the same purpose such as:

### -Outcome Approach

The use of this method in the comparison of methodologies is a form of quantitative evaluation, which assesse the approach based on some standard outcomes resulting from its employment. Attribute analysis is a type of analysis (Kitchenham, 1996) including the construction of an assessment framework that can appear as a group of features. The results of quantitative assessment methods are more reliable than qualitative methods. A number of OO methodologies are evaluated using results in terms of many measurable effects. For instance, product quality or operation complexity can be examined as a result of the application of certain methodologies. In (Jayaratna, 1994), examples of this evaluation approach exist, which provide a study of object-oriented methodologies as an official experiment in the evaluation of analytical techniques to determine what information is exactly required.

**-Comparison Against a Framework**

This evaluation method has attracted more important work on OO methodologies than the other evaluation and comparison types. Assessments using this method involve constructing a structure that contains a hierarchy of characteristics. These represent assessment merits, on which the evaluation is founded.. The assessment structure is also used in different works for example in (Walker, 1992), where a perfect expansion methodology is built geared toward objects that can be compared to others. This is while most other work build their own framework (Law, 1988).

# CHAPTER 3

# Evaluation Techniques (part 1)

In the preceding chapter, we characterized several current ways to compare and evaluate different software engineering approaches. Chapter 3 will deal with the definitions of technical aspects as well as different methods to assess the agent methodologies intended for the present research. This chapter is organized as follows: Section 3.1 is an overview to cover the processes, models, and techniques in the seven AOSE methodologies. In section 3.2, a structural analysis is performed to examine and identify the common core of all the models and processes as well as the various components of each methodology.

## 3.1 Methodologies Overview

This short overview covers the processes, models, and techniques in the methodologies intended for our research.

### 3.1.1 Gaia Methodology

Gaia (Zambonelli et al., 2005) was the first agent methodology which was designed specifically for agent-based systems by dividing the activity of developing software

into two main steps, analysis and design; the one groups and arranges the specifications as the foundation for the second stage, which is a computational organization. Gaia determines both the objectives and the expected conduct of the organizations which form the system in general. For this purpose, it breaks down the global organization into highly interrelated subsidiary institutions. The preliminary roles model may not be fully defined but can be determined without imposing a particular organizational architecture. Also, the concept of roles in this category is abstract from any form of mapping into agents (Zambonelli et al., 2005).

In this respect, the first step (that is, preliminary interaction) describes the main interactions expected to achieve the elementary roles. The environmental model aims to represent the abstract and computational environment where MAS is to be developed. The organizational rule model provides a set of principles that the organization has to take into account and put into effect in its global conduct (Wooldridge et al., 2000).

The design stage, using the output of the analysis stage, is separated into two phases, architectural design and detailed design. The topology and control systems are defined through the architectural design step, which also employs catalogs of different institutions' models, take into account the organizational rules, completes the preliminary models, and finally defines the agent model defining the agent classes to be shaped form the system and the agent to be created from these classes (Zambonelli et al., 2005). Figure 2 shows the processes in Gaia stages.

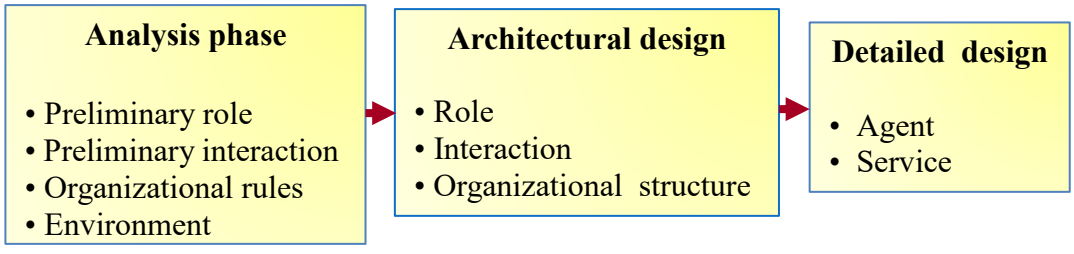


Figure 2. Gaia methodology process

### 3.1.2 Tropos Methodology

This methodology was first proposed by Giorgini et al. (2005) at the University of Toronto, Canada, and was later updated by several other universities in Europe. The methodology covers the very early stages of requirements analysis. To do this, the Yu's i* framework (Castro et al., 2002) is adopted as the main theory for requirement analysis. The i* provides crucial characteristics like actors, goals, and relationships, that aim to model social structures, and depicts detailed relationship dependencies among them.

The methodology comprises five stages: early requirements, late requirements, architectural design, detailed design and implementation. In the first phase, the main system requirements are extracted. Later, most of the general features determined in the initial requirements analysis are used in the late requirements analysis stage. Throughout both these phases, the same concepts and technical methods are applied, especially during the first phase as it is intended to extract the main system requirements(Bresciani et al., 2004).

In this phase, the primary function of the system actor is to deliver system-based services to actors based on services from the recent analysis stage (Bresciani el al., 2002). In this respect, the architectural design and the detailed design stages concentrate mainly on the system specifications based on the requirements as obtained from the previous stages. In this way, the architecture of the system is materialized in terms of interconnected sub-systems through data and control flow. These sub-systems are depicted in the diagram in the same way as actors and data links are depicted as dependencies. This stage also maps the actors in the system to a group of software agents, each with its own certain abilities (Mylopoulos et al., 2000).

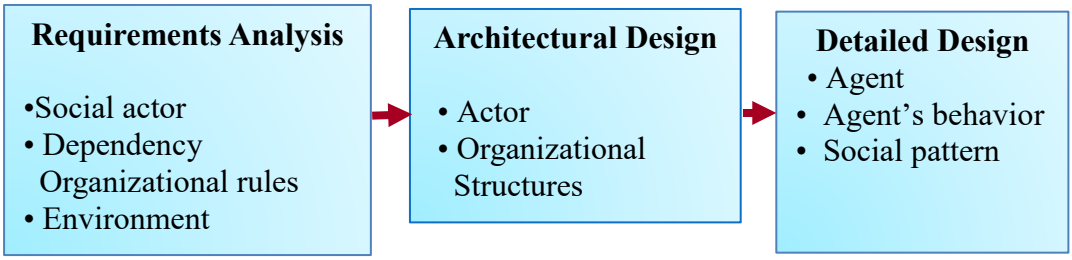


Figure 3. Tropos methodology process

In the elaborate design step, the purpose is to define the agent abilities and reactions, by which point the application platform is usually selection already and, so, can be utilized to an exhaustive design that will move directly to the code stage. The execution processes follow a step-by-step pattern of the exhaustive design characterizations based on the path among the execution platform constructed and the detailed design concepts (Giunchiglia et al., 2002) (see Figure 3).

### 3.1.3 PASSI Methodology

Process for Agent Societies Specification and Implementation, or PASSI for short, was proposed by Cossentino (2005) to construct agents software support all software processing stages, from the requirements phase to code phase. It consists of five steps involving all the stages adopted in the UML modeling language. Since UML is highly accepted in both academia and the industry, PASSI uses UML as its language of modeling with expanded schemes to include certain notions left out from UML.

These are later updated so as to represent the best of what modeling should be in a particular artifact within the agent design community (Cossentino et al., 2002). In PASSI, an agent is an important software entity, both at the abstract level and the design level. In light of this fact, an agent represents a class, thus making it an independent unit when designing software by reaching a goal through independent resolutions, procedures and social relations.

To do so, the agent may perform different roles during its interactions with other agents. A role is defined as a set of functions performed by the agent in pursuit of a sub-goal. In turn, the task is known as a meaningful element of individual or interactive conduct. In what follows, the five major models of the PASSI methodology address the different design elements alongside the twelve steps in the process of constricting a model (Cossentino et al, 2000):

a) System Requirement Model: It describes the needs of the system in terms of agency and objectives, and includes four steps: domain characterization, agent identification, role identification, and task determination.

b) Agent Society Model: It is used to represent the social interactions and dependencies among the agents involved in the problem-solving initiative, and includes four stages: role specification, ontology, role and protocol characterization.
c) Agent Implementation Model: The solution architecture in terms of the methods and classes is provided in this model, which comprises two steps: defining the agent structure and determining the agent behavior.

d) Code Model: The solution at the code level is modeled through this stage, necessitating code generation and manual completion of the source code.

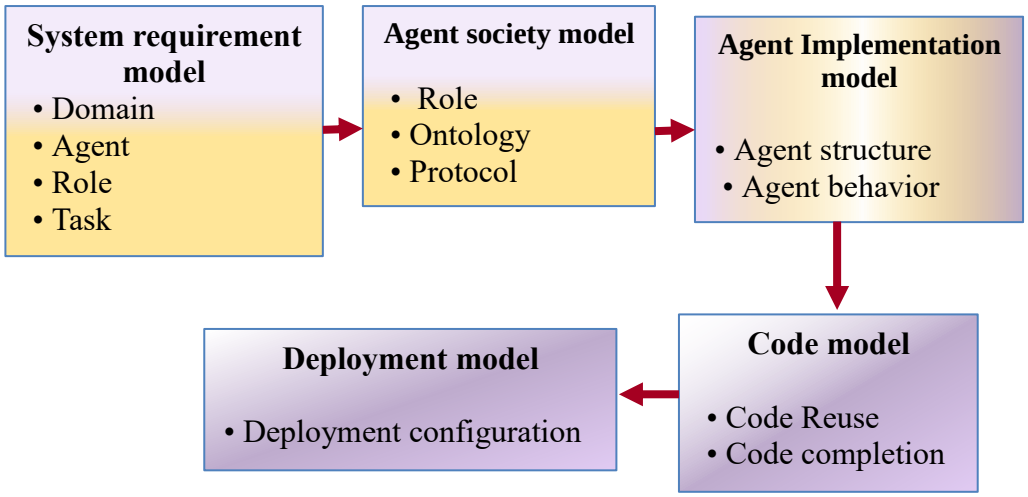


Figure 4. PASSI methodology process

e) Deployment Model: This model is used to represent the distribution of system units throughout the hardware components.as well as the distribution of their immigration. There is only one step: deployment configuration. Figure 4 shows the processes in PASSI stages.

### 3.1.4 O-MaSE Methodology

The O-MaSE process structure (Organization-based Multi-agent System Engineering) (DeLoach & Garcia-Ojeda, 2014) (DeLoach & Garcia-Ojeda, 2010) assists processing engineers in order to determine custom MAS analysis activates. It offers the models, principles, approaches fragments, and guidelines necessary to collect O-MaSE-compatible operations with the main goal to allow the construction of agent-software activities.

O-MaSE is composed of three major models: (1) a meta-model, (2) a group of method fragments, and (3) a group of guidelines. The O-MaSE meta-model is regarded as the main factor needed to develop and execute MAS. As for the method fragments, these are processes followed to provide a set of work products, possibly including diagrams, documents, or codes. The relationships among the method fragments are specified by the guidelines. Figure 5 shows the processes in O-MaSE stages.

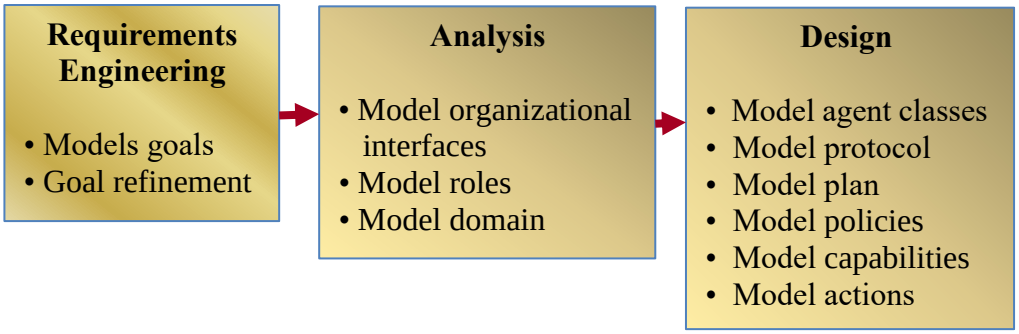


Figure 5. O-MaSE methodology process

### 3.1.5 ADELFE Methodology

The ADELFE methodology (Bernon et al, 2002) has been designed to focus on several aspects left disregarded by current approaches. It unifies AMAS theory (Adaptive Multi-Agent Systems), and offers a specific process derived from an interpretation of RUP (Rational Unified Process). Several addendums have been proposed as regards AMAS Theory so as to include features characterizing the environment in the system in order to identify certain shortcomings in the collaboration process. Each user and task deliverer has an individual goal to deal with the required request without the need to understand the entire system functions. In this methodology, the engineer, AMAS specialist or not, is directed to use adaptive MASs in order to define the agents with the help of the environment models.

ADELFE offers the analyst tool to evaluate the sufficiency of AMAS technology at two levels: global (system) and local (components). At the local,level, three parameters are taken into account, whereas the global one involves eight. To demonstrate agent interaction protocols, the AUML (Agent Unified Modeling Language) (Odell et al., 2000) principle is used together with UML and RUP (Jacobson et al, 1999). Two other tools are incorporated into the framework: the first one, open tool, is a graphics modeling tool that provides backing for UML and ADELFE notation, which present new stereotypes and AUML interaction protocols. As to the second one, interactive tool, it depicts the modeling process and assists in directing the development of its application. ADELFE contains these three phases: Preliminary and late requirements, analysis and design (Bernon et al, 2003):

a) Preliminary and Late requirements stage offers the environment model, presents a general view of the system, converts that general view to a use-case diagram, and finally arranges and manages the requirements as well as their primacy.

b) The analysis stage evolves a comprehensive system by structuring in terms of entities.

c) The design phase has four steps: detailed architecture and agent modeling, agent architecture, non-cooperative situations modeling and forming class diagrams.

ADELFE is supported by an internal CASE-tool called OpenTool, as described earlier (see figure 6).

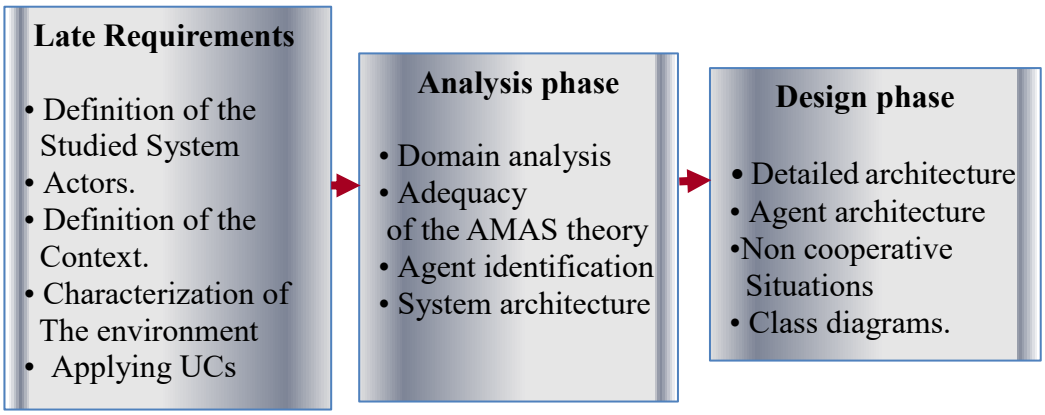


Figure 6. ADELFE methodology process

### 3.1.6 ASEME Methodology

ASEME (Agent Systems Methodology), proposed by (Spanoudakis & Moraitis) in 2011,contain the requirements, design, and implementation stages. It backs modular agent design processes and offers the notions of an agent within control to determine the agent's conduct by formatting various modules which execute his ability, and IAC which defines the protocols that control coordinate the activities of the agent community. The requirements analysis stage identifies the actors, their goals and interactions and use cases diagrams so as to assign actors to roles and goals in accordance to the capabilities needed to fulfill those tasks.

The partake doers are recognized by the objectives assigned to each. Furthermore, data is collected as to the specific requirements that constitute the anticipated tasks of the project. The analysis stage is based on the notions of ability and function, comprising two steps: the use case and the roles diagrams. Initially, the actors in the previous steps are converted to roles, which can be more abstract than the roles of specific actors depending on the scope of implementation (Spanoudakis & Moraitis, 2007).

The design stage encompasses the functional and behavioral sides of the MAS, and the related models are the agent IP and IAC that carry out a certain IP by assuming the crucial roles and relations between them (Spanoudakis & Moraitis, 2011). The execution phase is the language programming for different levels throughout the development process. Later, during the verification phase, the system functions are checked against the requirements, and phase is completed in relation to three abstraction levels (societal , agent , capability); nevertheless, the preferable method is successive, that is the software elements are checked for the effective running of algorithms, the agents are tested for the effective execution of abilities, and the MAS is checked for its general valid procedures in the end.

At the optimization stage, the system is optimized with respect to the algorithms in the capability level at the time of implementation. The number of simultaneously carried out abilities can be improved at the agent level. Finally, at the community level, the number of agents to be developed and those to be destroyed can be improved while the system is in actual operation (Spanoudakis & Moraitis, 2007). Figure 7 shows the processes in ASEME stages.

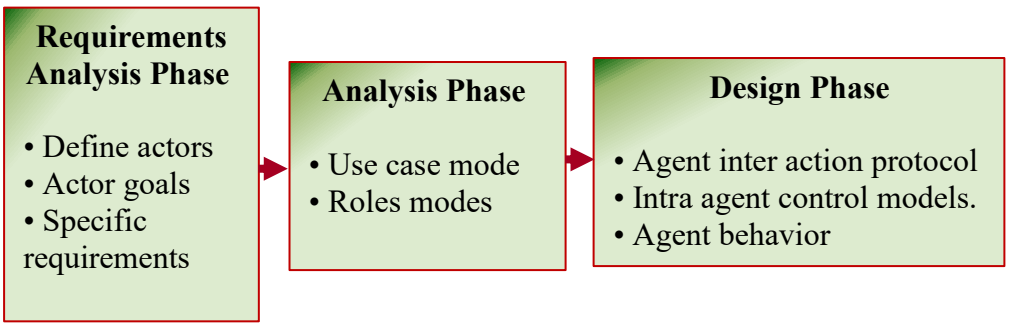


Figure 7. ASEME methodology process

### 3.1.7 Prometheus Methodology

The Prometheus methodology (Padgham & Winikoff, 2005) is composed of three stages: first, at the specification stage, the system is defined by the objectives and use case scenarios; the system mediator to its surroundings is explained in terms of the

events, concepts information; and tasks. Next comes structural design, through which the type of agents are determined. Here, the general structure of the system is described in the system overview model while UCS is incorporated into the communication protocols. Finally, the third stage involves detailed design used to develop and define the details of the internal agents such as capabilities, beliefs, plans, tasks, and events.

Then, process models are applied as a starting point among the protocols and interaction plans. In general, each of these steps contains models that concentrate on the dynamics of the system in the form of schemas and models regarding system architecture or its ingredient as well as textual depicter forms that deliver the details for individual elements. Figure 8 shows the processes in Prometheus stages.

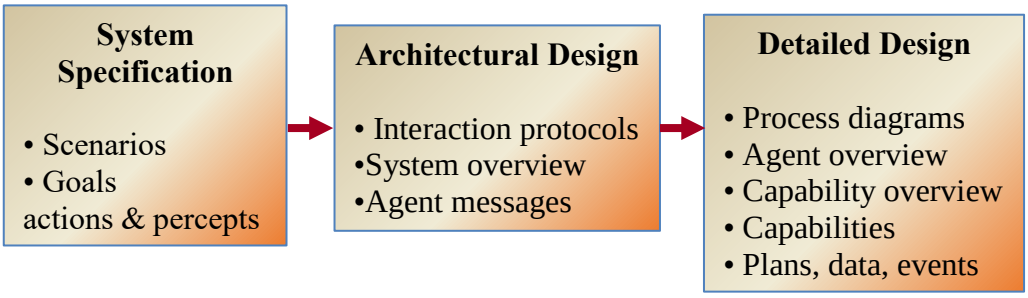


Figure 8. Prometheus methodology process

### 3.2 Structural Analysis of Methodologies

In this section, methodology processes and models are addressed so as to identify the common cores and components of the seven methodologies. To do so, structural analysis is performed on each process and model, allowing us to determine the similarities between these methodologies as well as the benefits of each one.

#### 3.2.1 Structural Analysis - The Commonalities

**-Initial requirements**

Of the seven selected AO methodologies, Tropos and ADELFE makes use of this technique, which is an important stage in this methodology; Tropos focuses on the intentions of stakeholders. In addition, this model is concerned with key agent notions such as objectives, intents, plans, etc., as used in agent-oriented programming. Through a variety of forms of objective-oriented analysis, these preliminary aims ultimately lead to the functional and non-functional demands of the target system. Therefore, the difference among various development stages decreases. Also, this can lead to the formation of unified and consistent techniques and tools for designing software. The goal-oriented requirements analysis approach was used in this model i* (Yu, 1995) to study an organizational base involving the users, their intentions, and relationship.

In addition, users are presented as actors who rely on each other for objectives to be fulfilled, functions to be completed, and resources to be provided. In the i* framework, the strategic rationale model is used to describe and support the logic that each actor passes through depending on its dependencies with other elements. (Bresciani el al., 2002). ADELFE initial demands stage objective to convert this seeing to a UC diagram, and arrangement the requirements (functional or not) at this step, the developer has to determine the task of the thoughtful system and to model its environment.

**-Applying use-case requirement analysis**

This technique is applied in order to characterize each system's practical requirements. The model has been successful in object-oriented methods to gather system requirements apart from assisting developers to identify the main interactions among system entities. Of the selected methodologies ADELFE, PASSI, ASEM and Prometheus have applied use-case models. In ADELFE, while complementing the workflow requirements, this operation consists of three steps: designing use cases, clarifying the linked sequence diagrams, and determining collaboration failures.

In these use-cases, only energetic components are implicit and appear as the products of an effective requirements group (Bernon et al., 2002). Exploring situation collaboration failure in the system and within its conditions is carried out so as to help

developers in identifying problems and non-cooperative items and incidents. This definition serves as a filter across the evolution activities to later define the agents in the process (Bernon et al., 2002).

Instead of using the objectives in the requirements engineering stage, PASSI authors prefer the approach of Jacobson (1992) and use the UC diagrams to describe requirements. In this respect, the domain characterization stage practically depicts system components of a hierarchical set of UC diagrams. The sequence models explain graphical scenarios for the detailed use case, while agents are presented as a pack of use cases in the functional analysis of the previous stage. Typically, the relationships among the use cases of various agents are regarded as communications, whereas the joints among the use cases of the same agent follow the same usual UML syntax and stereotypes.

In ASEME, the purpose in using the UC model is to show that artificial agents are designed as components reactive with other artificial agents or human agents. No new elements other than those offered by UML are not needed, but shifts in semantics are displayed. First, the actor interacts with the system and supposes a role. Then, agents are developed as roles either inside the system boundary, i.e. for elements to be developed, or outside the system boundary, i.e. for agents in the environment (Spanoudakis & Moraitis, 2001).

Similar to conventional UML UC models, human actors are demonstrated as roles outgoing the system box. Later, the various UCs should be linked to at least one artificial agent role. The UC diagram in ASEME also adds three new ideas regarding the actor diagram and offers actors designed inside the system boundary. The diagram can add abstraction roles to ensure that agent IPs and goals are viewed from a developer standpoint by adding sub-goals related to implementation in the shape of UCs (Spanoudakis & Moraitis, 2007). The third step of system specification stage in the Prometheus approach is to set up UCs as comprehensive descriptions of a series of events to fulfill specific goals or to react to a specific event (Padgham & Winikoff, 2005).

**-The concept of environment**

What the concept of environment is to MAS is essential as agents operate in an environment and, in order to support complex open systems, agent methodologies need evident models characteristic of the domain as well as the environment of their implementation. Often times, complicated open systems have very dynamic and diversified environments, with which defining and familiarizing is necessary if constant change is an issue. As a result, an agent system needs to have models to accurately represent the environment in which it operates. Despite the importance of developing an environment model, ADELFE, Prometheus and O-MaSE manage to specify such models.

In ADELFE, before identifying use cases and during the final requirements, the environment should be thoroughly planned by the developer. Afterward, one procedure is added to the RUP to describe the system environment. Specification starts by determining which elements interact with the system as well as the restrictions in these connections. In UML, an entity represents an actor and can be characterized as effective or ineffective in ADELFE. In turn, an effective or active element may act independently and be capable of working dynamically with the system.

Whereas, an ineffective element can be taken as an exporter of the system and, as such, utilized or updated by an effective one, but it cannot be changed independently. This distinction between entities is necessary because the agent that makes up the system and is not yet recognized at this step will be set between the effective ones. By offering a mediator model that includes a series of concepts and events conceivable in the surroundings of the agents to showcase the environment, Prometheus presents the environment from the system perspective.

The motivation behind this showcase is that the system, in general, and agents, in particular, see the surroundings across a number of sensors. For this reason, it is essential to obtain obvious input/output specifications for the required system features. The model offered by Prometheus only covers this interface, and the main

components of the surrounding in which agents will work are shown by the domain diagram in O-MaSE. These components are described in the form of objects from the surrounding, which contains agents, and interactions among those objects. It can also be used to show the general characteristics of the environment to see how the objects connect (DeLoach & Valenzuela, 2006).

To show how an organization may conduct in a certain circumstances a developer uses the elements determined in the objectives, roles, and agents diagrams along with those elements determined in the domain model in order to define organizational policies. Nevertheless, most of the other seven approaches fail to properly consider this important point. In fact, Gaia does not provide a comprehensive model as to the environment design for developers, and such data is encoded only in the authorizations and protocols of specific roles. This handicap makes Gaia unsuitable for designing implementations with active and diverse environments. In this respect, Tropos offers resources as an entity, but no more.

**-Capturing goals**

In the domain of artificial intelligence, there are two forms of agents, weak and robust. The first group is distinguished from the next in terms of independence, reactivity, proactivity, and socialite capabilities (Philippe, 2000). Strong agency is defined by all such properties as those of weak agency but, sometimes, it also attributes to other features as mobility (the ability of an agent to move around an electronic network), veracity (not knowingly communicating false information), benevolence (not having inconsistent objectives, and all constantly attempting to do what is required) (Hoa & Winikoff, 2004). In general, agents must have proactive abilities to accomplish their goals pursued over time.

For this reason, all agent methodologies subject to analysis here pay more intention to agent goals. The exception in this case ADELFE and PASSI. Gaia, models the goals in the form of roles and responsibilities. The other methodologies identify goals in the requirements analysis stage to be applied as a basis for identifying agents. In Tropos, setting goals is an important process – also known as goal modeling – and is done by

setting both system and individual goals, thus resulting in more goal-oriented outcomes (Giacomo et al., 2007).

Goal diagrams are designed through the early requirements stage by utilizing initially specified actors, objectives and, thereafter, described during goal models. These models operate identically while determining actor relationships in the late requirements stage and structural design (Giunchiglia et al., 2002). There are two different types of goals (Khanh , 2003): hard goals, which are associated with functional requirements, and soft goals, which lead to non-functional requirements.

The objectives and plan diagrams enable designers to understand the goals and plans by completing certain activities, namely means-end analysis, which converts a goal into sub objectives to determine schemas, and resources, as well as soft goals, which offer a means for accomplishing the goal; next comes contribution analysis which enables developers to assess the objectives in terms of either positive or negative contribution.

While there are a variations in terms goal setting in agent societies, O-MaSE specifies a goal concept as a main objective of the institution and set whenever the situation calls for it. According to DeLoach & Garcia-Ojeda (2014), the objectives of the organization are set in the form of a behavioral goal hierarchy and involve the goal characteristics and the relationship of precedence, see section 3.3.6 for a more detailed description of the use of the goal diagram in O-MaSE.

As to ASEME, the goal model is similar to the Tropos actor model but without the same diagrams. ASEME defines more concepts used by Agent Modeling Language (AMOLA), which offers more appropriate models for system analysis, and uses the goal and actor concepts in the related diagram (Spanoudakis & Moraitis, 2007). Determining objectives is also a vital ingredient in Prometheus and presents such models as an essential part of the institution. The goal-capturing stage has two steps: identifying goals and structuring goals (Padgham & Winikoff, 2005). Firstly, the objectives are specified by analyzing and understanding the group of requirements.

Then, they are arranged in a shape that can be transmitted transferred to the design phase (Padgham & Winikoff, 2005). In general, each objective has a role (one to one). Also, in its analysis stage, GAIA defines the goals of the organizations alongside their predictable behavior. According to some studies, ADELFE and PASSI methodologies do not address the issue of goals well enough (Giacomo et al., 2007).

**-Social system structure**

Computer system applications are becoming more and more complex by the day. In this respect, the benefits associated with using MASs in all types of software engineering programming requires strong support for engineering complex open systems which emphasize the social aspects of an agent system. Among the most important features to add agent-oriented paradigms in the software engineering is the ability to develop and implement complex systems (Mehdi, 2015). Consequently, it is quite necessary for AOSE approaches to offer a clear way to comprehend the general framework of the system. Except Gaia, most of the selected processes handle this matter fairly well. Between the seven approaches, only Gaia that does not have a thorough vision when it comes to social system (Hoa & Winikoff, 2004).

O-MaSE provides a social overview by supporting the concepts of roles and objectives, and represents the conversations among the main elements in the system. Tropos has extensive schemas which offer the relations among actors, objectives, resources, and functions in the system. If the significance of a comprehensive look at the structure of an organizational system lies in its ability to show a fixed structure, by the same token it is necessary to capture high-level system dynamics (Hoa & Winikoff, 2004).

The conversations and connections taking place between agents involves characterization of the mechanism that agents employ to organize their other complicated social actions or interactions such as rivalry, discussion and teamwork. Accordingly, O-MaSE and Tropos show conversations at two various standards of detail by including a group of high-level connections and more granularity performance in terms of IP (Giacomo et al., 2007).

From our observations, ADELFE, PASSI, Prometheus, Tropos and O-MaSE share certain features as they model high-level interactions using sequence/interaction diagrams extracted from UML sequence models. Each methodology has different interaction diagrams; for example, Tropos and ADELFE depict conversations among agents. This is while the sequence models in O-MaSE show reactions among roles and actors. PASSI uses serial graphics to explore the tasks of each agent through role-specific screenplays (Giacomo et al., 2007).

The role model in ASEME is basically derived from the Gaia methodology since, in both cases, the conversations are explored and modeled at the role-level instead of agent-level, and serial interactional schemas are not used for the objective (Spanoudakis, 2011). Getting into more IP detail, ASEME, ADELFE, and Tropos suggest application of the AUML IP model, with certain tasks added to the AIP to suit AMAS's demands in ADELFE.

**-Agent acquaintance model**

In Prometheus and Gaia, the dependency and interaction among agents are illustrated in terms of agent acquaintance schemes in the form of guided diagrams including rectangles (referring to agents) and lines with arrows (referring to links, interactions and relations) among different system elements. While Gaia realizes the connection links found among agents without determining the current properties of those links, Prometheus indicates the types of agent as well as the connections among them. However, in the design stage of Prometheus, these connections appear in more detail. For this, agent knowledge diagrams in these two approaches are intended to help developers in determining potential bottlenecks formed among the system elements.

### 3.2.2 Structural Analysis - The Differences

**-Model-driven engineering (MDE) approach**

In ASEME, the Model-Driven Engineering (MDE) method appears with special features concerning the AOSE community, and can be applied in all stages of conventional software processes (from requirements to execution). It allows the transition from one stage to another through typical transformations. The three forms of conversion used for automation between the ASEME models are: model to model (M2M), text to model (T2M) and model to text (M2T). Process designers simply enrich these models in each stage with specific information, thereby leading to execution progressively.

Furthermore, the steps in the design stage within this approach is a state chart designing models known to developers that can be performed by either different programming languages or agent-oriented platforms (Harel & Kugler, 2004). Yet, unlike ASEME, the Tropos methodology nominates methods and tools to automate, but only in several stages of software processes.

**-Documentation of non-functional requirements**

One of the most important differences between ASEME and the other participant methodologies is in its backing registration of non-functional requirements, in the requirements analysis step, where they are utilized to make management decisions and choose which technologies will be used for design and development (Spanoudakis & Moraitis, 2001).

**-Deployment model**

An important feature which distinguishes PASSI from other participant methodologies lies in its strong focus on the deployment model, regarded as one of the main models in UML (Booch et al., 1998). There are important benefits in this model. For example, in the process of building the deployment model, developers can have a better grasp of the intricacies to operate the system. Also, developing high-scale deployment offers a basis for evaluating the feasibility of executing the system, and the use of the spread concept. It also provides an assessment of different other measures, such as costs. Using this activity, PASSI describes the system based on agent classes and their position on the available processing units in the diagrams.

While building the deployment diagrams, designers have the opportunity to accurately set the system in a design step by thinking about agent-linked data such as the number of agents, their types and locations. PASSI offers elasticity in the system spread design by enabling the developer to create various system arrangements and to update them regularly.

**-Data Coupling Diagram**

An important aspect which distinguishes Prometheus from other participant methodologies is about data coupling as it offers evident processes to cope with agent characterizations. One of them is the use of data coupling models. The second is the model of agent acquaintance. These models are composed of system functions and external resources in terms of specified data. In this way, developers are able to assemble functionalities into agents by simply visually assessing the data coupling techniques and providing guidelines and processes. This depends on both minimizing coupling and increasing cohesion. It appears that placing agents that can read or write the same type of information with each other reduces the association of agents.

# CHAPTER 4

# Evaluation Techniques (part 2)

As a sequel to the previous chapter, we will continue to describe the methods we used to analyze and pass competing methodologies. Section 4.1, display a case study by applying all the available methodologies on the same model. Section 4.2 demonstrate common concepts by conducting an analysis of these Methodologies, comparing the meta-models related with each chosen approach.

## 4.1 Case Study

As a second step, we adopted a case-study approach in the form of feature analysis to develop and design Guardian Angel: a patient-centered health information system (GA: PCHIS), using the seven agent methodologies selected in this research. The goal of this case study is to explore the potentials of each approach in presenting solutions for GA: PCHIS problems. In addition, an attempt is made as to discover whether an approach is comprehensible and applicable in practice. For this purpose, numerous notes were taken and compiled as to the advantages and weakness of these methodologies during the study analysis, as well as the relationships among their models.

### 4.1.1 An Overview of Exemplar

Agent -based systems have the ability to provide greater pliability, improved careers, better power, reliability and security than traditional information systems. All the selected methodologies are used in this research to analyze and design the GA: PCHIS [1] as another application for the agent-based so as to form the basis of the case study. Our pilot assessment discussed in the following section includes the application model, which, as an outcome, provides of agent-based methodologies used in this study in case of GA: PCHIS, patients receive surveillance and access to healthcare services from various healthcare providers available in the same way as it is done by hospitals, doctors, nurses, pharmacies, laboratories, clinics, emergency centers, consultants (Ericsson, 2004).

Healthcare systems have become inactive, cumbersome and costly. There are no studies and or good planning for healthcare operations, not to mention obstacles faced by patients ranging from healthcare records to referrals to specialists and prescriptions. These problems are related to cost control as well as poor quality of healthcare supplier systems themselves (Ericsson, 2004).

The objectives of the GA system are to build data systems focused on patients rather than service suppliers. In this system, a group of agents collect all health-related data related to the health status and financial details of individuals in order to follow the health history of the subject and advise the patient and the supplier accordingly. The system is devised so as to preserve inclusive, progressive, correct and consistent medical records, and can be accessed in a timely way as the priority shifts throughout the individual's lifespan and in business tasks and with regard to healthcare suppliers' operations .

All GA is considered as an energetic operation which carries out many paramount assignments: it gathers patient information, examines, explains and clarifies the subject and facts, and plans medicinal pertinent. The system adapts its advice in accordance to past experience with the subject, conducts "safety tests" on medical efficiency and the cost of diagnosis and treatment plans, monitors the progress with program agents from service providers and insurers, and helps to encourage and

educate the patient. All of these operations improve the quality of medical procedures and solutions, increases patient's confidence, and reduces exposure to disease and medical errors(Szolovits et al., 1994).

Agent technology can be used to achieve greater functionalities and flexibilities in such type of systems. For this, effort is being made here to build an agent information system by applying the GA: PCHIS on the seven AO methodologies selected. The system is established so as to represent the hospital, family at home and healthcare providers all in one place.

### 4.1.2 Exemplar Scenario

The beginning of the GA scenario is that the patient wants his or her treatment to be managed efficiently. The scenario's path starts with registering the personal medical information via a measuring device. Then, the path moves to the data schedules stub, which is responsible for automatically downloading and storing the data into the internal note of the PDA, scheduling, and systematizing of this data. After all the processes related with data scheduling are completed, the next step is to upload the data by means of communicating with the home-computer by the PDA agent and display this data to the family. After thus, the path moves on to monitoring the treatment stub which follows the treatment plan and patient's condition.

In case any problem occurs throughout the process, of treatment, the PDA notifies the home-computer. After that, the path moves on to making suggestions and introducing alternatives for the problem to be dealt with. If the family does not feel content with following these proposals, the next path is to request communication with the healthcare provider agent, where there are many other choices for contacting, such as emails and emergency phone numbers.

After processing the request, the path moves on to access control, which receives the request, checks the list of participants and notifies the home-computer agent. The request has two responses, a and b; 'a' is followed when the request is accepted and moves the path to provide authorization, and 'b' is followed when the request is rejected. The authorization process gives the participant permission to access their

medical information, and this process also has a path that follows two responses, a and b; ‘a’ is followed when the healthcare provider agent needs the medical history of the patient and moves the path to access this medical history; the ‘b’ port is followed when the path moves to personal medical information without the need for access to the medical history.

The personal medical information path provides the related information and, then, the path moves to the alternative treatment plan to provide the patient through the PDA with alternative treatments. If the parents agrees with the proposals, then the new information is uploaded to the hospital, which add these recent data to the medical records of the patient. Later, the path moves to update this medical history. In the next sections, we will summarize our notes and comments on these methodologies.

### 4.1.3 Gaia Methodology

The five models generated by Gaia (Wooldridge et al., 2000;Zambonelli et al., 2005) methodology are examined in two stages: the first is the analysis stage, and the second is represented in the design phase. Figure 9, extracted from (Wooldridge et al., 2000) depicts Gaia Models. Below is a detailed description of each development stage.

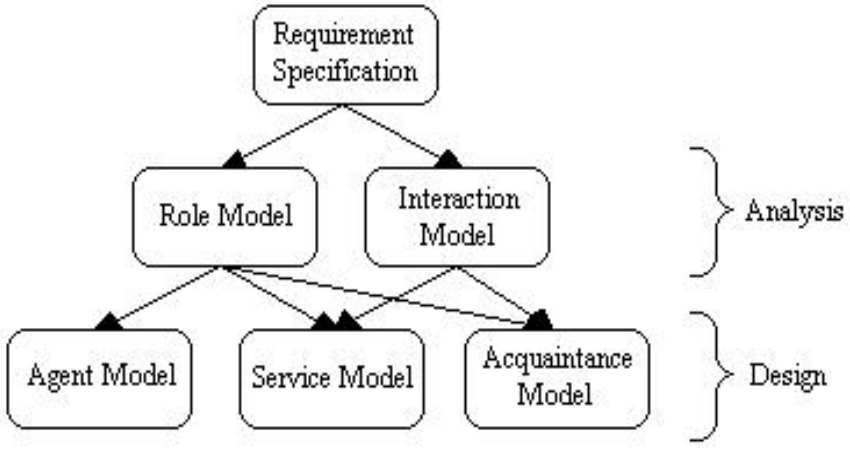


Figure 9. Gaia methodology models (Wooldridge et al.,2000)

**- Analysis Stage**

This stage covers the roles and the interactions to form a conceptual model of the GA: PCHIS. Different roles are available in any MAS, the main aim of this model is to distinguish the system roles, to define these roles and clarify their functions. Based on a number of protocols, these roles interact with each other to achieve the main goals of the system. Each protocol defines the purpose, the initiator, the responder, the inputs, the outputs and the processing during the interaction.

Figure 10. shows an example of two protocol definitions: one for querypatientpreference and the other for „replypatientoptions. The Figure illustrates that the protocol querypatientpreference is initiated by the role PDA and includes finding out about the patient‟s preference. The second protocol, on the other hand, includes the protocol replypatientoptions and the role PDA is now the responder.

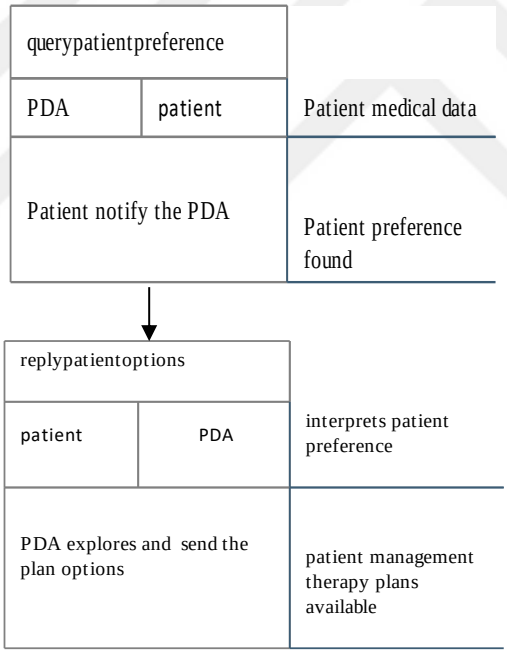


Figure 10. The querypatientpreference and replypatientoptions protocol definitions

In the last step of the Gaia analysis phase, one can define the main roles identified in the first step. This process involves the identification of permissions of roles and their responsibilities in addition to the protocols and activities in which they participate. This detailed description of a role is depicted by a Role Schemata. A set of role

schemata forms the Role Model, which is considered as the key artifact of this analysis phase.

There are two forms of functionality when it comes to the issue of responsibility: liveness properties and safety properties (Wooldridge et al., 2000). Permissions define which resources the agents playing that role can and cannot use when performing a particular action. Activities are private" actions which do not involve interactions with other roles. Protocols are actions dealing with interactions related to other roles. They are taken from the protocol model which is constructed in the step before. Figure 11 shows a role schema for the role PDA.

| ROLE SCHEMA: PDA |
| --- |
| Description: This role presents management therapy plan options,<br>monitor and customize treatment |
| Protocols: get patient data **involves** Store patient's information. Data<br>sorting. Data schedule<br>Monitor treatment **involves** monitoring tests results.<br>Interface with devices. Get results from patient &Get<br>information about meals and exercise & Assessment<br>treatment compliance. Suggest changes in diet orexercise.<br>Suggest changes in treatment plan<br>Data Update **involves** update results. Update changes in medicines |
| Permission<br>Reads Patient's condition<br>Reads Patient's preference<br>Monitors progress<br>Changes plan follow Communications<br>Allows patient to customize therapy plan |
| Responsibilities:<br>Liveness:<br>PDA = $(All)^{\omega}$<br>PDA = get patient data, Monitors progress of patient<br>condition, Data Update<br><br>Safety:<br>Consistent (patient's medical information)<br>Consistent (patient's profile) |

Figure 11. Role schema for role PDA

This role involves detecting and interpreting the facts and medically-related plans and options depending on the patient preference and available activities. There are several protocols associated with this role, such as querypatientpreference, replypatientoptions, etc. This role has the permission to read the patient's condition, monitor the progress, and allow the patient to customize the therapy plan. The liveness property of the role PDA indicates the sequential execution of its associated protocols and activities in patient treatment.

**Design Stage**

All analysis models are carried into the design phase. During this process, they are updated to reflect the design decisions. Three design models - the agent model, the service model and the acquaintance model - are created from the updated analysis models. These models are refined iteratively until sufficient design information is finally attained. The agent model is the first step in the design process when map roles are identified within the analysis process.

In the design phase, this model is created by assigning roles to agent types, A role can be mapped to one or more agent types. For every agent role, a number of related services can be specified as to properties such as inputs, outputs, pre-conditions, post-conditions. The inputs and outputs are derived from the protocol model. The pre- and post-conditions which define the constraints on the services are derived from the safety properties of a role.

An example of the Service Model is shown in Table 1. The GA-PDA agent playing the PDA role has three main services: receive patient data, monitor treatment, and update data. Each of these services is described with its inputs, outputs, pre-conditions, and post-conditions.

Table 2 .Service model for GA-PDA agent

| SERVICE | INPUT | OUTPUT | PRE-CONDITION | POST-CONDITION |
|---|---|---|---|---|
| Get patient data | patient-data | patient-preference | patient-preference=nil | (patient-preference=nil) V (patient-preference!=nil) |
| Monitor treatment | medical-data | alternative treatment | alternative treatment=nil | (alternative treatment=nil) V(alternative treatment!=nil) |
| Data Update | recent data available | data-processing | data-processing=nil | (data-processing=nil )v (data-processing!=nil) |

### 4.1.4 Tropos Methodology

Tropos methodology (Bresciani et al., 2004), agent-related concepts such as goals, plans, tasks, etc. are included in all of the development phases with the purpose to cover a wider range of software development life-cycle activities. The work stages in this methodology are divided to five stages: early requirements, late requirements, architectural design, detailed design and implementation phase. A main difference between Tropos and others is the emphasis put upon analysis of early requirements.

**-Early Requirements**

The main objective of early requirements analysis is to determine the stakeholders in the target system and their intentions. To model stakeholders and intentions respectively Tropos uses the concepts of actors and goals. The goals are divided into two different groups. In the end, hard-goals result in functional requirements, whereas soft-goals deal with non-functional requirements. There are two models that represent them at this point in the Tropos. Firstly, the actor diagram describe the stakeholders and their relationships in the domain. Secondly, are called social dependencies that

depicts how actors depend on each other for goals to be accomplished, plans to be executed, and resources to be provided.

Next, the goal diagram displays goals and plans analyzed in terms of the specific actor in charge of realizing them. Figure 12 shows the dependency of GA:PCHIS to provide information (hard goal). It also needs a usable GA:PCHIS (soft-goal). These goals are then decomposed into sub-goals. For example, the goal 'provide information' is fulfilled by the composite achievement of two sub-goals 'search information' and 'display output'. The sub-goal 'search information', in turn, has several sub-goals such as 'access to patient data', 'access to physician data' and 'access to pharmacy data'. The soft-goal 'easy to use' is also shown. A user-friendly interface that offers simplicity and provides guidelines also realizes the goal 'easy to use'.

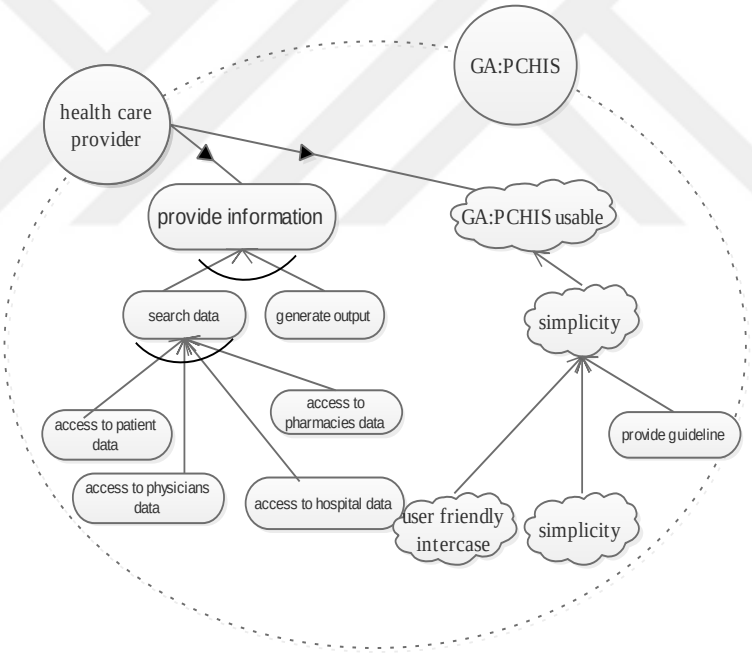


Figure 12. GA:PCHIS goal diagram

**- Late Requirements**

Late requirements stage modeling the target system in its environment. The system intended for use is modeled in the form of one or more actors. The dependencies of these special actors are also identified by following a similar process to that used in the Early Requirements phase. In reality, such dependencies clarify

both the requirements regarding the functional and non-functional aspects of the system.

**- Architectural Design**

During the architectural design, Tropos methodology determines three steps which system developers can apply in this phase. The purpose of the first step is to define the comprehensive architectural organization of the target system; whereas the second step include identifying the capabilities required by the actors to achieve their goals and plans. In the final step, agent types are defined together with the allocation of functionality to be assigned to them. Figure 13 shows the decomposition in the sub-actors of the target system and the delegation of some of the goals from the GA: PCHIS to them.

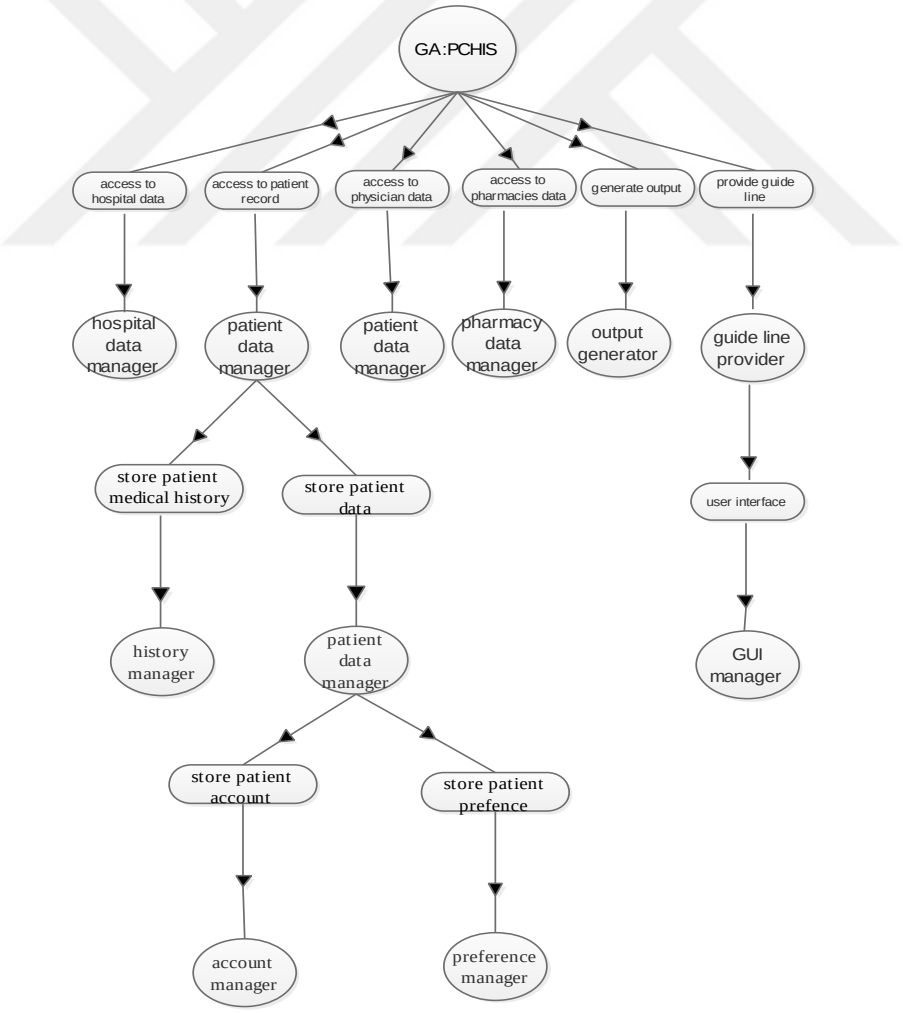


Figure 13. Actor diagram for the GA:PCHIS architecture

The system depends on the patient database manager to have access to the patient database, the hospitals database manager to have access to hospitals data, the output generator to generate output, and so on. In addition, each sub-actor (e.g., patient Info Manager) can itself be decomposed into sub-actors (e.g., account manager and preference manager) responsible for the achievement of one or more goals (e.g., store patient account and store patient preference).

### 4.1.5 PASSI Methodology

PASSI (Cossentino, 2005) is a step-by-step requirement-to-code approach for designing multi-agent software and combines notions and ideas from both object-oriented software engineering and MAS utilized by extending the UML models more evenly. PASSI's activities comprise five steps and are distinguished by their work definition (Figure 14).

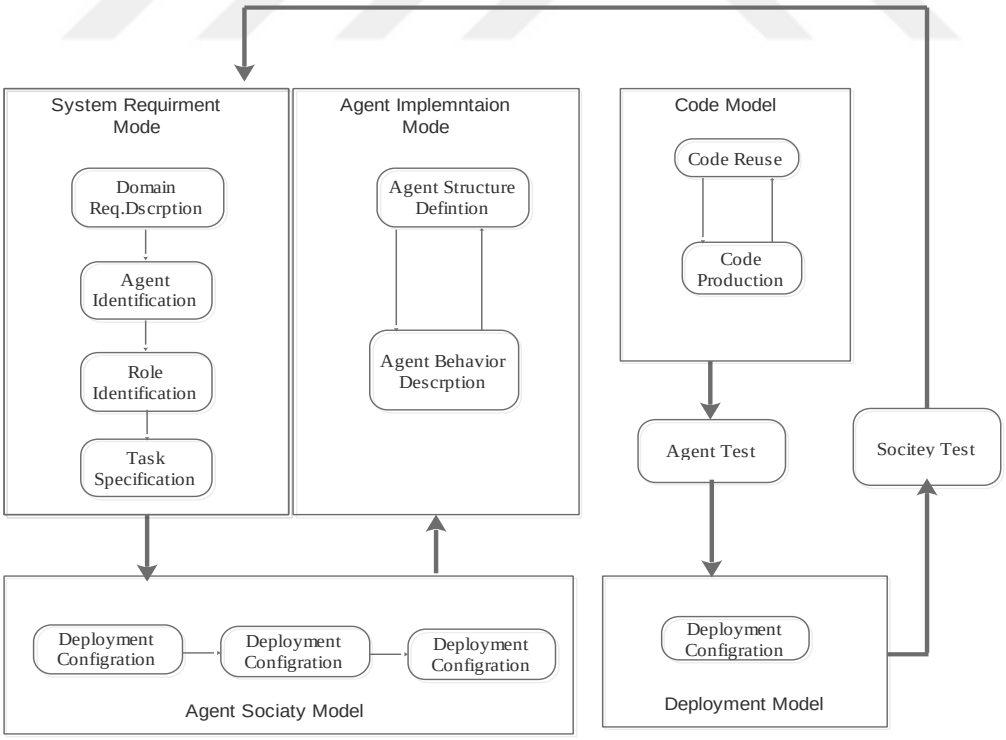


Figure 14. PASSI methodology models (Cossentino, 2005)

**-Domain Description Phase**

This stage comes as an outcome of a functional description of the system and is composed of a sequence of UCs. Then, the screenplays of the elaborate UC schemas are demonstrated which employ sequence diagrams. Figure 8 shows a domain description diagram depicting our analysis for the GA:PCHIS case study. The stereotypes applied here are driven from the UML criterion. The convention adopted for relationships between the external actors and the system is to direct the arrows from the communication's initiator to the participant.

**-Agent Identification Phase**

It begins with the UC diagrams of the preceding step. At this stage, it is possible to see an agent as a UC or a pack in the functional decomposition of the preceding step. Beginning from an adequate and elaborate schema of the system functionalities (Figure 15), we set one or more UCs into these stereotyped packs so as to compose a novel model (Figure 16).

As a result, each pack realizes the functionalities of a certain agent. Figure 16 illustrates the portion of the agent identification model that covers only the agents included in the screenplay. Connections among the UCs of the same agent follow the customary UML syntax and stereotypes (see PDA and home computer agents in Figure 16), while the connections among the USs of various agents are stereotyped as "communication". This model guides the interactions among agents, from the initiator to the entrant.

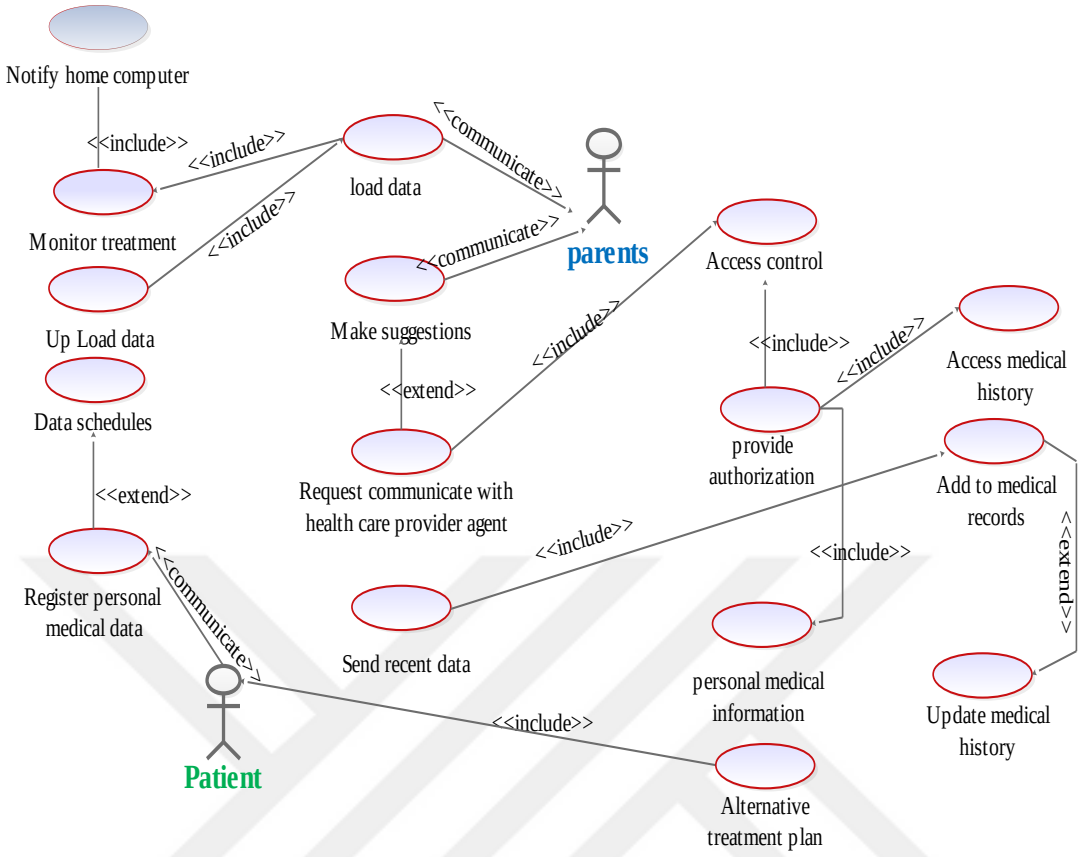


Figure 15. A portion of domain description diagram

**-Roles Identification Phase**

This phase is based on the discovery of all potential routes of the agent identification scheme and includes one-to-one connections between agents. The path characterizes the scenario of connecting agents that act to fulfill the behavior conduct expected for the system, and comprises several connection routes. It has to be stated that the connection path is merely a "communication" relation among the two agents in the model. Each may belong to several scenarios, which are depicted by serial graphs in which objects are utilized to denote the roles. Figure 17 shows the presented scenario, occurring when patient information is processed from the role data processor of the PDA agent to the role data loader of the home computer agent. Although the diagram resembles an UML sequence diagram, the syntax is slightly different.

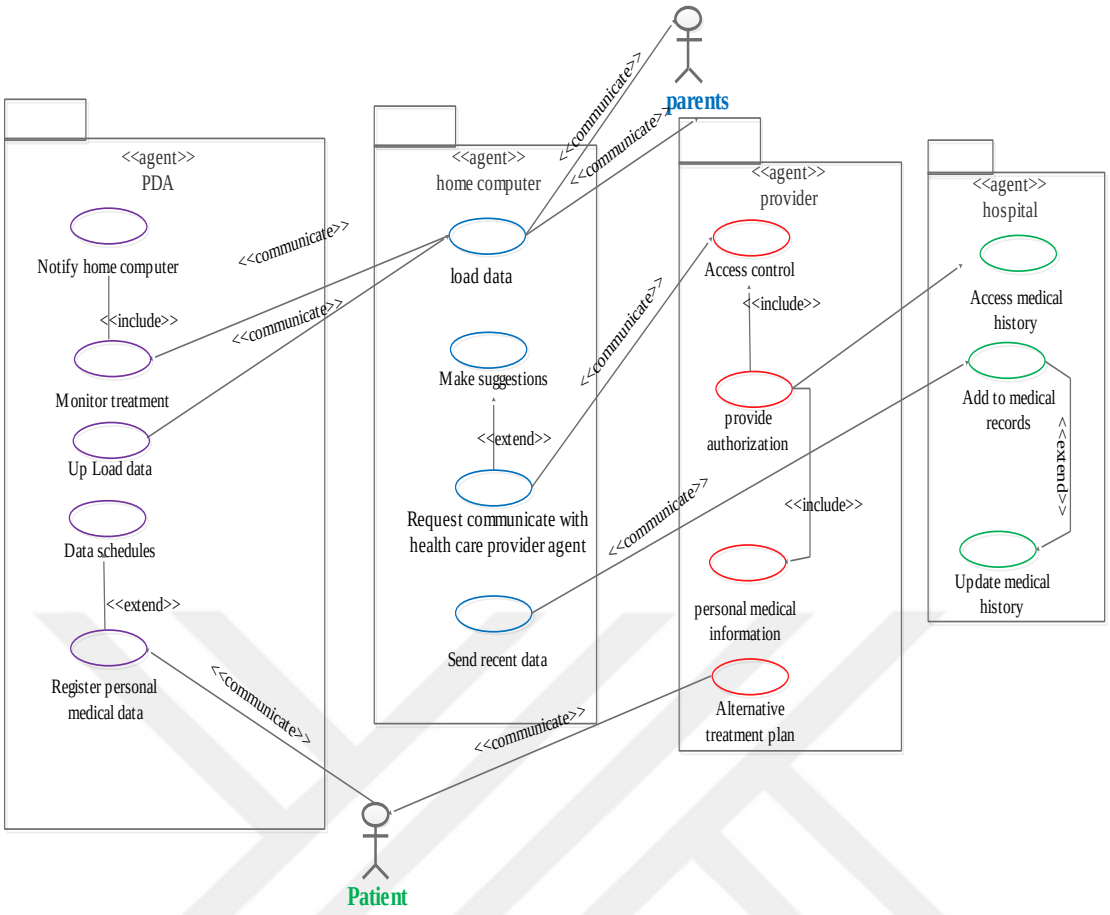


Figure 16. Agents identification diagram

**-Task Specification Phase**

This stage has been suggested to epitomize the agent's abilities and neglect the data about roles that an agent can take. The relations among these processes indicate either connections among the functions of the same agent or connections among the functions and other communicating agents.

**-Ontology Description Phase**

In the PASSI approach, the developing of ontology is conducted in the range ontology depiction stage, where a class schema is utilized. It characterizes the ontology in terms of notions (classes and elements of the range), predicates (with emphasis on the attributes of notions), events (taking place in the range), and their interactions.

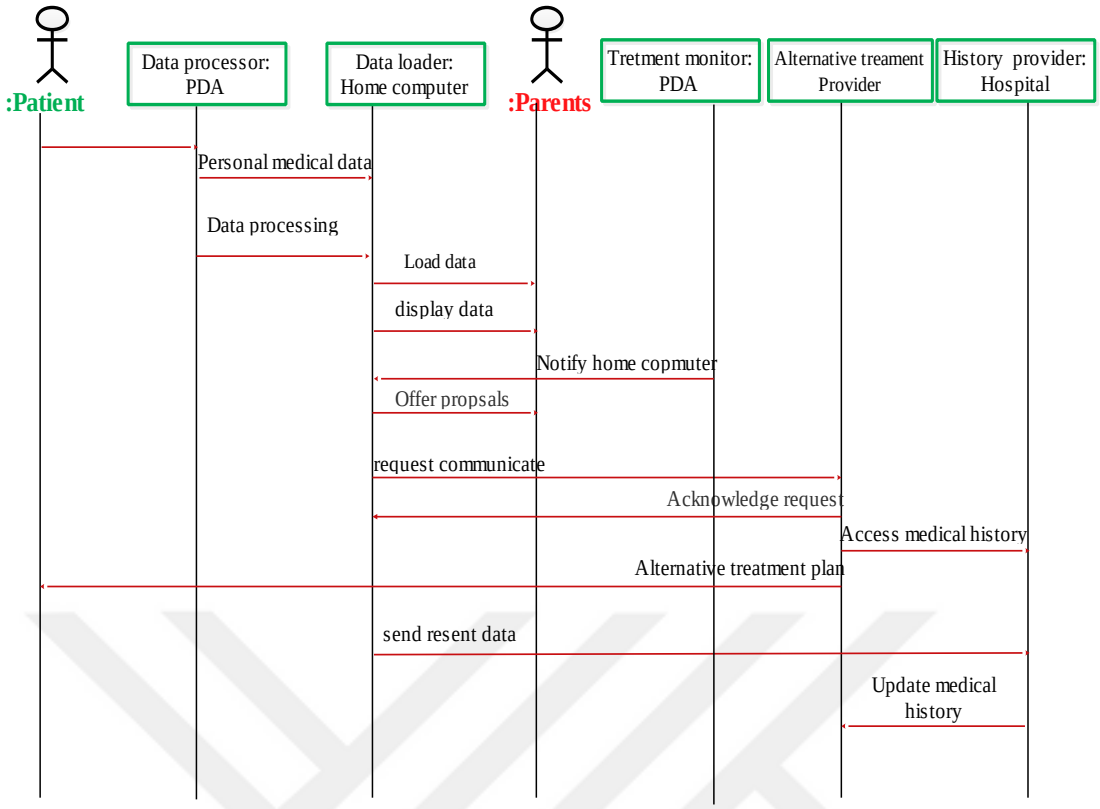


Figure 17. The roles identification diagram

### 4.1.6 O-MaSE Methodology

The O-MaSE approach (DeLoach & Garcia-Ojeda, 2010) (DeLoach & Garcia-Ojeda, 2014) was presented from the start as a group of fragments that could be collected by designers to achieve the specific demands of their target system. O-MaSE employs a consolidated meta-model-based framework that permits developers to choose method fragments from a warehouse and build custom operations utilizing a specific structure and guidelines (Firesmith & Henderson-Sellers, 2002). There are three major stages involved in their operations and the process developer is able to determine their own group of steps and iterations as well as define activities and tasks for those steps and iterations as convenient.

**-Requirements Analysis**

This stage consists of six work products: system description specification, goal model, GMoDS model, domain Model, organization model, and role model.

**-Goal Model**

The top-level GA:PCHIS goal model is shown in Figure 11 with the overall goal as broken down into five sub-goals: StoreDataInPDA, DataProcessed, UpdateHomeComputer, AccessControl, and UpdateMedicalHistory. The AccessControl goals is further decomposed into three sub-goals: ReceiveRequest RequestProcessed and ProvideAuthorization, The UpdateMedicalHistory goal is also decomposed into the ReceiveData, AccessMedicalHistory and MedicalDataUpdate sub-goals.

The goal model allows for the creation of new target instances through the event trigger , which is indicated by an arrow between two goals labeled by an event signature, such as upload (d, p) on the arrow between the StoreDataInPDA and DataProcessed goals. In this case, it means that when an upload even occurs during the pursuit of the StoreDataInPDA goal, a new DataProcessed goal is created.

We use this to create a DataProcessed goal for each patient data uploaded to be processed. Similarly, the send event during the pursuit of the UpdateHomeComputer goal will create a new ReceiveRequest goal, which receives the request and checks the list of the participants. The goal model uses a precedence relationship to guarantee the correct sequence of actions in the system.

Thus, since the data must be registered and stored before it can be uploaded to the home computer, the StoreDataInPDA goal precedes the UpdateHomeComputer goal. In addition, the patient information must be processed (DataProceesed) before it is sent to the home completer (UpdateHomeComputer) . The StoreDataInPDA and UpdateHomeComputer goals are further decomposed as shown in Figure 19. The StoreDataInPDA goal is decomposed

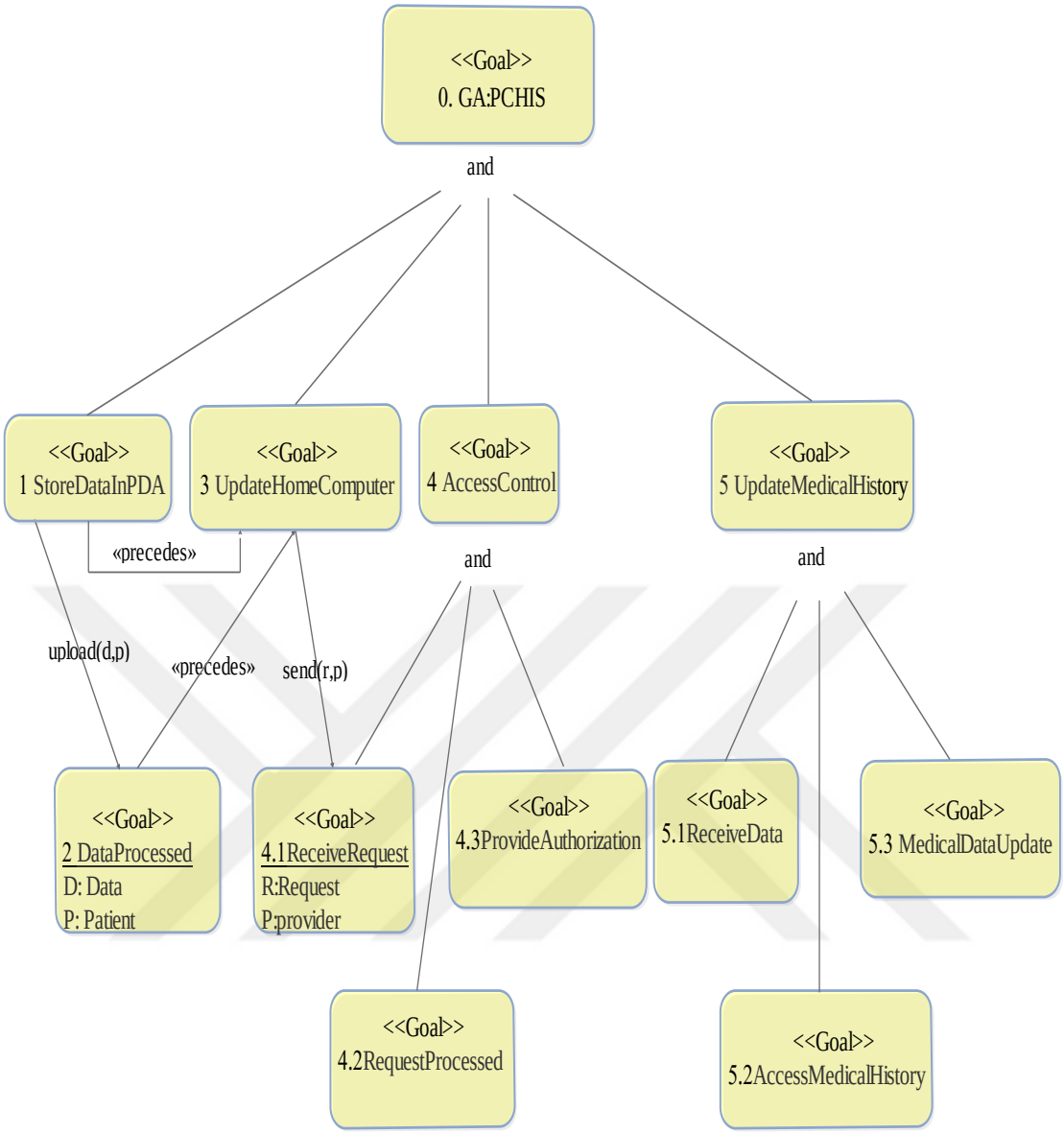


Figure 18. Top-level goal model for GA:PCHIS

in the DataEntred and MonitorTreatment goals. MonitorTreatment also decomposed in the MakeSuggestions and ResultsEvalute. The UpdateHomeComputer goal is decomposed into two sub-goals: ReciveData, StoreInHomeComputer, which decomposed in the CommunicateHealthCareProvider and ModifyPDA.

-O**rganizational Model**

The Organization Model for the GA: PCHIS project is shown in Figure 20. There are three external actors: patient, hospital manager, and parent. Every connection with the system is made using specific protocols. With the organization refining into role and class diagrams, these external actors and protocols should be presented in a harmonic way.

**-Role Model**

Figure 14 shows the role model for the top-level GA: PCHIS system. There are four specific roles to implement the objectives set out in Figure 21, each developed to fulfill one goal as described by the <<Achieve >> attribute in each role. In this respect, the two exceptions are the provider role, which is designed to provide the patient through the PDA with alternative treatment plans and medical data; and the PDA role, which is designed to schedule the data and monitor the treatment.

The external actors which interact with each role are also illustrated in this diagram. In the model, the protocols that support the passing of information appear as arrows between the roles and between actors and roles. Here is a summary of how the system works: The process starts with the patient actor registering the personal data with the PDA role which is communicates with home computer role to upload the information.

Then, the home computer role displays this information to the parents or family and sends a request to communicate with the provider role in the event of any problem through the treatment plan. Next, the hospital computer role which interacts with the hospital manager actor updates the medical history and allows the provider role to access the medical history and, subsequently, provide the alternative treatment plan to the PDA.

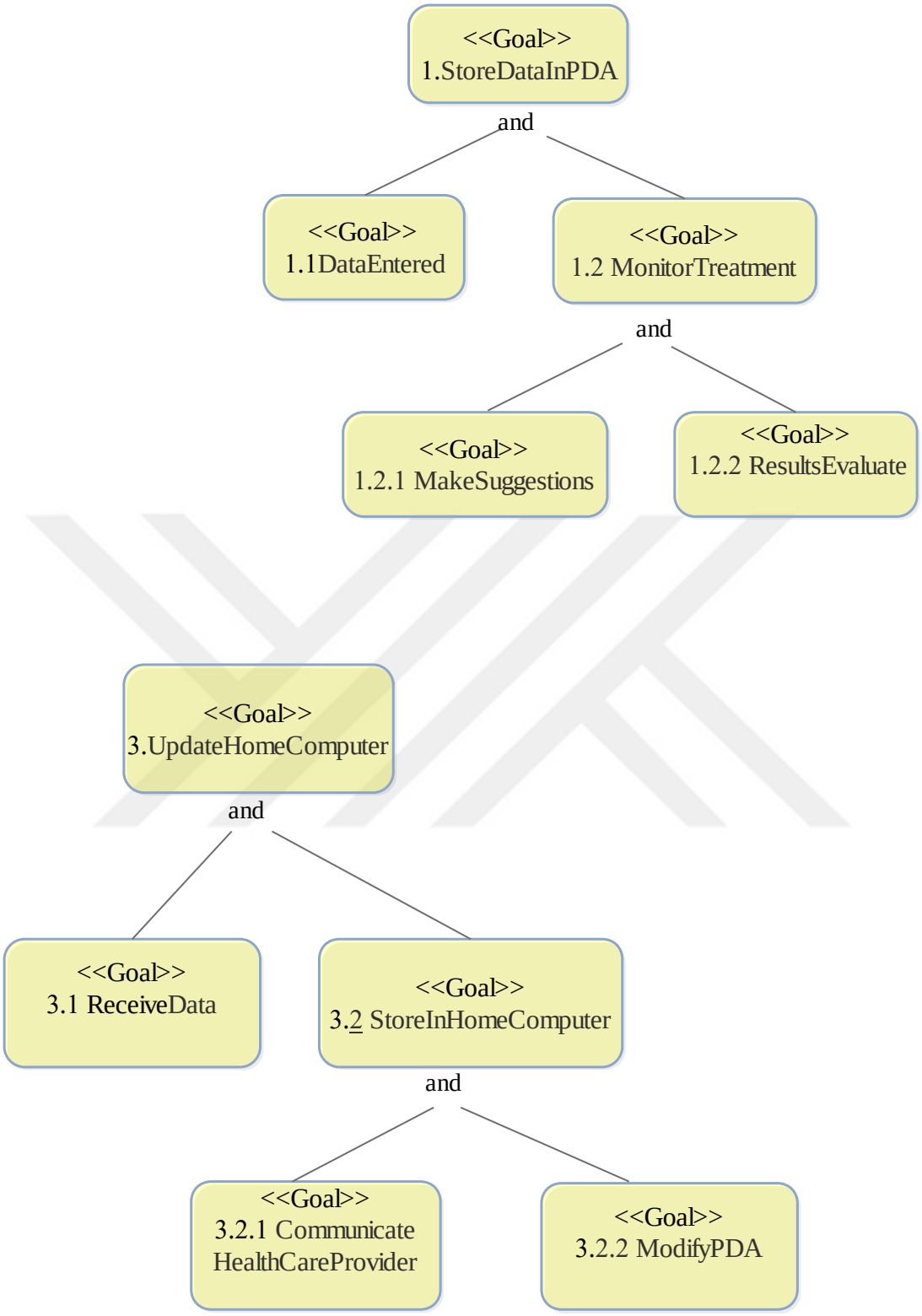


Figure 19. Second-level goal model for GA:PCHIS

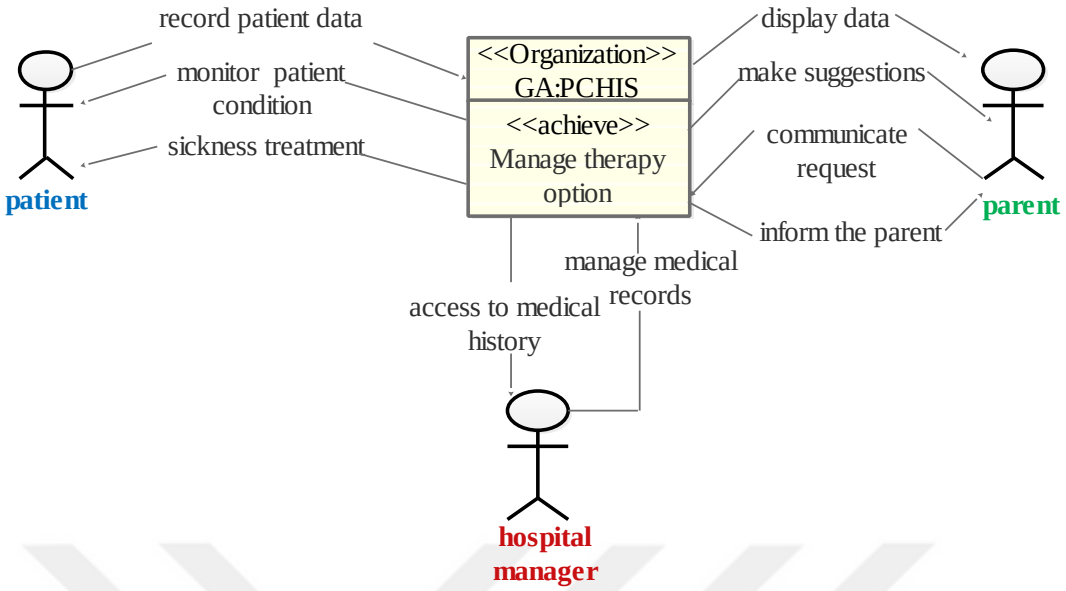


Figure 20. Organization model for GA:PCHIS

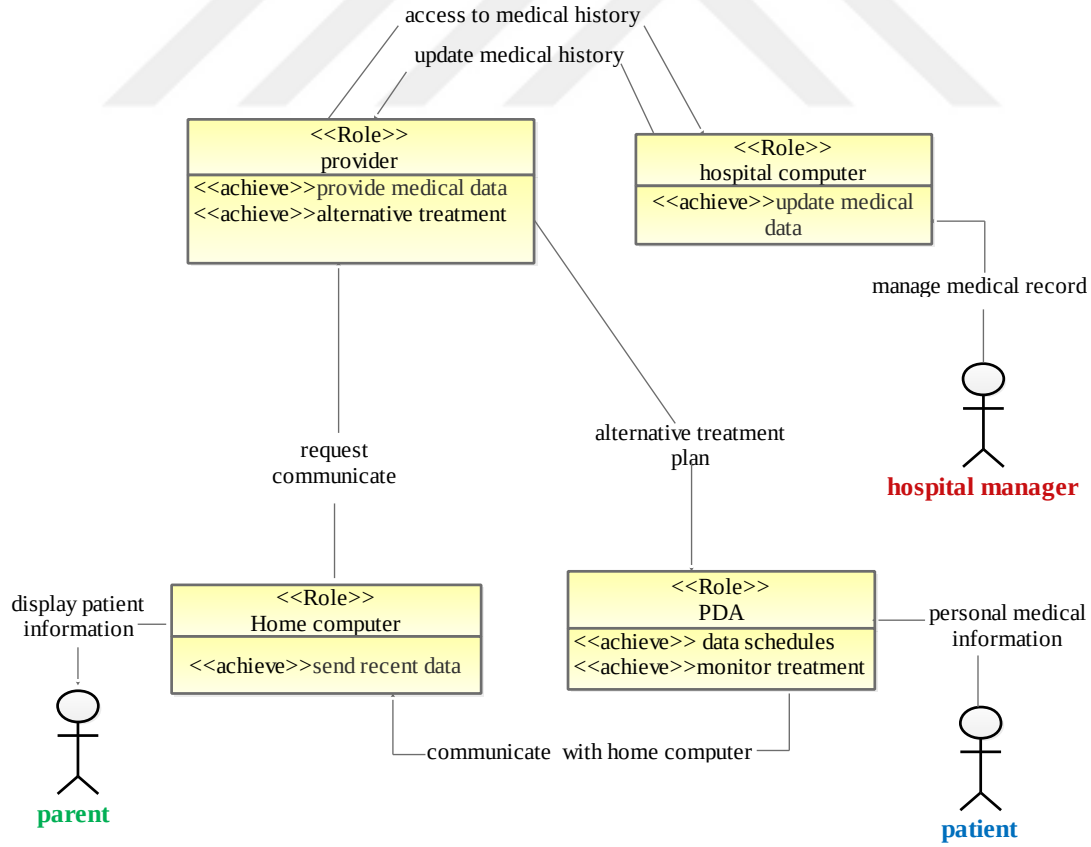


Figure 21. Role model for GA:PCHIS

### 4.1.7 ADELFE Methodology

As mentioned earlier, the ADELFE approach (Bernon et al, 2002) has been presented to focus on several aspect left disregarded by current approaches. It unifies AMAS theory (Adaptive Multi-Agent Systems) and offers a specific process derived from an interpretation of RUP (Rational Unified Process). This methodology suggests six stages, whose processes and activities are shown in Figure 22. Only the work definitions of demands, analysis and design need changes based on the AMAS. The remainder of the RUP is utilized without any such amendments.

**-Preliminary & Final Requirements**

The purpose of this step is to realize a convention on the initial demands related with the characterization of the system and the surroundings in which the system will be distributed. It is created to determine what the most suitable system can be for end-users. The aim of the definitive requirements is to convert this seeing to a UC diagram, and arrangement the requirements (functional or not) .

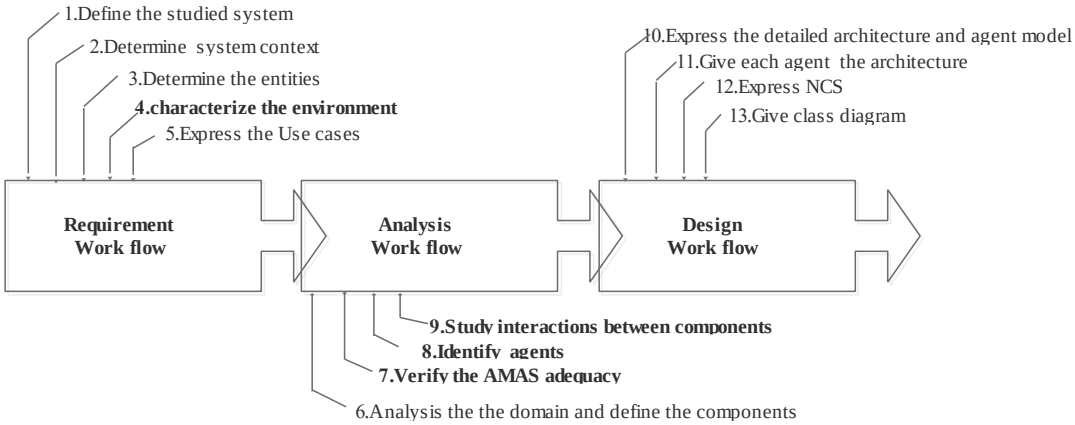


Figure 22. ADELFE methodology models (Bernon et al, 2002)

**-Determination of the Use Cases**

This processes illustrate the various tasks the system has to react to. This phase is divided into three steps; these are to build the UCs, elaborate the associated sequence models, and define cooperation failures. Only active elements are implicit in these UCs, which are the outcomes of a functional demands group. The UCs for the GA: PCHIS are shown in Figure 23.

**-Analysis**

From the perspective of a multi-agent approach, the definition of the agents should also be taken into account in this workflow. This processes have to provide for a thorough comprehension of the system and its structure in terms of ingredients, as well as a definition as to whether AMAS theory is an option. As shown in Figure 24, this stage describes the system components and examines the relations among them.

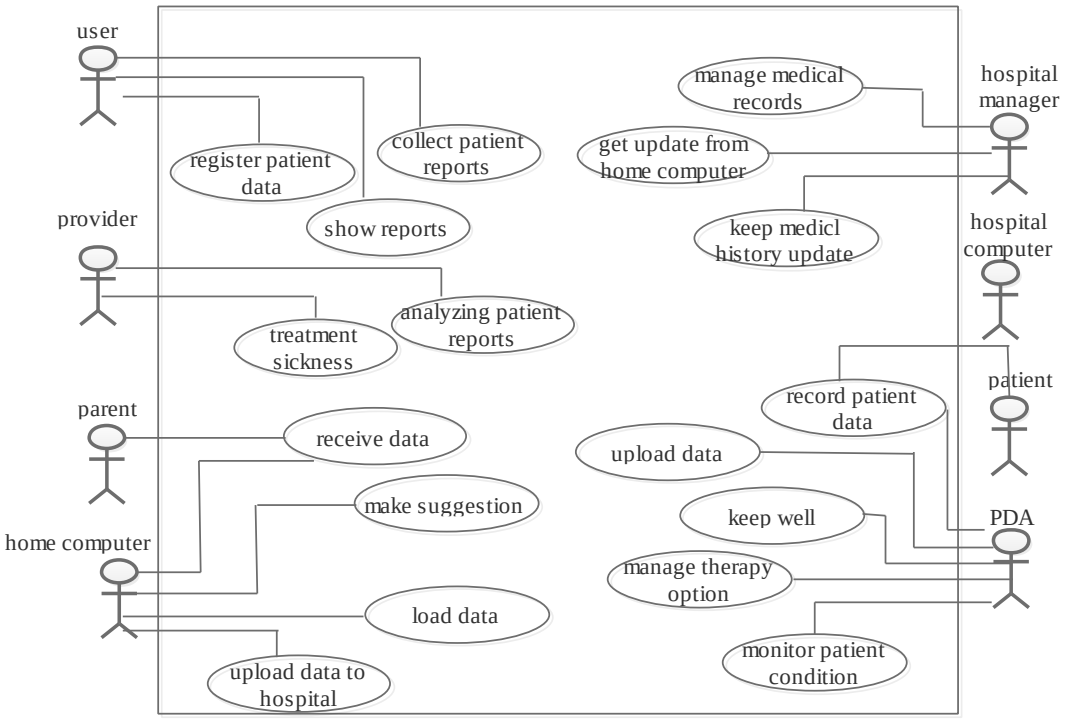


Figure 23. The use cases for the GA:PCHIS

**-Adequacy of AMAS Theory at the Local Level**

The first step, the adequacy of AMAS theory indicates a possible decomposition, and each agent has to be analyzed as a system. The identified agents have to be added to the preliminary class diagram as shown in figure 18. The main class diagram is for the GA: PCHIS problem. There are four classes of agents that appear: the PDA, the Provider, the Hospital and the Home computer.

**-Design**

The objective of this phase is to create models that concentrate on non-functional demands and the solutions' domain, and to prepare for the application of the system. ADELFE also offers a diagrams for developing cooperating agents, and the developer should define specific abilities for each kind of agent as well as the connection language and non-cooperative cases likely to be faced by the each agent.

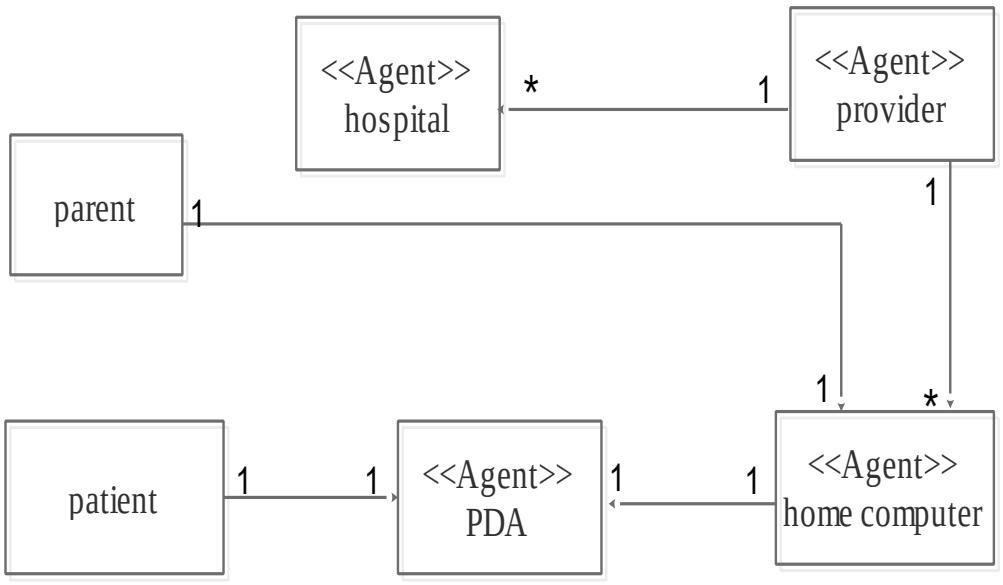


Figure 24. The class diagram for the GA:PCHIS

### 4.1.8 ASEME Methodology

ASEME methodology (Spanoudakis, 2009) and (Spanoudakis & Moraitis, 2011) supports the requirements, design, and implementation phases. Also, it covers a modular agent development method and offers the concepts of intra agent control which determines the agent's behavior by formatting the various modules which execute his capabilities. And the inter-agent control (IAC) defines the protocols that control the coordination of the agents society(see figure 25).

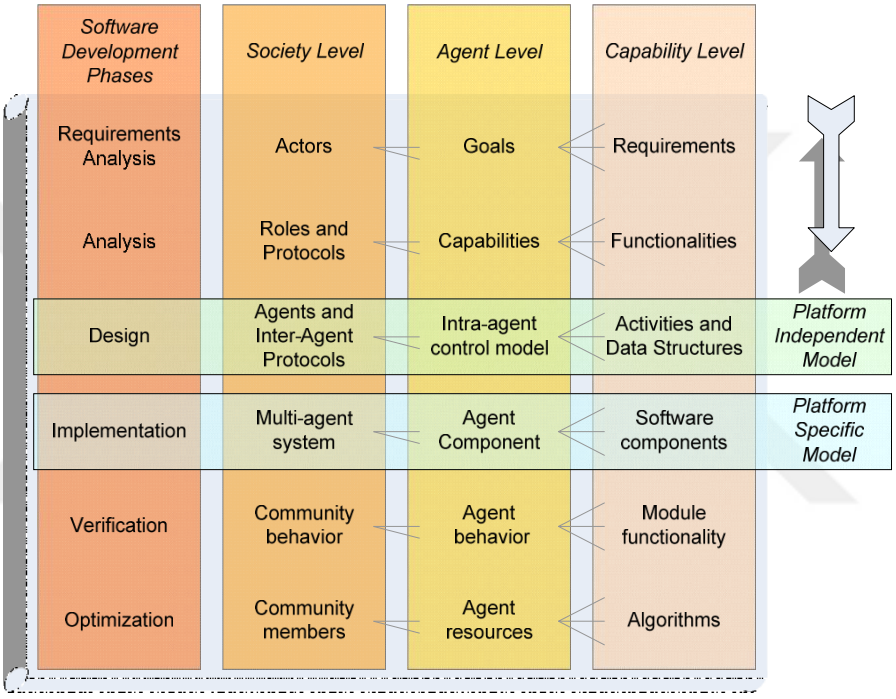


Figure 25. ASEME methodology phases (Spanoudakis & Moraitis, 2011)

**- Requirements Analysis Phase**

In this stage the participating actors are identified along with their respective goals. Furthermore it, information is collected about the specific requirements that dictate the expected system functions. At the first level, the analyst determines which actors will act in the system. In the second level actors is related with their goals. In the third level the specific requirements associated with each goal of each actor are

defined (Spanoudakis & Moraitis, 2008). Figure 26 describes the actor diagram for the target system.

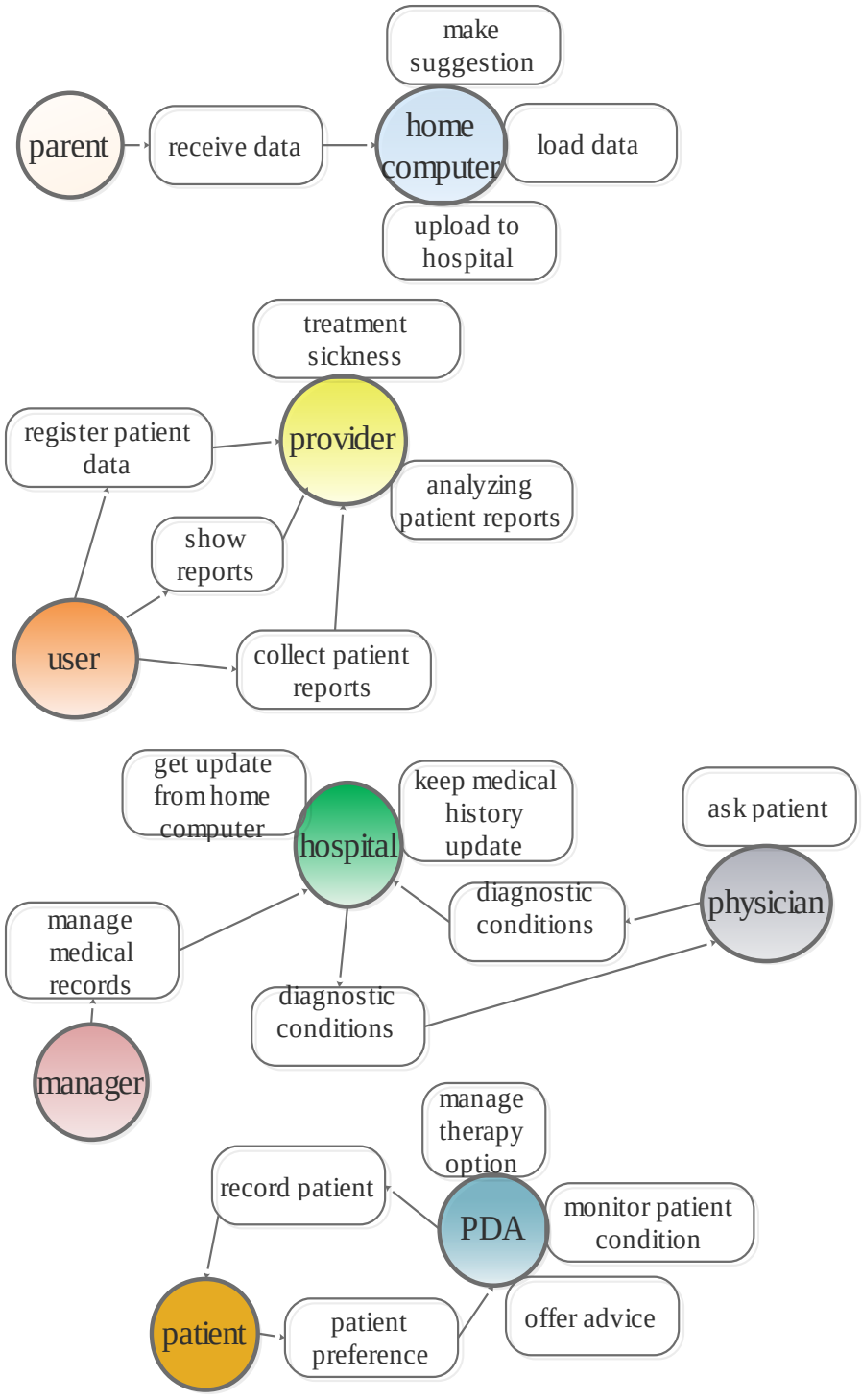


Figure 26. GA:PCHIS actor diagram

**- Analysis Phase**

In this stage there are main concepts: *roles*: human and agent roles can correspond to the actors of the requirements analysis phase as concrete roles. Roles are initially represented in the use case diagram and then moved to the roles model which includes the role name, the inter-agent protocols which the role participates in and its liveness model as shown in figure 27.

ROLE : PDA

Protocols: get patient data involves Store patient's information.

Liveness:

agent = (data schedules) $^{\omega}$ //(monitor patient condition$^{)\omega}$

data schedules =downloaded data. store data.

monitor patient condition = get patient data. data schedules. monitors progress*, data update.

Figure 27. The role model

**-Use Cases:** The use case model shows the relation between the system and its environment. Also defines the system functionality. Use cases are derived from the goals of the actor diagram in the previous phase (see Figure 28).

**-Capabilities:** In general, an agent's capabilities describe what the agent "can achieve". The capabilities which derived from the use case diagram are related to the tasks that the role can do by itself or through interaction with other roles.

**-Functionalities:** A functionality is associated to the various types of technologies that use to implementing reasoning mechanisms. *Activities*: Each capability is analyzed to simple activities. *Agent Interaction Protocols*: A protocol is implemented as a capability. *Liveness model*: A process model which shows the dynamic behavior of the roles in the system.

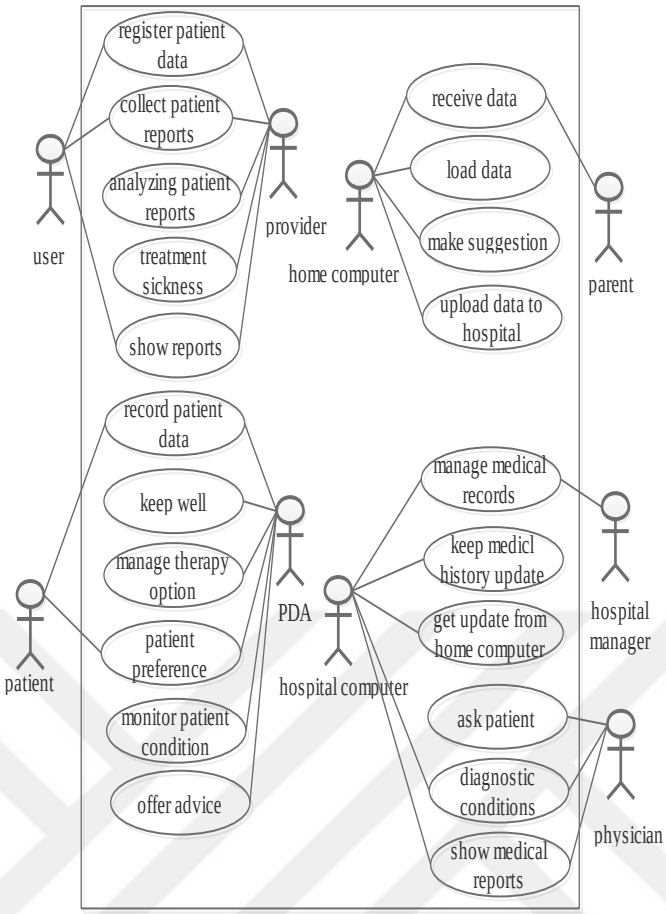


Figure 28. GA:PCHIS use case diagram

### 4.1.9 Prometheus Methodology

Prometheus (Padgham & Winikoff, 2005) is an itemized AOSE approach designed for non-professionals and apprentices. It has been effective as taught and applied by university students. Prometheus depends on three stages: system characterizing, structural design, and detailed design. Also, it is supported with two tools**.** Figure 29, extracted from (Padgham & Winikoff, 2005) depicts Prometheus Models. One of the main aspects of the Prometheus method in the architectural design stage is to identify the kinds of agent in the system. This is achieved by aggregating functions, and data coupling scheme is helpful in identifying potential clusters. Figure 30 displays a data coupling model for the next functionalities:

- Personal data management: responsible for automatically downloading, scheduling, and systematization of the data.

- Data up loader: communicates with the home-computer to upload patient data from the PDA agent to the home-computer agent.

- Access controller: receives the request, checks the participants' list, and notifies the home-computer agent.

- Authorization provider: provides authorization to the participant so as to obtain medical information.

- Access medical history:  to access the patient's medical history.

-Treatment monitor: follows the treatment plan and the patient's condition.

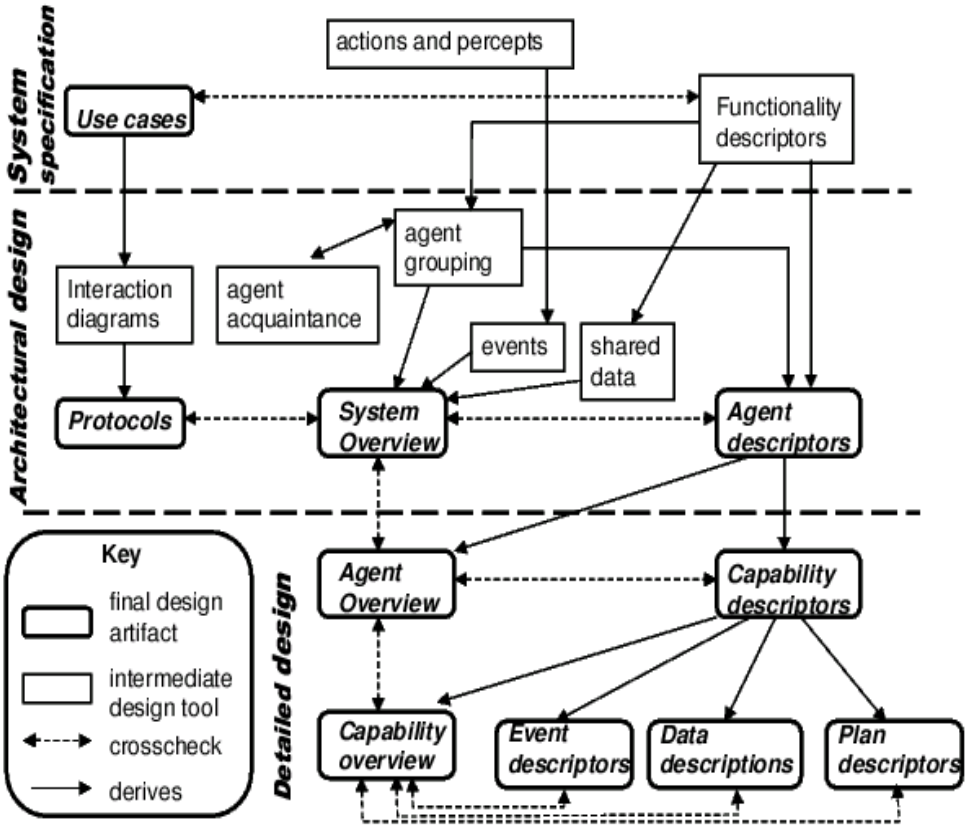


Figure 29. Prometheus methodology models(Padgham & Winikoff, 2005)

-User interaction: Interacts with the user, whether patients or their family members, as well as with the hospital manager.

-Parent/patient notify : Reminds family or patient of the events by means of several options for communication including electronic mail with attached content, directions for paging or an emergency phone number.

- Receives data: receives and loads the recent data to the home computer.

- Monitors progress: follows the treatment plan and patient's condition.
Under these conditions, there are five types of agents created as follows:

- PDA: based on the personal data management functionality and involving the data up loader database;

- User Interface: integrating the functionalities of user interaction and parent/patient notification. This is a logical combination because both of them are related to communication with the patients or their families;

- Provider: combining the access controller and authorization provider functionalities. These tasks are both linked and react with each other;

- Hospital manager: based on access to medical history tasks and including the treatment monitor; and

- Home computer: integrating the tasks of receiving data and monitoring the progress. These two functionalities are related with family or parents.

As outcome of this gathering is that several agents write data correspond the data that exists in other agents. For instance, the User Interaction task within the UserInterface agent writes to the provider database. These writings require to be converted to messages to the suitable agent, which then edits the content.

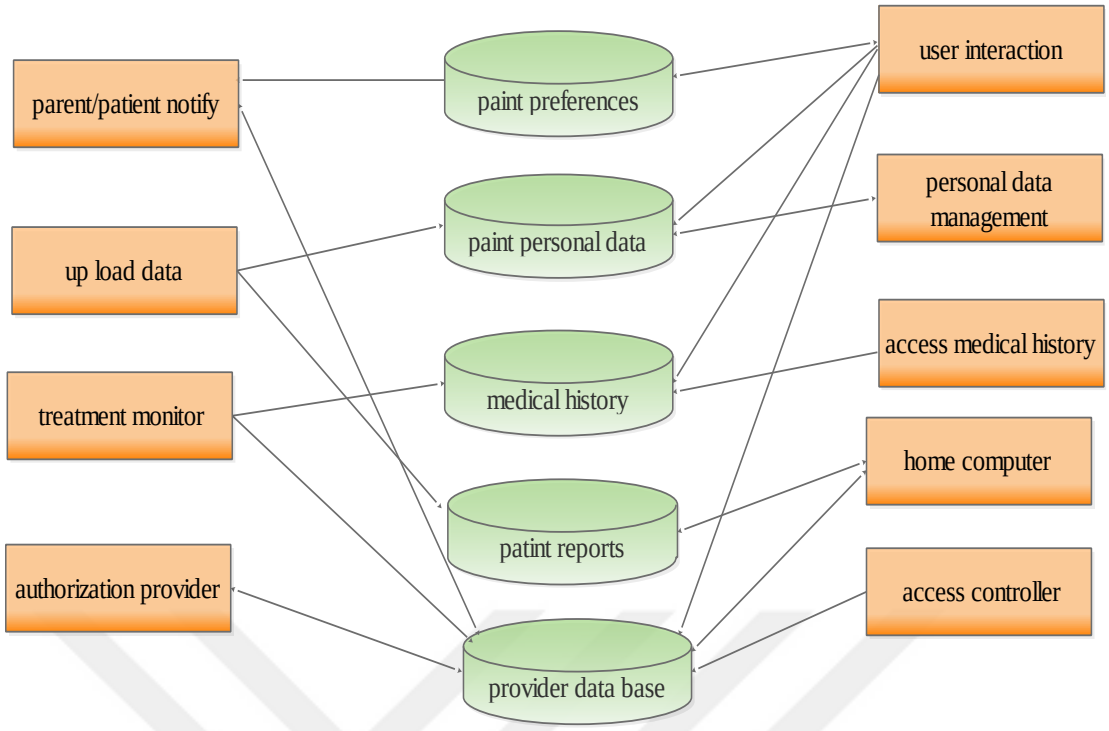


Figure 30. Data coupling model

After determining the kinds of agents, we now move to the development of the agent overview scheme from the overall system model. Figure 31 offers the general design of the system, which characterizes the four kinds of agents as specified and also displays the messages among them – that is, notions received by the User Interface agent and the information read and written .

### 4.2 Meta-Model

This is the third assessment method used in our evaluation experiment. This section will specifically discuss and compare the MAS meta-model for AO approaches as participants in this comparative study. The non-appearance of a unique MAS meta-model will eventually lead to a methodology that formulates its own concepts. According to the agent domain, this would render the component as important when executing the method-engineering model and with regard to diversity in the attitudes of the MAS meta-models. The configuration stage consists of many steps; however,

the first step should be choosing the mechanisms which establish the MAS meta-model to be constructed by designers.

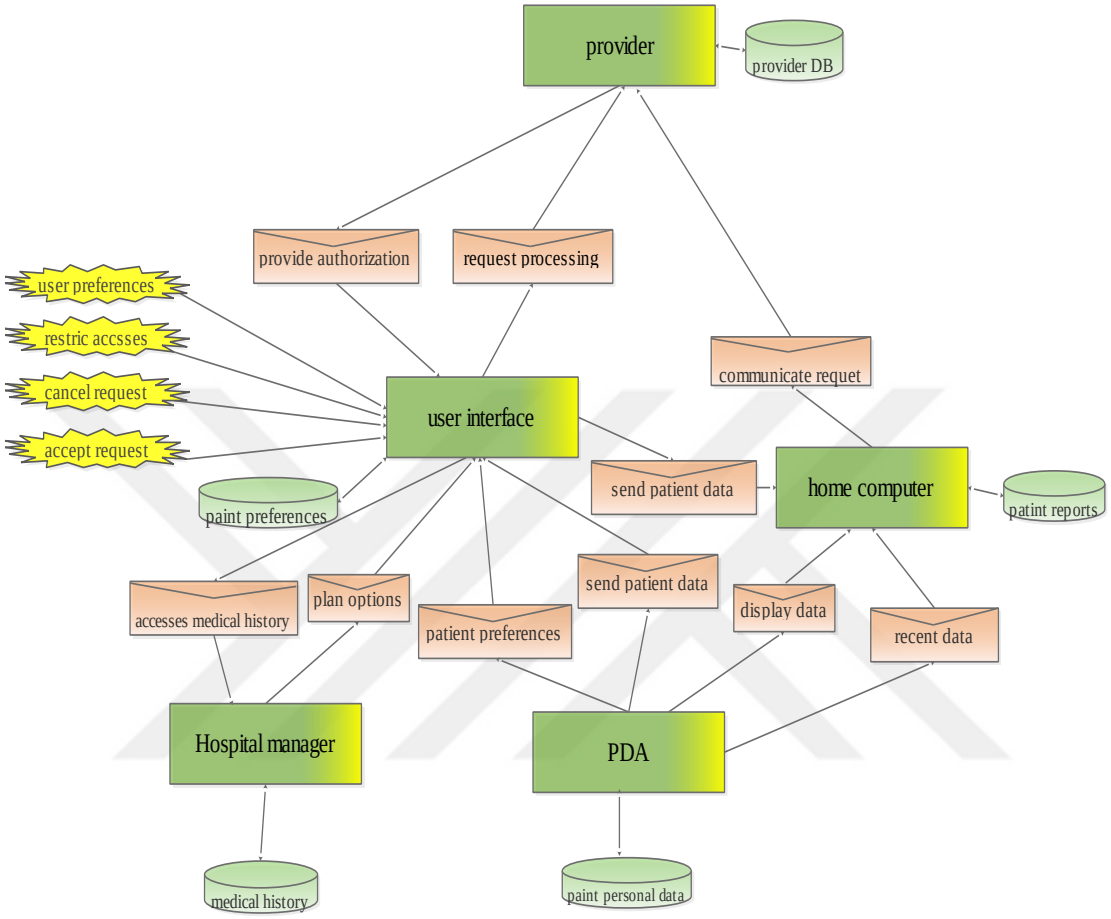


Figure 31. System overview diagram

By not considering the approaches which feature dissimilar components, the meta-model extracted in this way will be useful as a technique of fragment selection. Without a distinctive MAS meta-model, the different notions and system architectures which exemplify diverse approaches could result in very problematic or even impossible implementations in the process of fragment configuration (Bernon et al., 2004).

In the course of this study, the inspection of the seven MAS meta-models is implemented together with the design practices, thereby a united MAS meta-model.

These models are obtained using the integration of the most significant features of each meta-model with the intention to generate a notable and distinguished one.

### 4.2.1 Gaia Meta-Model

It is an established fact that the main objective of the Gaia approach is to deal efficiently with small-scale sealed agent-based structures (Wooldridge et al., 2000). It includes agents, characters, and relations while clearly ignoring the modeling of social features for a multi-agent scheme. The chief achievement of Gaia is based on the key notion that society is not just a group of roles and agents (Zambonelli et al., 2005). The main alteration, thus, is that it has been predesigned to responsively embody the societal features of open-agent systems while carefully considering the communal objectives and chores.

The MAS meta-model procedure concentrates on the managerial of the system and all other notions - roles, tasks collaborations, as indicated in Figure 32, in order to better recognize the links among different components in the framework of a definite society.

The elementary blocks of MAS meta-model – that is, agents, roles, actions, services and procedures – characterize the lengthy nature of Gaia. In more detail, the agent for instance is nothing but an entity responsible for performing a role or two by representing services to be initiated and de-initiated as per precise pre- and post-conditions (Bernon et al., 2004). As for the role itself, it is an accurate manner that has to be executed via an agent, explained in respect of consents, tasks, and undertaking and of its relations with other roles. These obvious notions are the basis for extending the Gaia design, which advances them by employing them in the setting of a precise atmosphere and of a precise society.

It must be stated that Gaia does not cope with the demands stage, and that it realizes such demands as an input for the approach. In order to paint a better picture of the whole system, the surroundings in which a MAS is operated is chosen at the initial analysis stage, and all objects and resources a MAS may relate with are clearly identified by the environment concept.

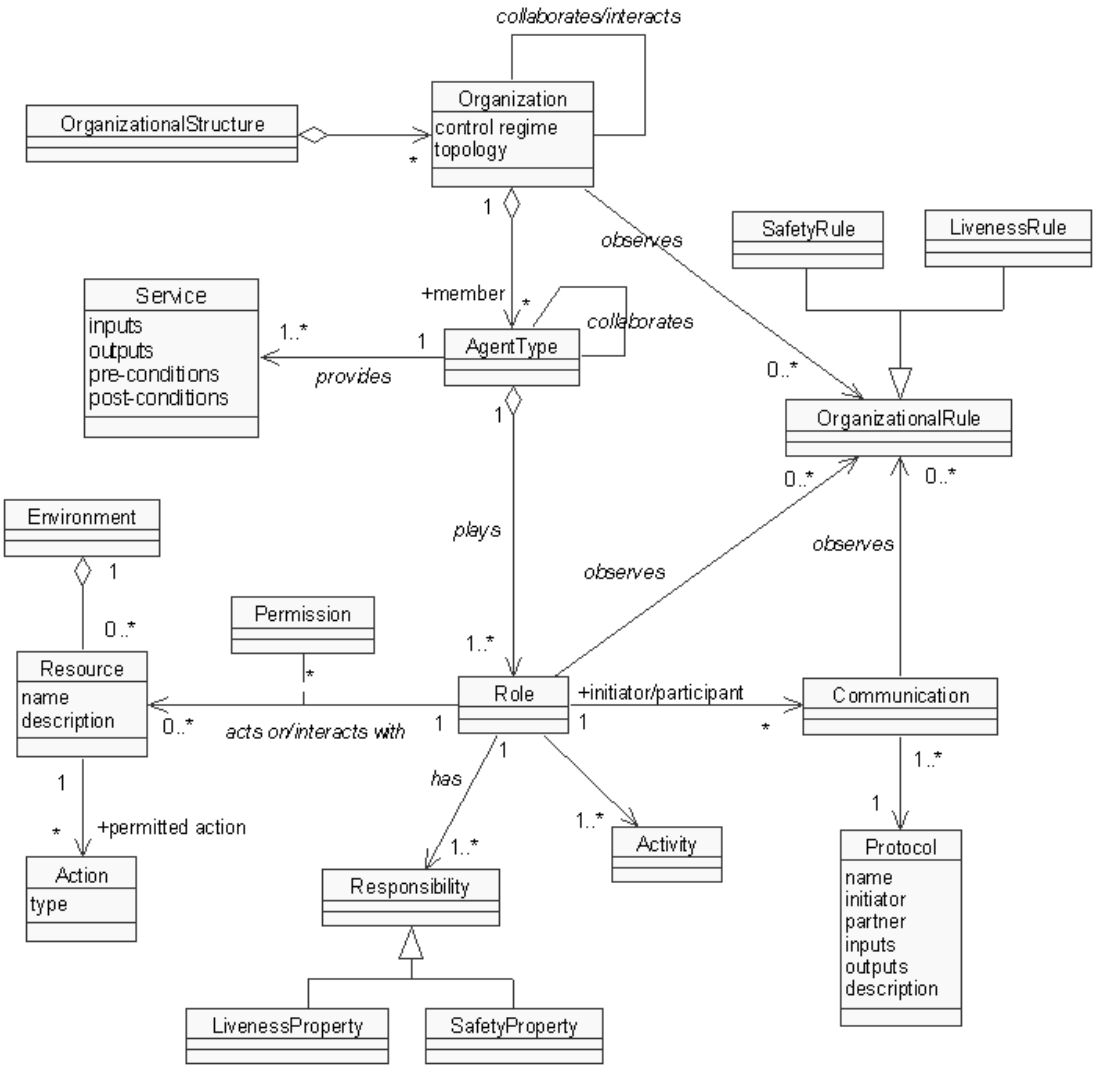


Figure. 32. The MAS meta-model adopted in Gaia (Bernon etal,2004)

The environmental resources that can be employed through agents can be seen as an indicator of the problem scope and, to a certain degree, as the clear image of the environmental recourses. Organizational rules have certain restrictions that the society has to perceive when planned through such instructions so as to address the conduct of the community as a whole or when regarding only exact roles or procedures.

The system is fully constructed through society configurations as the state of affairs of every role in the institution and its association with additional roles.

These organizational guidelines are the basis for assisting creators in the identification of the configurations that more logically outfit these instructions, bearing in mind that such organizational guidelines and configurations are firmly connected together. The organizational configuration is not indirectly distinct through the role model, in the extend version of Gaia; however, as an alternative, the determination of the roles is a result of an investigation into selected organizational configurations. In the creation stage, when precise documentation of the organizational structure occurs, the role model and the connected interface model will be totally distinct (Bernon et al., 2004).

### 4.2.2 Tropos Meta-Model

The models in Tropos are persistently developed as examples of a theoretical meta-model based on the subsequent notions/relations (see Figure 33). An actor indicates a physical and social agent in addition to a role or situation, which prototypes the unit that has tactical objectives and is located intentionality in the interior of the scheme or the organizational background. The properties needed to include when devising software - for example, autonomy, social aptitude, reactivity, and proactivity - are inherent in the AI meaning of software agents (Nwana, 1996).

Tropos defines a role as a depiction of the conduct of a social actor inside some dedicated surroundings. Here, a location signifies a group of roles, usually performed by one agent, which can subjugate a place, though a location itself is thought to shield a role.

**Goal**, it appears as the actors' strategic intent. We differentiate durable objectives from soft-goals with no clear-cut description or norms and devise them with the presumption as if they will be fulfilled or not. This process is based on the notion that objectives are fulfilled while soft-goals are contented (Chung et al., 2000) as soft-goals are characteristically utilized to model non-functional requirements.

Figure. 33. The MAS meta-model adopted in Tropos (Bresciani et l.,2004)

**Plan.** It indicates, at an intelligent level, a technique of executing something. The application of a plan can be a way to reach an objective or a soft-goal.

**Resource:** It embodies a physical or data item.

**Dependency**: In between two actors, it designates that one actor relies, for some specific reason, on the other so as to attain some objectives, fulfill some plans, or

offer a resource. The former actor is named “the depender” and the latter is “the dependee”.

**Capability:** This is the ability of an actor to describe, select and perform a plan so as to achieve objectives under certain circumstances and within a certain case.

**Belief:** This represents the actor’s knowledge of the world.

### 4.2.3 PASSI Meta-Model

Founded by Bernon et al. (2004), the PASSI meta-model (Figure 34) depicts the troubles in terms of scope as concerned with the handler’s problem in terms of situations, necessities, ontology and capital. In this description, situations define a group of connections amongst actors and the scheme; necessities are characterized with conservative UCs model case. The common idea is that many qualified experts existed in various corporations who can easily adapt themselves to using AO methods if familiar with the main notions. An ontological explanation of the scope consists of notions (that is, classifications of the scope), actions (accomplished in the scope ) and establishments (declaring something concerning a portion of the scope).

This signifies the scope in a manner which is considerably better-off than the characteristic fundamental symbols fashioned in the object-oriented examination stage. The latter components of the problem domain are resources as they can be retrieved or employed by agents. A resource can be an exporter of information, such as a relational database. The components of the agent-based resolution are exhibiting the agency domain.

The actual center of this part of the model is the concept of the agent, and all agents in this approach are accountable for comprehending several tasks sloped from one or more requirements. One of the tactical conclusions reached when considering PASSI is the straight connection between an obligation and the accountable agent.

Occasionally an agent has also access to obtainable resources. Every single agent, throughout its lifetime, takes certain roles considered as the lot of that agent’s common conduct categorized through some privacy, for example an objective or offering a tasks or a facility.

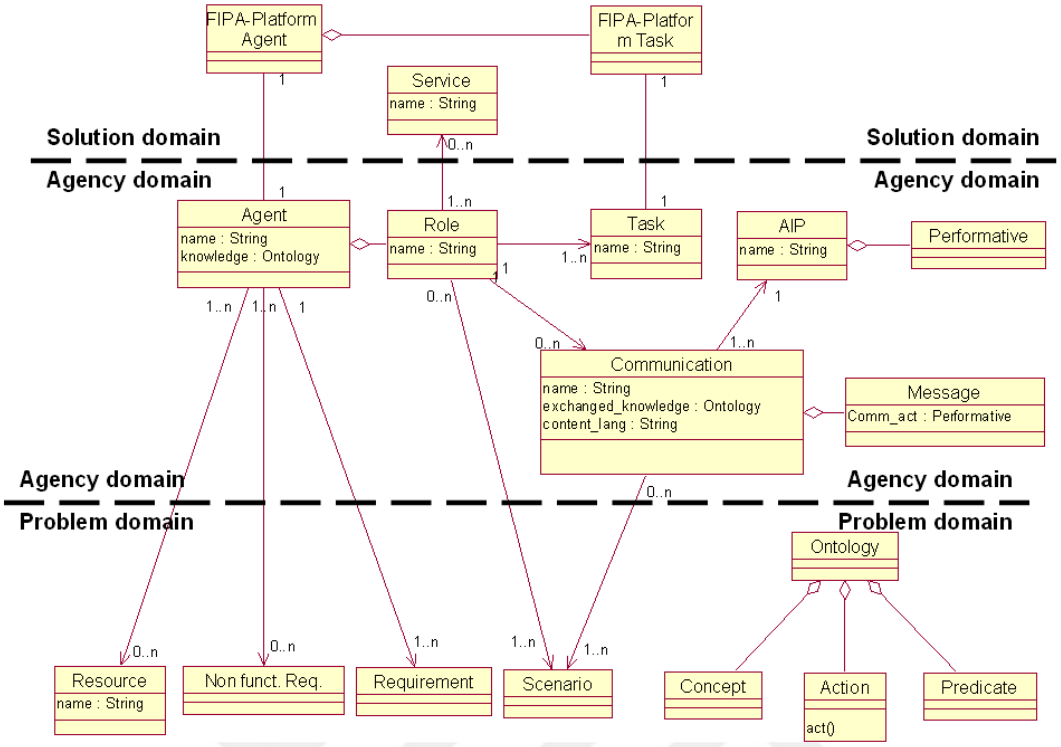


Figure 34. The MAS meta-model adopted in PASSI (Bernon etal,2004)

This means that roles could use communications with the intention of understanding their relations or portions of conduct (called tasks) to activate the agents relations. A programming language that is completely clear to agents is the means of expression of a group of messages for communication. Each connection clearly denotes to a portion of the ontology (which means that the information swapped includes notions, establishes or events definite in the ontology) and its movement of messages is managed by communication protocols.

### 4.2.4 O-MaSE Meta-Mode

The major ideas and relations employed to clarify multi-agent schemes are defined by the O-MaSE meta-model, which is configured based on a structural method and encompasses designs that authorize for ranked and team-founded decomposition of institutions. The O-MaSE Meta model is based on the OMACS and embodies the data compulsory for a system's institutional configuration and capabilities so as to

authorize it to start and be re-arranged at the allocated execution time (DeLoach & Miller, 2014). An organization consists of seven elements: objectives, functions, agents, organizational, a scope model, and policies, as showed in Figure 35.

However, different from light definitions as in the artificial intelligence and agent groups, O-MaSE defines an object as an impartial part of the organization and mostly developed in terms of some expected case of the world. The role describes an organization whose conduct is predictable to reach a specific objective. Specialists are set to play a role and accomplish the predictable conduct of those roles. Agents, which can observe and act upon their surroundings, are independent beings (Russell & Norvig, 2002).

Learning the soft goals or the hard goals is possible, and an agent has all the important skills to take on a role. Capacities can be identified as (a) a group of sub-abilities, (b) a group of events that may react with the environment or (c) a plan that uses events in a special manner. Regulatory agents are departments that operate as agents at a top level, thus understanding the concept of organizational hierarchy.

The scope model is applied to capture the crucial components of the surroundings in which agents work. The domain model is to identify organizational strategies and to determine how a system may act in a specific state. Policies are frequently utilized to determine the safety and privacy characteristics of the planned project, while protocols are described as the relationship between the roles or between the organization and other outer actors. The concepts of the algorithm plan are applied by agents in respect of activities with the environment and messages in protocols specific strategies.

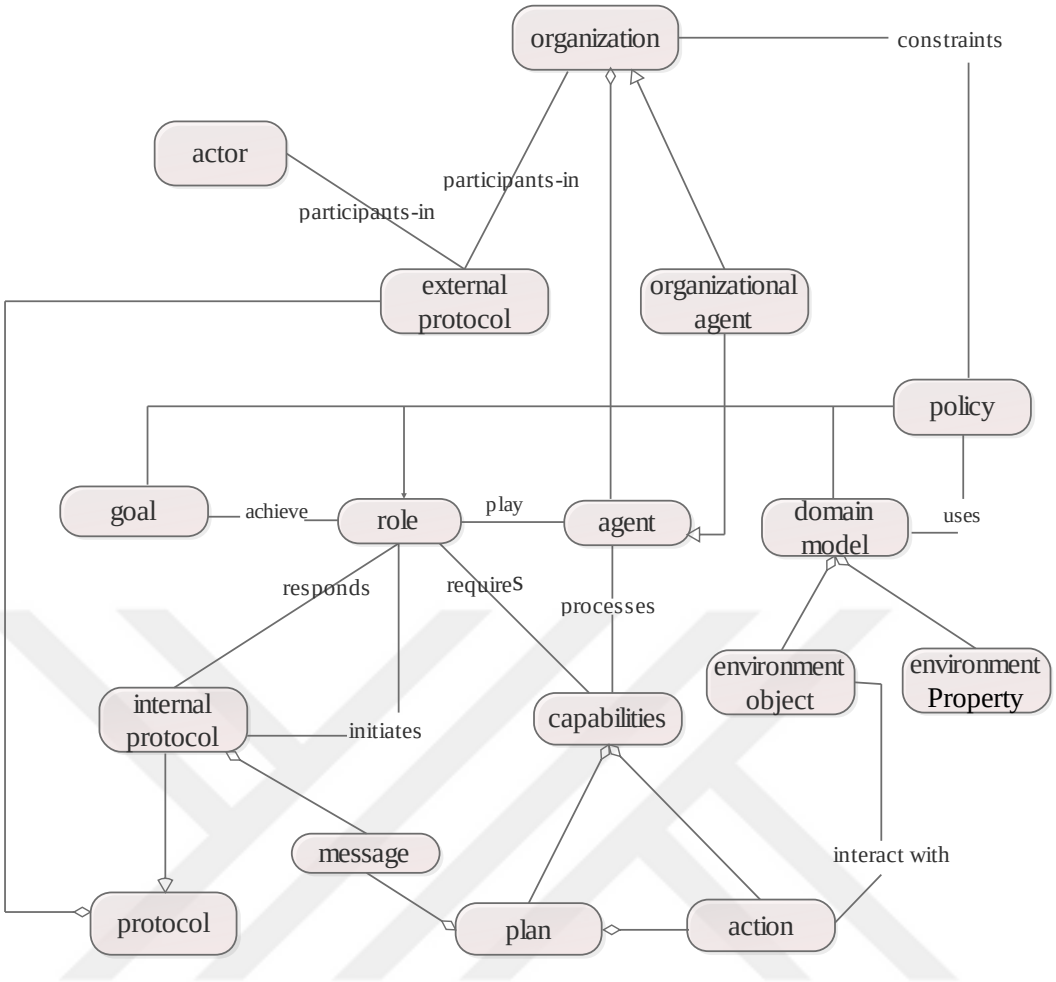


Figure 35. The MAS meta-model adopted in O-MaSE (DeLoach & Miller, 2014)

### 4.2.5 ADELFE Meta-Model

ADELFE is a practice dedicated to software manufacturing of adaptive MASs (Bernon et al, 2002). In circumstances in which the setting is changeable or the system is exposed, adaptive software is exploited. To fix these problems, ADELFE-confirmations help set up the software based on the AMAS model (Capera et al., 2003). According to Bernon et al. (2004), the meta-model for ADELFE (Figure 36) is mainly clarified through this feature within the approach and by the characteristics a cooperating agent owns. Its lifecycle is a traditional cycle and requires devising concepts, making resolutions, and then carrying out activities. Furthermore, local collaboration instructions allow it to sense and resolve NCSs.

These NCSs are cooperating fail which are unpredictable with its cooperative social attitude, from its point of view. Depending on the settings in a related project, a variety of such failures can be sensed; these are, namely: incomprehension (when an agent does not realize a recognized signal), vagueness (when they have many inconsistent interpretations of the recognized signal), ineffectiveness (once it cannot satisfy the request of another one), unfruitfulness (when it receives information that is already known and leads to no reasoning for it ), concurrency (when many agents need the arrival of an exclusive supplier), conflict (once the agents need to carry out the same actions) and, finally, ineffectiveness (in case the agent may do an unhelpful act, according to its beliefs, to other agents).

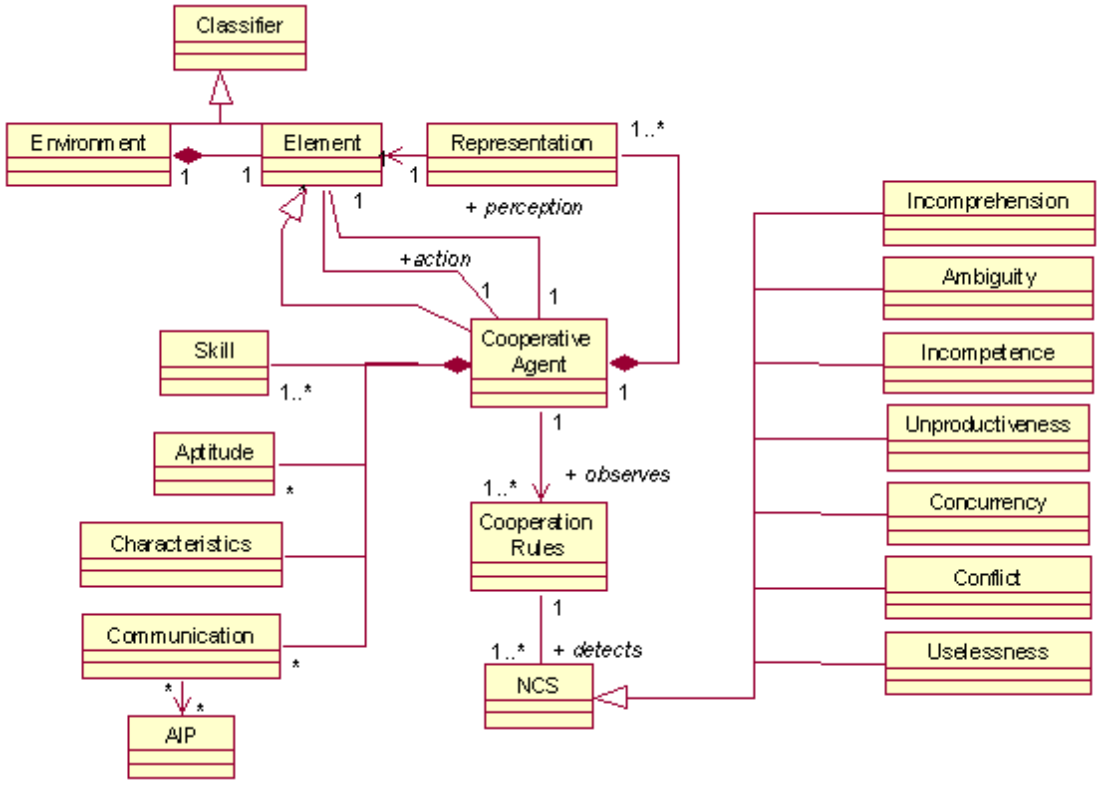


Figure 36. The MAS meta-model adopted in ADELFE (Bernon etal,2004)

An agent owns realm symbols that are opinions regarding another agent, the physical setting or the agent itself. To define its conduct, the agent uses these symbols. A representation can be shared by various agents. An agent is capable of interacting with

other agents or its surroundings. This connection can be completed in a straight way via message exchanges or additional connections. AIPs may also be utilized in order to define the interaction configurations among the agents. By ways of perceptions and actions, an agent can act with its surroundings. In this respect, aptitudes display the skill of the agent to reason both about the data and the philosophies it possesses. For example, through an inference engine at the base of the instructions or any other handling on conceptions and world exemplifications, an aptitude of a software agent can be expressed. The agent has certain capabilities, which are particular information that allow it to assess its own tasks.

### 4.2.6 ASME Meta-Model

ASME, the practice using Agent Modeling Language (AMOLA) was first introduced by Spanoudakis & Moraitis in 2008. ASME utilizes the Actor and goal notions. Here, the target refers to an exclusive source of zero or more, the actor directs his objectives using a reference to its goal. The reader should observe an option to add the demands EAttribute of goal. Each goal is connected to practical and non-practical demands, by this feature, which are recognized in plain text shape (see Figure 37(a)). In Figure 37(b), the UC notion has been well-defined is encompassed in other UCs notions. It interrelates with one or more roles that can be human roles or agent roles (system role). In Figure 38(a), the AMOLA meta-model describes the notion role that references the concepts:

– Activity, which indicates a plain process with two features, name (its name) and functionality (describes what the activity does)

– Capability, which refers to sets of events (to which it refers) attaining an extraordinary level goal, and,

– Protocol. The name of the properties of the protocol and the contributor refer to the relevant elements of the AIP model.

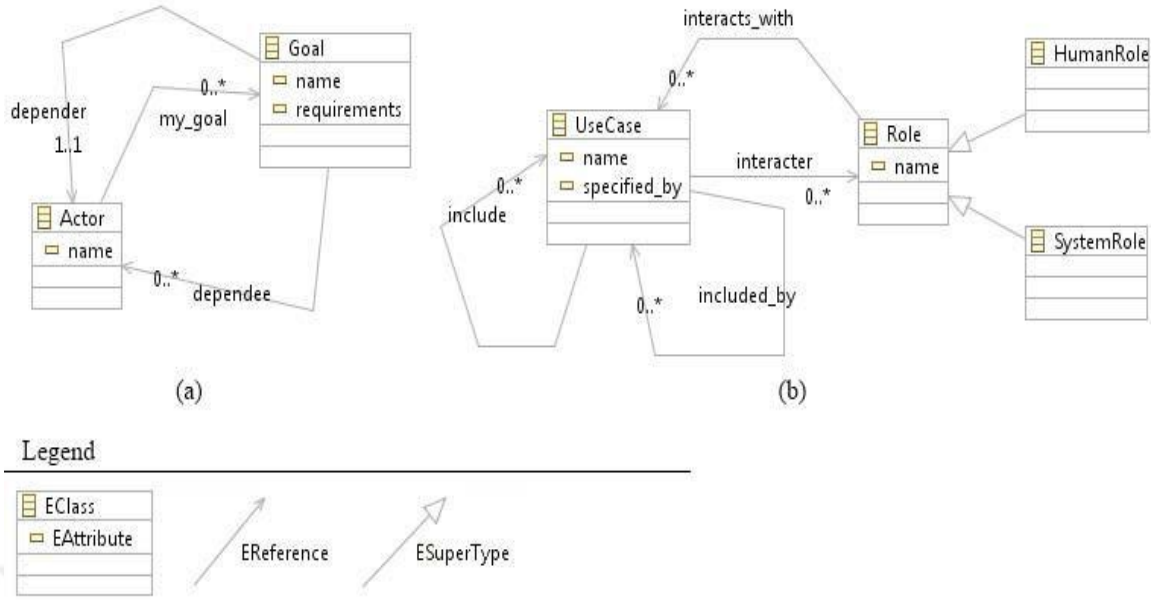


Figure 37. The MAS meta-model adopted in ASME (part-1) (Spanoudakis & Moraitis, 2011)

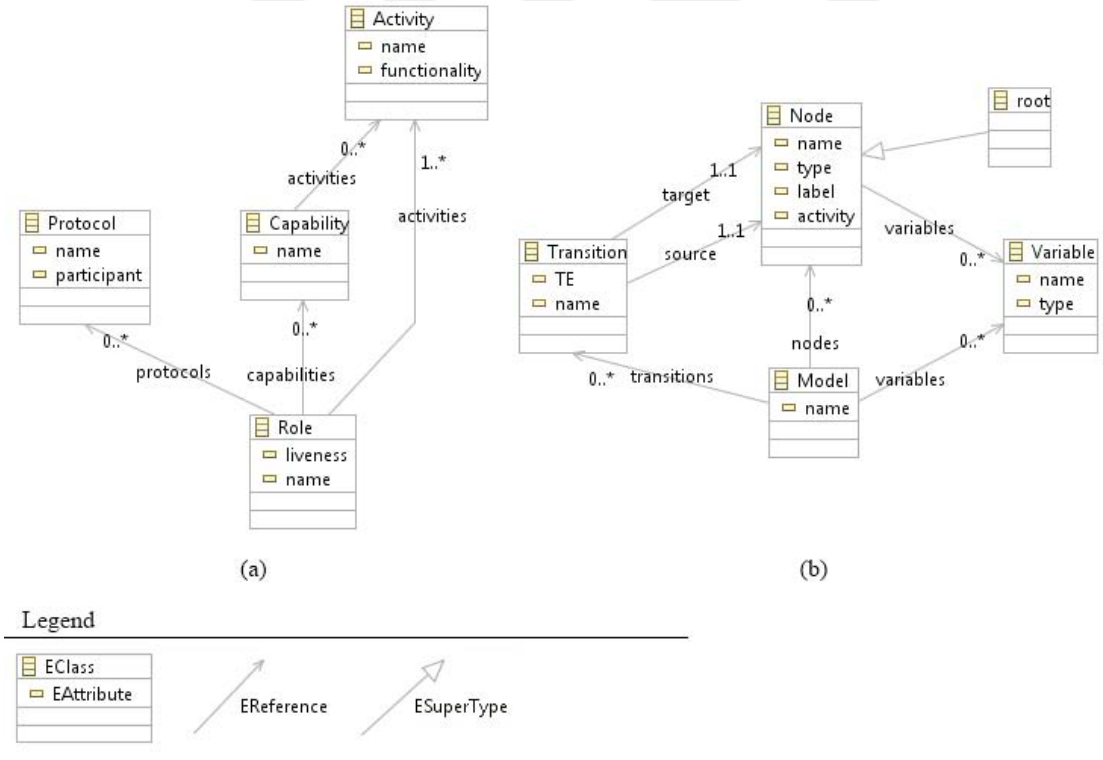


Figure 38. The MAS meta-model adopted in ASME(part-2) (Spanoudakis & Moraitis, 2011)

Yet, this model is not comprehensive and is used to classify the roles that contribute to the protocol, its commitments within the protocol and the rules for engagement. Figure 38(b) describes a model notion which has nodes, transitions and variables as EReferences.

### 4.2.7 Prometheus Meta-Model

A range of objects were produced by the Prometheus policy in the software evolution processes. These objects form a linguistically reliable concept of an agent project to be constructed. Every single one of the objects indicates a diverse feature or concept level of the fundamental system and considers a "viewpoint" on the whole fundamental model. Each viewpoint portrays numerous relations among the Prometheus notions or beings, such as actors, objectives, roles, notions, conduct, actions, etc. Figure 39 represent the construction of the fundamental model and all the stated connections between entities.

The entities labeled in the system specification phase are displayed in Figure 40. As a result, two kinds of objectives exist: abstract and concrete. As for concrete objectives, they do not have sub-objectives, whereas abstract objectives can have sub-goals.

A condition involves a structure of phases connected with compatible elements such as goal stages and objectives, action phases and actions, etc. An agent involves many competences or tactics and each competence comprises some tactics and/or sub-capabilities. It is noteworthy to state that if an agent has a competence, and that competence has several policies, then that agent is thought to also devise these policies and the association among agents and policies is regarded as transitive.

Sending and receiving communications and accomplishing some activities is an essential part of the plan, which may comprise repossessing information to deal with a percept or to achieve some objectives.

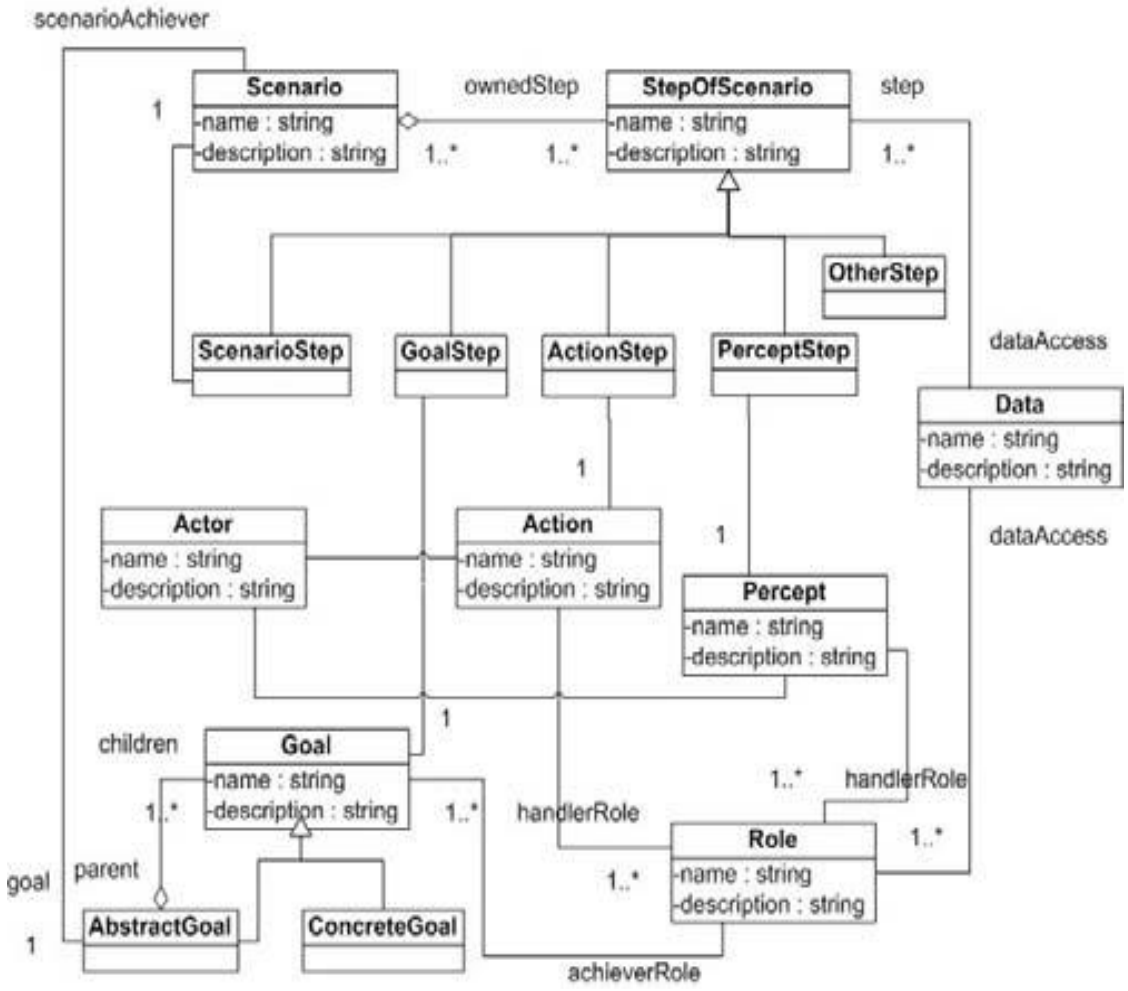


Figure 39. The MAS meta-model adopted in Prometheus (part-1) (Dam et al., 2006)

Notions, events, messages, and information are the connections that agents have with goals. Internal messages which are dispatched inside an agent to activate other plans, and exterior messages which are swapped among agents are the two kinds of communications in Prometheus. External message configurations are defined by the communication protocols (Dam et al., 2006).

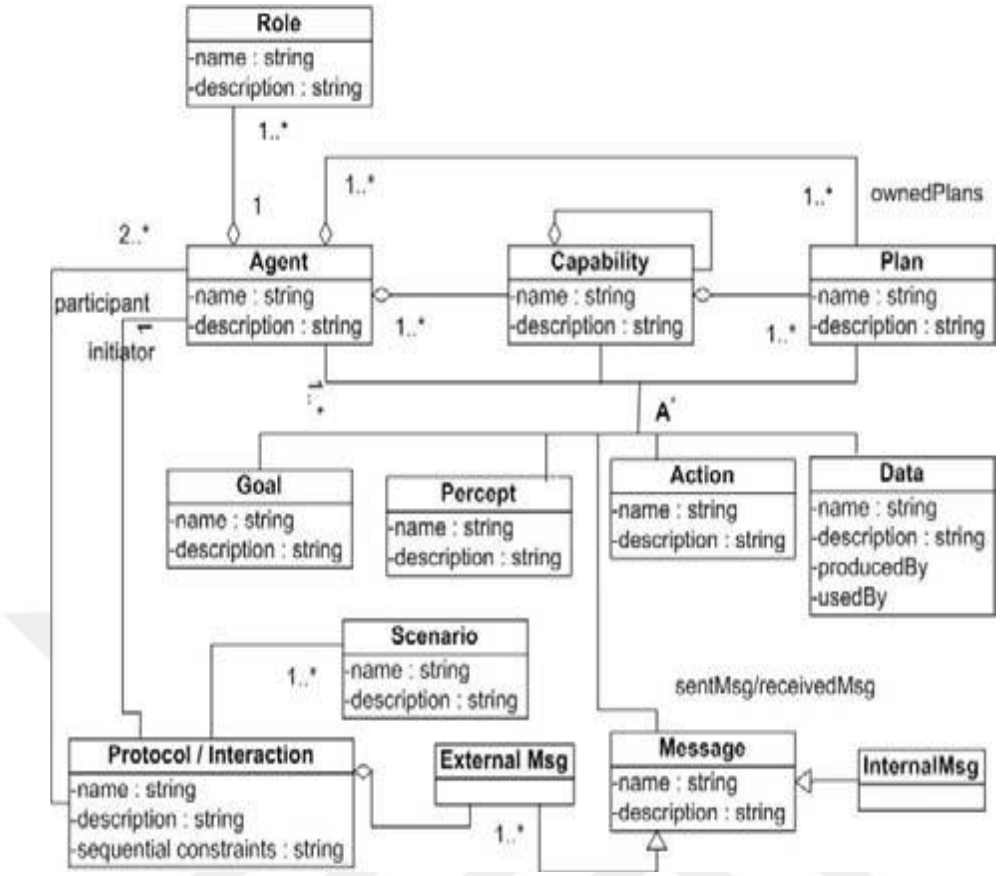


Figure 40. The MAS meta-model adopted in Prometheus (part-2) (Dam et al., 2006)

# CHAPTER 5

# UNIFIDED AO SOFTWARE ENGINEERING METHODOLGY

In the preceding chapters, a detailed description was provided of the techniques for experimental evaluations. In this chapter, we will introduce the results obtained from as the outcome of those assessments, and present a number of proposals as regards a unified agent methodology.

Within this chapter, the similarities and differences of the seven approaches which are specified as an outcome of the structural analysis are described in section 5.1. Section 5.2 presents the results obtained when we applied the methodology to the case study. In section 5.3, a list appears of the evaluation results regarding the meta-model technique and an attempt to unify the participant methodologies in one meta-model (Section 5.3).

In section 5.4 we will propose a unified model based on the seven agent approaches and according to the outcomes obtained in the assessment. Finally, a unified AO software engineering methodology was evaluating by utilizing the three valuation techniques.

## 5.1 Structural Analysis Outcomes

As mentioned previously, the methods used in the processes and models subject to this study are addressed so as to identify the common cores and components of the seven methodologies. To do so, structural analysis is performed on each process and model, thus allowing us to determine the similarities between these methodologies as well as the variations of each one. As shown in Table 3 the results of this analysis are presented as the following in terms of the commonalities:

- Initial requirements;
- Applying use-case requirement analysis;
- The concept of environment;
- Capturing goals;
- Social system structure; and
- Agent acquaintance model.

Table3.The summary of commonalities

| | **initial requirement** | **using use cases** | **capturing goal** | **social structure** | **acquaintance model** |
|---|---|---|---|---|---|
| **Gaia** | X | X | √ | X | √ |
| **Tropos** | √ | X | √ | √ | X |
| **PASSI** | X | √ | X | √ | X |
| **O-MaSE** | X | X | √ | √ | X |
| **ADELFE** | √ | √ | X | √ | X |
| **ASEME** | X | √ | √ | √ | X |
| **Prometheus** | X | √ | √ | √ | √ |

As shown in Table 4 in terms of differences, the results are as follows:

Table4.The summary of differences

| | **Model-driven** | **Document-ation** | **Deploym-ent model** | **Data coupling** |
|---|---|---|---|---|
| **Gaia** | X | X | X | X |
| **Tropos** | X | X | X | X |
| **PASSI** | X | X | √ | X |
| **O-MaSE** | X | X | X | X |
| **ADELFE** | X | X | X | X |
| **ASEME** | √ | √ | X | X |
| **Prometheus** | X | X | X | √ |

- Model-driven engineering (MDE) approach;
- Documentation of non-functional requirements;
- Deployment model; and
- Data coupling diagram.

## 5.2 Results of the Case Study

This method is the second evaluation method used in this work where the GA: PCHIS is applied to seven AO methodologies – namely, Gaia, Tropos, PASSI, O-MaSE, ADELFE and ASEME - to design an agent information systems. The system constructs software agents representing the hospital, family members at home, the patient being monitored, and healthcare providers. As stated before, the case study attempts to address the AO approach ability to resolve an actual and real-world problem and to determine, as an outcome, whether the approach is comprehensible and applicable. In the next sections our notes and comments are summarized on these methodologies.

### 5.2.1 Gaia Methodology

Gaia was the first methodology to be used in this research in its most recent form as in Wooldridge et al. (2000). It focuses on context agents. In all, Gaia's steps are easier to follow; yet, some are not clear through the example. Gaia analysis stage involves building a model and software protocols. Nonetheless, this approach lacks support to help analysts define special roles in the system. Apart from that, the protocol paradigm is descriptive and easy to learn.

Difficulties were spotted when working on protocol schemas and role models, with the underlying reason being that performing elaborated role models needs determining the liveness responsibilities of each role, decisive in plotting the protocols later. In addition, the authorizations and safely responsibilities mentioned in Gaia are not clear. As an outcome, in the paradigm of the role set up for PCHIS, this feature of roles is not well-described. Regarding the notation which characterizes these role's

characteristics, liveness responsibilities are rationally well-identified, while liability is not submitted.

In the analysis stage, there are some conclusions in distressed assistance to determine the roles of agent types, some obstacles are encountered when completing this step. As it appears, more specific and detailed guidance will be more useful; plus, safety permissions and responsibilities were identified during the process of building a service model for each role in a more clear and appropriate manner.

A limitation is observed in the service design used to determine the notations, mainly such that the pre- and post-conditions were absent. The final model of Gaia design agent acquaintance tends to be the simplest paradigm that can evolve. The paradigms depicted in the methodology are fairly well-described. However, some notations still need to be determined clearly. The analysis operation required a lot of duplication, resulting in issues related with model consistency, connections among models and tracking the changes made. These problems emerge in the form of shortages in backing for tools. It has to be considered that Gaia is a generic model and does not support design details.

### 5.2.2 Tropos Methodology

The goal-driven in Tropos which require engineering techniques established at this stage have helped to establish an approach of agent-oriented to analyze and design the target system. Indeed, the connotations described at this approach, such as actors, objectives, projects, etc., are identical to the concepts in agents. The early requirements analysis phase includes defining target system objectives, resources, plans and tasks to achieve these objectives. However, there are some problems at this stage. For example, in several situations it is difficult to differentiate among goal mean-end analysis and dismantling .

The Tropos structural stage is identical to the OO methodology. Determining the general architecture of the system, identifying possibilities, identifying types of agents and planning on capacity are the three steps of the process used in Tropos. Various uncertainties were faced when analyzing Tropos, such as whether to include

human agents' abilities in step 2. Also, in step 1 the same problems were faced, requiring an extended actor diagram provided by the designer to determine which actor should be focused on, the complicated one, the simple one, or both.
While carrying out step 3, we also noted the inadequacy of the descriptions contained in several papers on Tropos, thus facing issues in grouping the abilities into agent types.

The detailed phase of the Tropos design includes the construction of three types of graphs: ability model, plan diagram and interaction diagram. In Tropos there are hardly any detailed and sufficient descriptions related to this stage which is due to some ambiguity when presenting design elements such as agent and protocol interaction illustrations. Because of the different examples that show different types of charts throughout papers related to Tropos, certain constraints were encountered when drawing ability schemes**.**

### 5.2.3 PASSI Methodology

In PASSI, we applied the domain description stage to construct a practical depicting of the system utilizing classical use-case models from the system requirements model (see Figure 15). Furthermore, the agent identification phase assigns responsibilities to the agents, showing them as UML standard packages (see Figure 16) and determines the abilities of each agent with an activity model in the task specification stage. To show the already presented scenario, we also use the role identification model, which is applicable when patient information is processed out of the role data processor of the PDA agent to the role data loader of the home computer agent (as depicted in Figure 17).

The syntax is slightly different despite the similarities between the diagram and the UML sequence diagram. We can consider the results of the analysis and the design process as abstract specifications that need to be further developed with additional design methods. However, PASSI produces executable codes for a standard and increasingly used concrete structure such as FIPA. The community of agents taking into account its existential view is characterized by the PASSI ontology description

stage. A detailed description of the interactions is provided by automata, the complementary which represents the situation of the joint agents in the communication.

Together, what defines the protocol are the interlocking aspects of the dialogues, which include the definition of protocols in the interaction model. Each model presents its possess group of symbols and notions, thus driving to a complicated in side of vocabulary. As a final note, PASSI does not take into account the environment.

### 5.2.4 O-MaSE Methodology

In the O-MaSE analysis stage, we found the process steps clear and plain to follow. In the same way, the notation utilized in the methodology is obvious and easy to comprehend. The offered tool was very helpful in constructing analysis and design diagrams and to examine the cohesion between them. Also, every agent works as a software component that interacts with another agent or software component to accomplish a combined comprehensive goal. Even though some agents are smart and designed effectively while others are not, they all act in the same way to fulfill the required goal.

O-MaSE can also be regarded as an environmental paradigm. We applied the goal model covering the entire goals as classified into sub-goals as Figure 18; while Figure 20 represents the organization and their relations with external actors. Of the design processes presented in O-MaSE, we found the policy model too complicated to follow. Apart from these, the other steps are well-described and easy to effect.

### 5.2.5 ADELFE Methodology

The designer is directed by ADELFE to make a decision regarding the AMAS theory if needed in the system under development. This demonstrates the significance of verifying the extent of analytical work and the appropriate tool that analyzes the criteria provided by the developer to determine the usefulness of this concept. The

developer can utilize other AO methods if the project is not suitable for AMAS theory. We note that ADELFE allows for different UML/AUML schemes. UML notifications are used in this application; also, the reuse of the UML-oriented extension as well as specific steps for adaptive system design are added based on AUML formats for typical interactions between agents. It extends UML with knowledge level concepts and provides graphs to show them.

This is also the case in final stages and in the identification of cases of use to clarify the various functions that the system must respond to. The main objective is the state of use as a result of a set of functional requirements, and the use of a specific sequence to separate each case (see Figure 23). We have also utilized a class scheme to show the main categories of the GA: PCHIS problem as shown in Figure 24.

### 5.2.6 ASEME Methodology

In ASEME, the data to be added at each stage is obvious and the models applied are popular in the software engineering society, implying that any designer can adjust to the ASEME activities. Form conversions are automatically made during the software development process. In the previous chapter, we presented models of transformations occurring at various stages of ASEME, which clearly supports the transition between these models. As shown in Figure 28, in the analysis stage actors are specified and the UCs are linked to our target project. These documented outcomes employ UCs diagram in ASEME.

To build MAS, ASEME methodology uses model-driven techniques. Utilizing EMF together with meta-models, the various models included are represented. At each stage of development, the same process occurs with tasks to be completed (Jorge et al., 2015). ASEME uses the AMOLA, which describes both syntax and semantics for building models of MASs covering the analysis and design stages of a software development process. AMOLA handles the individual and societal side of the agents, describing how protocols and capabilities can be used in agents' designs. The guidelines are missing because the authors depend only on automated paradigm conversions.

### 5.2.7 Prometheus Methodology

The last methodology used in this study is Prometheus. There must be a process with specific results where the aim is to develop Prometheus which can be learned by university pupils who have no background in the agents and can be used to advance in smart agent systems . To aid designers in grouping demands a detailed designs, which Prometheus can provide with examples and heuristics. The system goal diagram in Prometheus in the system specification step is used to define goals and systems in a functional way (see Figure 29); indeed, the function of the system is determined by setting goals, identifying the functions that fulfill these objectives, and through determining UCSs.

The processes of realizing the objectives of the system starts by capturing a preliminary group of high-order objectives and by following each of these goals and asking how to achieve this goal as these primary goals are developed to later become a set of full goals. System agents and events sent by these agent are presented within the system overview chart. We can deduce this graph immediately from the model UC and connection model. The agent overview chart also illustrates the agent's inner workings as interactive abilities and beliefs by passing the event. Figure 31 shows another example.

The capacity graph illustrates the internal work of capacity as an interactive plan by passing the event. One characteristic of this approach is the numeral of places that automated tools can be utilized to verify consistency via different diagrams in the design steps. For instance, in the system and agent overview chart, the agent input events must be the same as the output events. We also follow how to use certain design elements like protocol definitions and capacity schemas to provide support, debugging, and tracking within the port system in terms of integrated methodology.

## 5.3 A Comparison of MAS Meta-Models

At this stage, we will compare all the hypothetical meta-models to identify their commonalities and study their distinctive differences as an initial move to an agreed combined and unified MAS meta-model. The seven agent methodologies have different backgrounds and focuses; each of them deals with particular sorts of agents

or MASs. These variation are mirrored through the meta-models constructed here to express the notions utilized in the design processes and the systems concerning these approaches. As a first step, the results of this comparison are summarized by discussing the MAS meta-models from the architectural viewpoint. Table 5 presents these results.

Table 5. Comparison of the discussed MAS meta-models

| Meta-Model | Common concepts | Peculiarities |
|---|---|---|
| **Gaia** | Agent type, Organization, Organization rule, Role, Communication, Responsibilities, Protocol, Environment, Resources, Service. | Interactions |
| **Tropos** | Agent, Actor, Goal, Plan ,Recourse, Dependency, Capability, Belief | And /OR decomposition, Mean-Ends analysis |
| **PASSI** | Agent, FIPA-platform, Role, Service, Task, Communication, AIP, Recourses, Action | Concept, Ontology, Predicate |
| **O-MaSE** | Organization ,Agent, Role, Goal, Capability, Domain Model, Policy | Domain model, Policy |
| **ADELFE** | Representation, Environment, Communication, AIP | Cooperative agent, Cooperation Rules, Non- cooperation situations, Element |
| **ASEME** | Actor, Goal, Role, Protocol, Capability, Activity | Transition, Node, Model |
| **Prometheus** | Actor, Goal, Role, Organization, Agent Capability | Domain model, External model, Internal protocol |

### 5.3.1 Towards a Unified MAS Meta-Model

As mentioned previously, among the objectives of comparing the seven AO approaches is to identify a unified MAS meta-model. We believe that each of them has several substantial characteristics, but that these features are fundamentally situated in different contexts. The seven meta-models shown in the previous sectionbs are various and are a good instance of the discussion in the agent society about these important matters.

To capture the core of each one, we have to think within the particular method pursued by the authors and the architecture of the system addressed. Because Gaia meta-model as a MAS system appears as a socialite organization, the roles more than agents are the main subject of the model and as a central constructing agents. Whilst a Gaia role is featured by an effective architecture and indoor duties, an organization is featured by a group of roles interactive with each other in accordance to particular protocols and by organizational duties which refer to the bases that an organization abides to. (see Figure 32).

As for Tropos, it is established in two major steps: First, the concept of the agent is used alongside all the objectives and plans associated with the various concepts of software development, from preliminary analysis to execution. Second, an analysis is made of the environment in which the software should work. In particular, Tropos focuses on constructing a paradigm of the target system and its surroundings, and offers a way for developers to design a MAS that can incorporate certain features of the societal model into all activities of the design process (Figure 33).

The PASSI meta-model is intent on reconciling classical software engineering notions, such as problem and solving, with the possibility of the agent-founded method whilst striving to fulfil the objectives in accordance to the needs related to code execution. Authors obviously refer to a FIPA-founded application of their projects and, thus, connections and execution matters are quite prevalent of those characterizations and are most popular on FIPA and JADE platforms . The concourse among agents and conventional matters of software engineering is acquired by inserting a novel abstraction level (agency scope) that supplements the recognized problem-solving scope (see Figure 34).

O-MaSE supports agent-centric, enterprise-centric, and open systems founded on parts of the method utilized in a suitable manner. In addition, each part of the O-MaSE approach is known via a mutual meta-model that also directly backs complicated adaptive systems. It defines parts of the method in grouping O-MaSE compliant methods with instructions for building the O-MaSE method. In addition, the agent tool is fully supported by O-MaSE, which supports the implementation and

creation of an O-MaSE-compliant method as well as system building by employing other particular methods (see Figure 35).

The purpose of resolving the issue with an adaptive MAS is obviously represented in an ADELFE meta-model, thus requiring a great deal of effort to understand, via collaboration bases, the cases that could cement or prevent the collaboration between agents. The agents' knowledge and conduct is addressed in the form of abilities, skills, and characteristics. Furthermore, agents connect through immediate communication or the environment in whihch they operate (Figure 36).

ASEME covers the notion of functionality to define the reasoning features of an agent. Then, it defines the notion of capacity as the capability to fulfill certain functions that demand utilization of one or more tasks. The agent is an element with specific abilities, also fit for inter and intra-agent connections (see Figure 37).

By providing views of various sides and at various development stages, Prometheus presents a multi-purpose evolution operation. The meta-model is not sufficient to officially characterize all conditions and interactions among Prometheus elements. It is hard for the meta-model to express the conditions that the plan should be triggered by the agent that is accountable for the task of operation. UML diagrams restrict expressiveness, which is also a popular issue for OO development utilizing UML (see Figure 39).

Based on the above analysis of different MAS meta-models, the common components will be identified to contribute in the construction and unification of the seven methodologies of MAS meta-models in one MAS meta-model. Figure 41 describes the the proposal of unified MAS meta-model; it includes a set of concepts as part of the proposed unified AOSE methodology. According to this Figure, the meta-model is directed by the objective of predefined organizations, is set in a particular structure, and puts constraints to systemize the work among agents, which plays different roles in the system; An agent exists in an environment for which it is capable to construct representations if it has cognitive abilities.

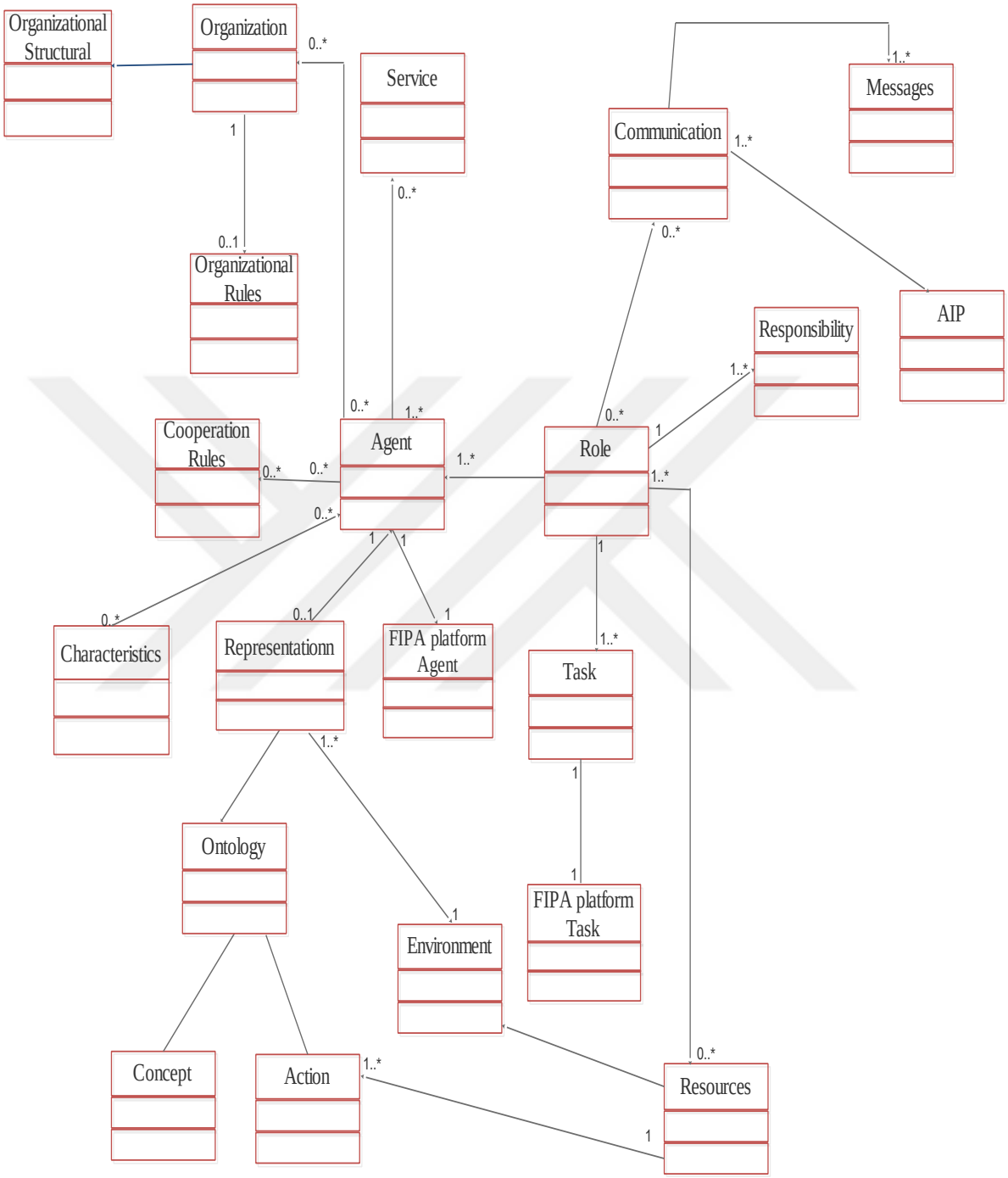


Figure 41. Proposal of unified MAS meta-model

### 5.4 Towards a Unified AO Software Engineering Methodology

As the last step in this research, we work towards the unification of selected agent methodologies. In the preceding chapters, 3 and 4, we applied the assessment techniques (structural analysis, case study and meta-models) of the seven competing AO methodologies, yielding in-depth comprehension of all these approaches as an outcome of their comparative analysis using techniques. We assessed their advantages and disadvantages according to a case study. In addition, their meta-models, their resemblance and variations were also investigated as to processes and models.

Following this analysis, we take the first step to combine their features with the purpose to construct a core approach and combine characteristics selected from various approaches. In this respect, some preliminary suggestions are introduced for the design of a relatively complete MAS methodology based on the selected methodologies. The purpose of this step is to contribute to the combination of the best features selected for agent development methodologies. Figure 42 describes the unified approach processes.

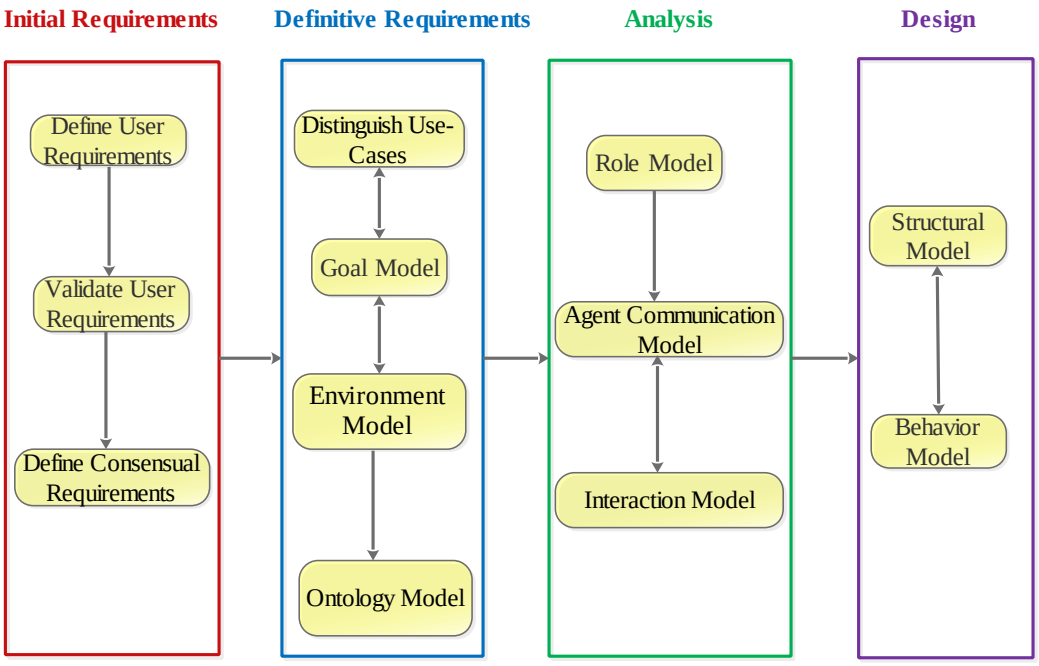


Figure. 42 A Unified AOSE methodology

As shown in Figure 43, with some exceptions, we will utilize the common and essential notations as presented in (Padgham et al. 2008).

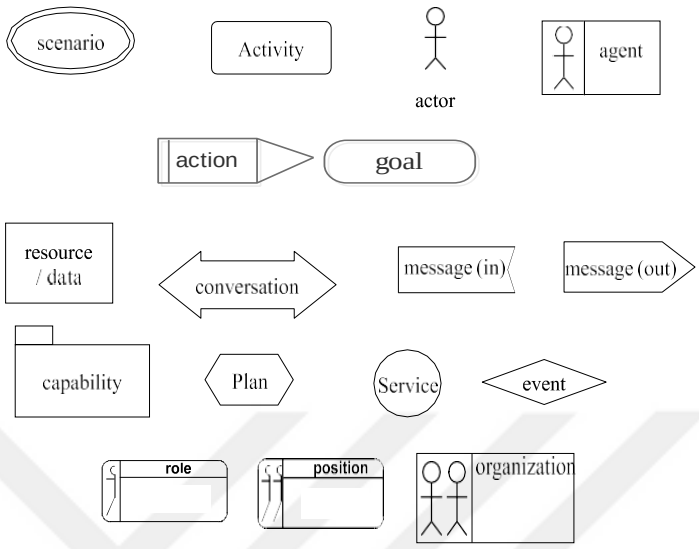


Figure. 43 A unified notation set (Padgham et al. 2008)

### 5.4.1 Requirements Specification

The first stage proposed is requirements specification. This stage is significant in software development processes, where it enables the system analysts to obtain the requirements of the target system. From our perspective, the success of any system development depends on the in-depth study of the system requirements.

#### 5.4.1.1 Initial requirements

This stage provides the initial step of the requirements analysis towards identifying basic stakeholders and to equip designers with useful knowledge about the environment in which software operates and the type of interactions to take place among system agents. As discussed before, the early requirements phase of the Tropos includes a study of the organizational preparation, including stakeholders, their objectives and relations. In ADELFE the purpose of this step is to realize a convention on the initial demands related with the characterization of the system and

the surroundings in which the system will be implemented. It is created to determine what the most suitable system can be for end-users. The unified approach should have an initial requirements stage, which can be adopted from ADELFE.

#### 5.4.1.2 Definitive requirements

According to this stage, in the ADELFE approach adapted in our proposed methodology, the aim of the definitive requirements is to convert this view to a UC diagram. To arrange the requirements (functional or not) in this step, the developer has to determine the task of the comprehensive system and to model its environment. The following steps, in our view, introduce a strong support for gathering requirements.

##### 5.4.1.2.1 Distinguish Use-Cases

This technique is used to characterize the different system functional requirements. It has been demonstrated to be an efficient model in OO techniques in grouping system requirements. Also, it assists the developers in deciding on the main interactions among system entities. Of the selected methodologies, ADELFE, PASSI, ASEM and Prometheus have applied use-case models. Particularly, Prometheus offers UCs, while PASSI proposes the use of UML-like UC diagrams, ADELFE utilizes this technique to clarify the linked sequence diagrams and to determine collaboration failures. In these use-cases, only energetic components are implicit and appear as the products of an effective requirements group (Bernon et al., 2002).

Exploring situation collaboration failure in the system and within its conditions is carried out so as to help developers in identifying problems and non-cooperative items and incidents. In ASEME, some shifts in semantics are displayed. First, the actor interacts with the system and supposes a role. Then, agents are developed as roles either inside the system boundary, i.e. for elements to be designed, or outside the system boundary, i.e. for agents in the environment (Spanoudakis & Moraitis, 2001).

Similar to conventional UML UC models, human actors are demonstrated as roles outgoing the system boundary. Later, the various UCs should be linked to at least one

agent role. The UC diagram in ASEME also adds three new ideas regarding the actor diagram and offers actors designed inside the system boundary. The diagram can include abstraction roles to ensure that agent IPs and goals are viewed from a developer standpoint by adding sub-goals related to implementation in the shape of UCs. To assist the analysts in defining the main connections/reactions among entities and adapt more with agent technology, the integrated methodology will combine the UCM on ADELFE and ASEME approaches.

#### 5.4.1.2.2 Goal Model

It can be said that the agents' objectives are one of the most critical model of agency which contribute to the strengthening of agents. Except for ADELFE and PASSI, most of the selected approaches concur on the significance of goal concepts and identify goals in the requirements analysis stage to be applied as a basis for identifying agents. Obtaining goals is one of the base processes in O-MaSE, and is a significant modeling activity in Tropos and ASME. Gaia expresses goals in form of roles' responsibilities in a way that is more realistic than goals in other approaches. Determining objectives is also a vital ingredient in Prometheus and presents such models as an essential part of the institution.

The unified approach should support capturing objectives by defining goals and their architecture and representation. It is as well substantial to address stakeholders' purposes and their relations through the functions and resources utilized to fulfill goals as carried out in the requirements analysis stage in ASEME. Goals can be organized and presented as a hierarchy of objectives as applied in O-MaSE (see Figure 45) in section 5.5.2, utilizing the suggested unified notation (Padgham et al., 2008).

#### 5.4.1.2.3 Environment Model

The notion of environment is basic for MAS since agents operate in an environment. To support complex open systems, agent methodologies need obvious diagrams to characterize the scope knowledge and the implementation environment. Complicated

open systems generally have extremely dynamic and heterogeneous environments. By officially defining the environment, a knowledge base is created that constantly deals with environmental changes. As a result, an agent system needs to have models to represent the environment in which it operates.

Despite the importance of developing an environment model, only O-MaSE, ADELFE, Prometheus manage to specify such models. In ADELFE, before identifying use cases and during the final requirements, the environment should be thoroughly planned by the developer. Afterward, one procedure is added to the RUP to describe the system environment. Specification starts by determining which elements interact with the system as well as the restrictions in these connections.

The model offered by Prometheus is a view of the environment in side of actors, notions and actions. The main components of the surrounding in which agents will work are shown by the domain diagram in O-MaSE. These components are described in the form of objects from the surrounding, which contains agents, and interactions among those objects. It can also be used to show the general characteristics of the environment to see how the objects connect.

However, most competing approaches do not take into account this important matter thoroughly enough. In fact, Gaia does not provide a comprehensive model of the implementation environment to the designers, and the environmental data is encoded in permissions and protocols for a specific roles. This omission makes Gaia unsuitable for engineering implementations with effective and diverse environments Tropos offers resources as an entity, but no more (Hoa & Winikoff, 2004). At this stage of our proposed approach, the environment and the concepts and interactions should be taken into account. we may adopt the Prometheus's analysis overview model and characterize the environment activity in ADELFE to address this concept.

#### 5.4.1.2.4 Ontology Model

The ontology model provides the notions utilized by the agent system. Of the competing approaches, PASSI and O-MaSE offer the ontology diagram. Instead of this model, ADELE adapts to the AMAS theory, which means that the agent is able to

handle its environment and the other agents. PASSI has a domain ontology description model which involves notions as classes, elements of the domain, predicates that emphasis on characteristics of notions and tasks that agents can do in the scope. PASSI also has a connection ontology model which offers connection tracks among agent kinds. O-MaSE's domain model also represents the main elements. These are demonstrated within the scope of different types of objects that agents operate with. It also displays the interactions among those object kinds and among them and the agents. Object kinds are realized by a name and a group of features and are, then, utilized in other approach steps.

Developing an ontology model has to be taken as a critical function in the activities of the proposed approach as the task includes specifying certain range notions, their properties and relations. We suggest that the unified methodology utilizes PASSI's domain and communication ontology models to describe the ontology of the system scope in best way due to its prominent advantage over the other methodologies in this regard.

### 5.4.2 Analysis Phase

The analysis stage intends to specify a comprehensive system's architecture and its conduct. These are obtained utilizing a role model as well as an interaction model. It seems to us that there are three important processes in the analysis stage: role model, agent communication, and interaction model. In respect of analysis activity, one critical stride is to determine the agents' roles. As debated in section 5.4.2.1, based on the information captured from the requirements steps, the system analysts specify a number of roles functionalities existing in the system. The relations and communications among agents are shown in sides of agent acquaintance schemes in section 5.4.2.2. Additionally, the system interactions are constructed according to UCSs and goal diagram (section 5.4.2.3).

#### 5.4.2.1 Role Model

Among the important demands of AOSE approaches is to help designers to distinguish the agents comprising the system; specially, agents are the main component in agent-based systems. A popular way applied in most selected

approaches coping with agent role is to begin from the smaller elements of the agents and, then, to group these elements to compose agents. In Prometheus, an agent kind is created by integrating one or more functionalities. Various sets of tasks present alternate designs that are assessed according to the coherence of the agent kinds and the range of connections among agents.

Tropos presents these elements as an ability. Gaia, O-MaSE and ASEME specify the agent as a role. Agents are specified by gathering UCs in PASSI. Tropos has extensive schemas which offer relations among actors, objectives, resources, and functions in the system. As mentioned earlier, the depiction of ADELFE does not offer adequately elaborate guidance to permit us to status whether it applied this technique or not.

#### 5.4.2.2 Agent Communication Model

In Prometheus and Gaia, the dependency and interaction among agents are illustrated in terms of agent acquaintance schemes. While Gaia realizes the connection links found among agents without determining the current properties of those links, Prometheus indicates the types of agent as well as the connections among them. Agent acquaintance diagrams in these two approaches are intended to help developers in determining potential bottlenecks formed among the system elements.

The data coupling offers an evident processes to cope with agent characterizations. One of them is the use of data coupling models. The second is the model of agent acquaintance. These models are composed of system functions and external resources in terms of specified data. An instance of this model appears in figure 30 of section 3.3.9. As noted earlier, using this technique, designers are able to assemble functionalities into agents by simply visually assessing the data coupling techniques and providing guidelines and processes.

This depends on both minimizing coupling and increasing cohesion. It appears that placing agents together that can read or write the same type of information reduces the association of agents. These intermediate models are adopted in the unified approach.

#### 5.4.2.3 Interaction Model

The importance of this step lies in describing the agents in the system and their functions, responsibilities and objectives. As mentioned in our structural analysis, ADELFE, PASSI, Prometheus, Tropos and O-MaSE share certain features as they present high-level connections utilizing sequence/interaction models extracted from UML sequence models. Each methodology has different interaction diagrams; for example, Tropos and ADELFE depict conversations among agents. This is while the sequence models in O-MaSE show reactions among roles and actors. PASSI uses serial graphics to explore the tasks of each agent through role-specific screenplays (Giacomo et al., 2007).

The role model in ASEME is basically derived from the Gaia methodology since, in both cases, the conversations are explored and modeled at the role-level instead of agent-level, and serial/interactional schemas are not used for the objective (Spanoudakis, 2011). Getting into more IP detail, ASEME, ADELFE, and Tropos suggest application of the AUML IP model, with certain tasks added to the AIP to suit AMAS's demands in ADELFE. The interactions of the system can be obtained at a high level utilizing AUML connection protocols for the proposed methodology**.**

### 5.4.3 Design Phase

This stage concentrates on determining agents' architecture and conduct via determining their elements and relations. It is quite necessary for AOSE approaches to offer a clear way to comprehend the general framework of the system. Not all current approaches present support for social structure and conduct (role, communication, and interactions). In O-MaSE the process designer needs to identify the specific set of stages and, then, identify the activities and tasks for each phase and re-use them again.

Since what has been said will be specific to every project under developed, there are no strict bases on processes to be developed at any stage. It characterizes the micro-level of dynamics employing finite state models. The design stage in ASEME encompasses the functional and behavioral sides of the MAS, and the related models are the agent IP and IAC that carry out a certain IP by assuming the crucial roles and

relations between them. In Tropos, every agent's plan is depicted utilizing a UML process model.

As a result, in order to rethink beliefs and plans in an adaptive manner, it is difficult to think of a changing environment. yet, it proposes several of potential notations that the developer could utilize to cover the agents dynamic conduct. The service model in Gaia offers a description of the inputs, outputs, preconditions, and post-conditions of each task offered by an agent. However, the models for showing agent's organizing abilities are not depicted. ADELFE directs the designer to decide whether AMAS theory is ideal in the project being developed.

This illustrates the significance of verifying the sufficiency of the analytical workflow and the sufficiency tool that analyzes the standards provided by the developer to determine whether this theory is beneficial. If the system is not appropriate to AMAS technology, the developer could utilize another AO approach. Defining agent structures includes determining their abilities and connections. The roles defined in the previous step can be useful in realizing such abilities. The conduct of agents can be getting by a set of potential current notations involving activity models, state charts, and finite state machines.

### 5.4.4 Implementation

Except for Gaia, All the participant methodologies provide backing tools that can produce code skeletons from design paradigms. This is a critical merit of modernistic software engineering approaches. ADELFE offers the analyst tools to evaluate the sufficiency of AMAS technology at two levels to demonstrate agent interaction protocols. In this respect, the AUML principle is used together with UML and RUP. PASSI utilizes the UML deployment model and expands it with merits to permit the mobility of agents to be defined. For example, in the process of building the deployment model, developers can have a better grasp of the intricacies to operate the system.

Also, developing high-scale deployment offers a basis for evaluating the feasibility of executing the system as well as the use of the spread concept. It also provides an assessment of different other measures, such as costs. Using this activity, PASSI describes the system based on agent classes and their position on the available processing units in the diagrams. O-MaSE also offers some limited backing for determining mobility processes in the case of a concurrent functions. The execution phase in ASEME is the language programming for different levels throughout the development process.

Later, during the verification phase, the system functions are checked against the requirements, and the respective phase is completed in relation to three abstraction levels: societal, agent, and capability. Nevertheless, the preferable method is successive, that is the software elements are checked for the effective running of algorithms, the agents are tested for the effective execution of abilities, and the MAS is checked for its general valid procedures in the end that is, all processes are carried out one after another. In general, this stage needs more improvement coinciding with the work on experimenting and addressing agent projects in recent years (Gomez-Sanz et al., 2008) (Nguyen et al., 2008).

## 5.5 Evalution a Unified AO Software Engineering Methodology

In this section, a unified AO software engineering methodology was evaluating by utilizing the three valuation techniques(structural analysis,case study, Meta-model) that approved in this study, comparison with other seven competing methodologies.

### 5.5.1 Structural analysis

Structural analysis is performed on each process and model, allowing us to determine the similarities between a unified AO methodology and the other methodologies as well as the benefits of each one. As already mentioned in section 3.2 the commonalities between the selected methodologies was as the following:

- Initial requirements
- Applying use-case requirement analysis
- The concept of environment
- Capturing goals
- Social system structure
- Agent acquaintance model

Our proposed methodologies take in account the initial requirements which offers the initial step of the requirements analysis towards identifying main stakeholders and to equip designers with useful knowledge about the environment in which software operates and the type of interactions to take place among system agents.

Like most of selected approaches the proposed methodology supports utilizing a specific UC technique which is used to characterize the different system functional requirements and has been demonstrated to be an efficient model in OO techniques in grouping system requirements. Also, it assists the developers in deciding on the main interactions among system entities. The unified approach adopted the new additions in ADELFE and ASEME approaches.

Also the notion of environment was existing in the proposed methodology .Often times, complicated open systems have very dynamic and diversified environments, with which defining and familiarizing is necessary if constant change is an issue. So, an agent system needs to have models to accurately represent the environment in which it works. We adopt the Prometheus's analysis overview model and characterize the environment activity in ADELFE to address this concept**.**

In the requirements analysis stage our proposed approach concur on the significance of goal concepts and identify goals to be applied as a basis for identifying agents. The proposed methodology address stakeholders' objectives and their relations via the functions and resources utilized to fulfill goals as carried out in the requirements analysis stage in ASEME. Goals can be systematiced and presented as a hierarchy of objectives as applied in O-MaSE. Also the social system and agent acquaintance concepts was rperesnted in our approach through the role ,agent communication and interaction models in the analysis phase.

Table 6. A proposed methodology and other methodologies

| | **initial requirement** | **using use cases** | **capturing goal** | **social structure** | **acquaintance model** |
|---|---|---|---|---|---|
| **Gaia** | X | X | √ | X | √ |
| **Tropos** | √ | X | √ | √ | X |
| **PASSI** | X | √ | X | √ | X |
| **O-MaSE** | X | X | √ | √ | X |
| **ADELFE** | √ | √ | X | √ | X |
| **ASEME** | X | √ | √ | √ | X |
| **Prometheus** | X | √ | √ | √ | √ |
| **Proposed methodology** | √ | √ | √ | √ | √ |

### 5.5.2 Case Study

To build an agent information project we applying the GA: PCHIS (see section 4.1.1) on the Proposed AO methodologie. The system is established so as to represent the hospital, family at home and healthcare providers all in one system.

**-Distinguish Use-Cases**

To assist the analysts in defining the main connections/reactions among entities and adapt more with agent technology, the proposed methodology integrated the UCM on ADELFE and ASEME approaches. Figure 44 shows an example of this combination by representing the UCM for the example used in this thesis (see section 3.1, Chapter 3).

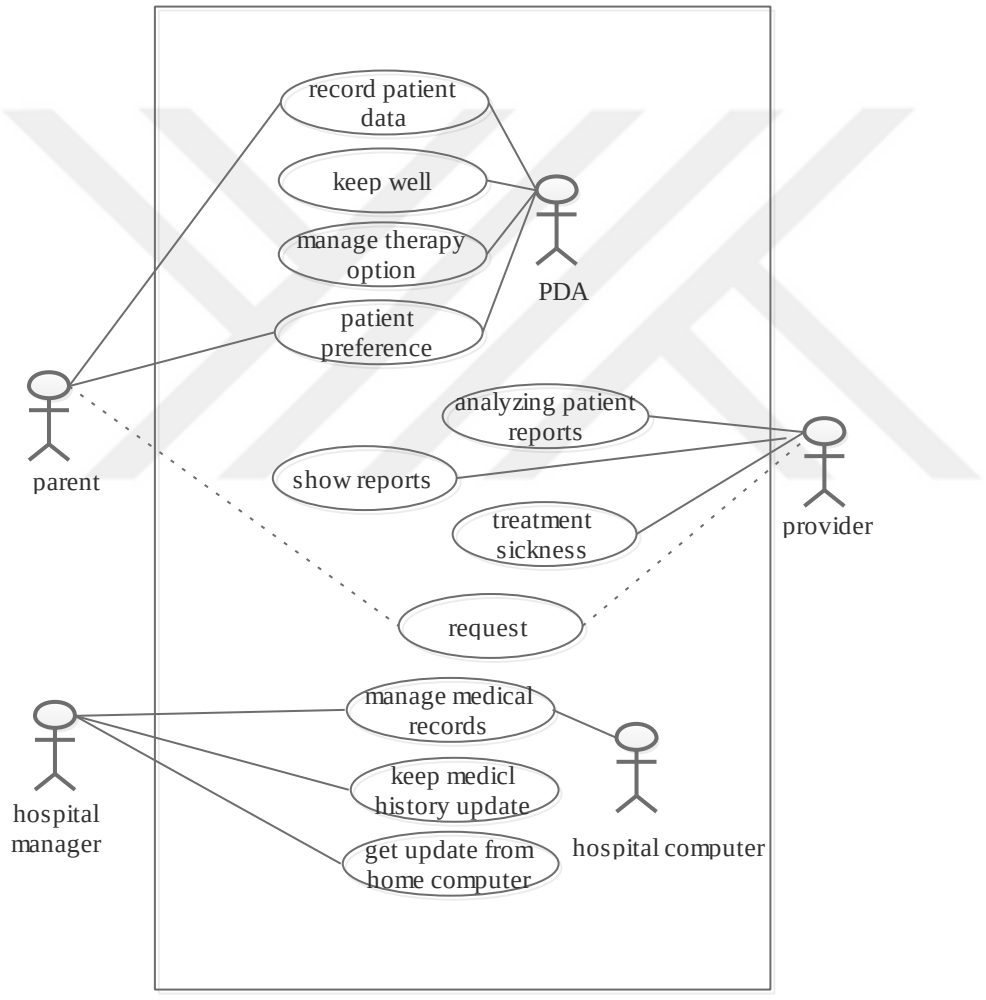


Figure 44. UCM for unified methodology

**-Goal Model**

It is substantial to take in account stakeholders' purposes and their interactions via the functions and resources utilized to fulfill goals as carried out in the requirements

analysis stage in ASEME. Goals can be organized and presented as a hierarchy of objectives as applied in MaSE, utilizing the suggested unified notation (Padgham et al., 2008). In the proposed methodology analysis phase the important step is geting the objectives, which picks the initial system charactarization and transforms them into an arranged group of system objectives, is offered in Figure 45.

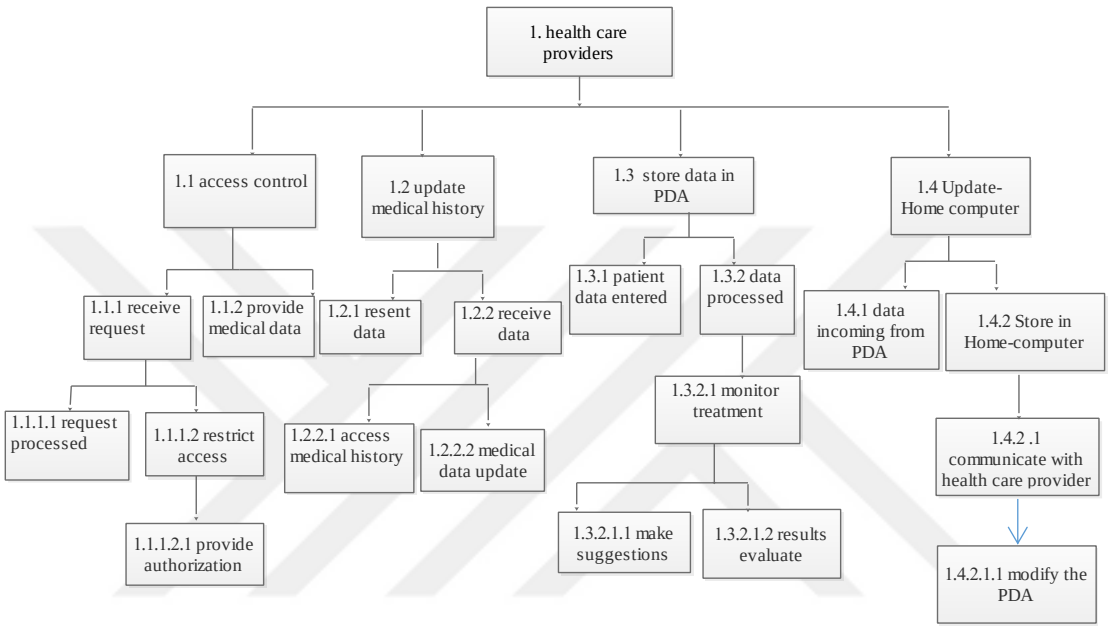


Figure 45. Goal hierarchy diagram

**- Role Model**

A basic stage applied in our unified approach is coping with agent role is to begin from the smaller elements of the agents and, then, to group these elements to compose agents. Figure 46 displays obvious and more full version of the role diagram, which includes information on the relations among role tasks. The goals linked with each role are represented under the role name. Also explained are the tasks linked with each role, utilized to define each role's conduct.

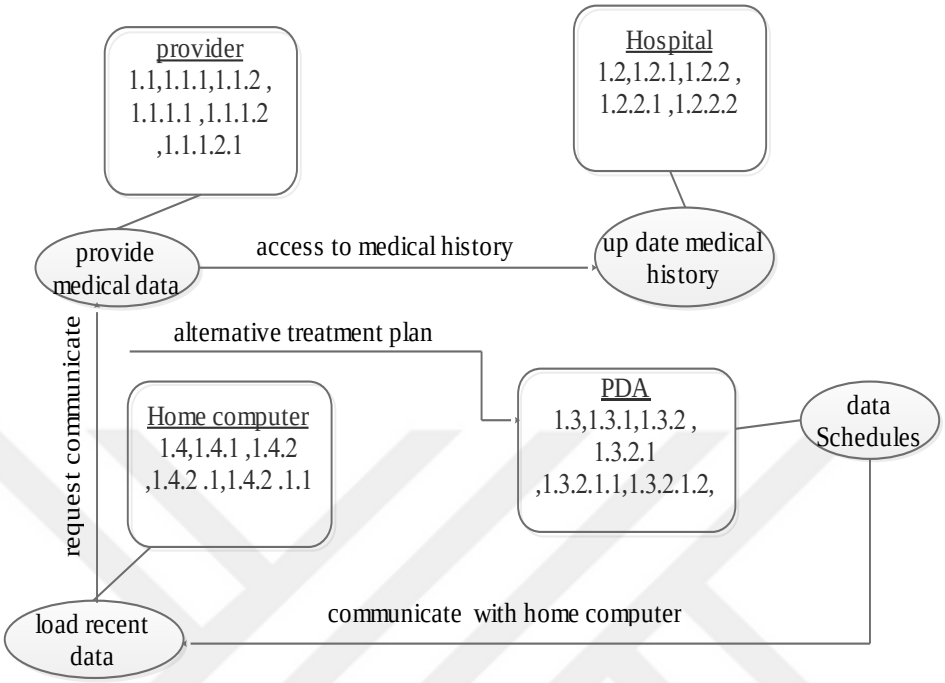


Figure 46. GA:PCHIS roles diagram

### 5.5.3 Meta-Model

The proposed methodology meta-model is intent on reconciling classical software engineering notions, such as problem and solving, with the possibility of the agent-founded method whilst striving to fulfil the objectives in accordance to the needs related to code execution. Based on the analysis of various MAS meta-models, the common entity will be provided to contribute in the construction and unification of the seven methodologies of MAS meta-models in one MAS meta-model. As showen Figure 41 the the proposal of integrated MAS meta-model offers a group of notions as part of the proposed unified AOSE methodology.

According to this diagram, the meta-model is directed by the objective of predefined organizations, is set in a particular structure, and puts constraints to systemize the act among agents, which plays different roles in the system; An agent exists in an

environment for which it is capable to construct representations if it has **these** abilities. Table 7 present a compration among a proposed methodology and other methodologies meta-models.

Table 7. A proposed methodology and other methodologies meta-model

| Meta-Model | Common concepts | Peculiarities |
|---|---|---|
| **Gaia** | Agent type, Organization, Organization rule, Role, Communication, Responsibilities, Protocol, Environment, Resources, Service. | Interactions |
| **Tropos** | Agent, Actor, Goal, Plan ,Recourse, Dependency, Capability, Belief | And /OR decomposition, Mean-Ends analysis |
| **PASSI** | Agent, FIPA-platform, Role, Service, Task, Communication, AIP, Recourses, Action | Concept, Ontology, Predicate |
| **O-MaSE** | Organization ,Agent, Role, Goal, Capability, Domain Model, Policy | Domain model, Policy |
| **ADELFE** | Representation, Environment, Communication, AIP | Cooperative agent, , Non- cooperation situations, Element |
| **ASEME** | Actor, Goal, Role, Protocol, Capability, Activity | Transition, Node, Model |
| **Prometheus** | Actor, Goal, Role, Organization, Agent Capability | Domain model, External model, Internal protocol |
| **Proposed methodology** | Actor, Goal, Role, Organization, Agent Capability, Representation, Environment, Communication, AIP, Service. | Interactions Rules, Hierarchy concept |

# CHAPTER 6

# DISCUSSION AND CONCLUSIONS

## 6. 1 Discussion

In an environment that is generally restricted, the current agents and MASs provide solutions for complicated applications. However, the present and future applications are convoluted and open to criticism. Like the Internet, these develop in an unpredictable environment and present other challenges to constructing software. As a result, it is important to offer novel paradigms, tools and approaches to meet these challenges. Many of the selected methodologis are picked according to “standards”, like the RUP, UML and UML models, to improve AO technologies in a manufacturing world, where OO technology is the base.

In terms of strengths, weaknesses and application domains of seven methodologies, the following tables describe the results of this analysis study. Gaia is a general approach that is usable to a wide domain of MAS; furthermore, it addresses the social agent aspect of systems. It is based on the idea of MAS as a computational organization consisting of different interacting roles (Wooldridge & Jennings, 1999). Gaia does not directly deal with special modeling technicalities. Also, it does not include an implementation stage.

As we mentioned the difficulties were spotted when working on some design schemas and role models, with the underlying reason being that performing elaborated role models needs determining the liveness responsibilities of each role, decisive in plotting the protocols later. In addition, the authorizations and safely responsibilities mentioned in Gaia are not clear. As an outcome, in the paradigm of the role set up for the target system, this feature of roles is not well-described. Regarding the notation which characterizes these role's characteristics, liveness responsibilities are rationally well-identified, while liability is not submitted.

The results of the Gaia process are detailed but technologically neutral, making it supposedly easily implemented using a suitable program framework (e.g., a FIBA compliant agent system), a modern object component, or a component framework.

Gaia does not overtly tackle the processes of requirements collection, specifically of preliminary requirements engineering (Mylopoulos et al., 1999). The importance of these activities is being increasingly along with the ongoing work in this area. Also, Gaia can fit and easily integrate with the modern goal-oriented methods for engineering requirements (Castro et al., 2002) where abstractions closely correlate with those AO computing.

As for Tropos, it was affected by the framework (Frank, 1998) analysis of the initial demands of the intended system. It directs designers toward understanding an AO system as an organizing of actors. Through plans that are dependent on other actors, which are seeking to accomplish goals. The Tropos methodology aims to cover all analyses and design processes in the software development lifecycle, starting with the analysis stage to system implementation.

It provides the early phase of the requirements analysis with identifying basic stakeholders and equips the designers with a satisfactory knowledge about the environment in which the software operates and the type of interactions which should happen among system agents.

In the requirements analysis, the main supposition that distinguishes Tropos from other methodologies is that actors and objectives are utilized as basic notions for

modeling and analysis during all software lifecycle stages, not just early requirements. It concentrates on the preliminary stage of requirements engineering. The discovery of stakeholders and credits among them, lies in the knowledge of the engineer. Based on various perspectives that could lead to different results, achievement of objectives and their relations among stakeholders can be derived.

In the analysis phase, a new sub-goal is created to improve the target system, and the dependencies between the new objectives and each element in the system must be recalculated (Presiani et al., 2004). In the analysis phase, the analysis of dependency plays a very important role.

According to the developers' interpretation, statements of goals and dependencies are likely to be unclear. Depending on the interpretations of the software engineer, there are no uniform guidelines for follow-up while analyzing goals or tasks in sub-goals or sub-tasks.It is still difficult to track back all the dependencies in spite of the tracing being available through the notation graphs.

The system actors is added in the midest of the analysis. To accommodate the new actor, the dependencies among the actors are rearranged. For supporting this approach, there is no formal justification. Throughout the entire development process, Tropos is based on the unified use of small groups of intentional symbols (Perini & Sussi, 2003).

Nevertheless, in order to rethink beliefs and plans in an adaptive manner, it is difficult to think of a changing environment. O-MaSE is designed as a set of parts that developers combine to meet specific goals. In fact, they do not need a particular set of stages. They assume that they follow a traditional waterfall approach to allay this problem. Requirements analysis, design, and implementation are the three main phases with the main activities allocated as expected.

It is the beginning point in MaSE (DeLoach et al., 2001). In order to design MASs, there are several drawbacks. For example, MaSE presents MASs with a specific organization. The number of agents and their roles in MaSE is limited. Also, MaSE

does not cover the concept of sub-teams and has no diagrams that represent interactions with the environment.

In addition, there is only one-to-one connection among agents in the system and it does not clearly define the usage case model (Sukhvir et al, 2012). While many recurring problems have been processed in O-MASE, there are some necessary missions for a mature agent approach like management, product distribution, and testing and assessment which are absent.

In PASSI, an agent is an essential part of the software at both low- and high-precision levels. The agent is an example an example of software execution of an independent element able to achieve a goal via its independent resolutions, activities and social interactions based on this view (Cossentino, 2005). While interacting with other agents to achieve their objectives, the agent can play many useful roles representing a set of functions carried out by the agent in pursuit of a sub-goal. As a meaningful unit of individual or interactive behavior, the task is defined in PASSI.

ADELFE directs the designer in making the resolution as to if AMAS theory is desired in the project under development. This illustrates the necessity of verifying the sufficiency of the analytical workflow and the sufficiency tool that analyzes the standard provided by the developer to determine whether this technology is beneficial.

If the system is not appropriate to AMAS technology, the developer could employ another AO approach. ADELFE does not assume that the agents of the system are designed to be known. It provides certain actions and standard to assist the developer to determine elements in the system that need execution as agents (Bernon et al., 2003). Their characteristics should be studied in addition to the relations and the cooperation failures which may determine if entities should be regarded as agents.

This can be reutilized in other agent approaches, and the ADELFE processes can then be analyzed in the form of fragments. It will also be easier to combine the parts from other approaches into ADELFE (Hendreson-Sellers & Giorgini, 2005).

ADELFE's main strong point can also be a major constraint; it is highly specialized and cannot be used to develop all kinds of agents (e.g., BDI). For example, to develop a system like a simulation software, integrating them within another system like a simulation platform would be required. As it permits the designer only to test the behavior of certain agents and verify them based on specifications, there is a need to improve many activities, especially fast prototyping (Hendreson-Sellers & Giorgini, 2005).

To improve agent behavior by insertion or deleting portions of it, the designer will be able to react with the system as it is designed. Many work definitions still do not exist. Currently, for direct execution and testing, there is no running tool like the platform or a group of programming tools associated with this approach (Hendreson-Sellers & Giorgini, 2005).

As mentioned previously, ASEME, the Model-Driven Engineering (MDE) method appears with special features concerning the AOSE community, and can be applied in all stages of conventional software processes (from requirements to execution).

It allows the transition from one stage to another through typical transformations. Also it backs the registration of non-functional requirements in the requirements analysis step, where they are utilized to make management decisions and choose which technologies will be used for design and development.

Prometheus suggests complete lifecycle processes, from requirements specification to elaborated design, and supports the agent developer according to goals and tasks. It offers elaborated guidance on how to carry out the different stages. Prometheus has an environmental paradigm that describes the surroundings where agents work and provide support for this concept compared to other competitive approaches. However, the Prometheus environment model is limited to clear input and output specifications with respect to the distinct needs and requirements of the system.

In this methodology, the formation of the kinds of agents can be oriented to data association. In addition, the agents are composed of other component roles and abilities. Also, for individual agents, Prometheus supports both dynamic and static

models (Bawa et al., 2015). Developers can get the information that agents draw from the environment and the actions that the agents begin to respond to in these events using the system specification stage in Prometheus.

Nevertheless, Prometheus is not without drawbacks and its support for the socialite aspect of agents focuses on the lowest common divisor: messages and protocols. Also, Prometheus does not handle mobile agents at all, and has less focus on initial requirements and business processes analysis than an approaches, such as ADELFE and Tropos. lastly, Prometheus is not established on UML.

Table 8. Comparative Analysis of GAIA and Tropos

| | **Strengths** | **Limitations** | **Application Domain** |
|---|---|---|---|
| **Gaia** | • Directs designers to a well-defined design for the MAS<br>• Model and coping with the features of complex systems<br>• Is easy to apply | • Agents cannot share common goals.<br>• It does not handle systems while acting as a actor.<br>• The coherence between the protocols is rather weak.<br>• Components of system may enter or leave at runtime in open or dynamic systems.<br>• It addresses goals, but verification of goals is still out of scope<br>• It does not use case scenarios. (Sukhvir et al, 2012) | • Application of Agent Methodology in Healthcare Information Systems (Abdalla & Mishra, 2017)<br>• Multi-agent system vulnerability detector for a secured E-Ieaming environment (El Hajj et al., 2016)<br>• Multi-agent based Transformer Condition Monitoring (Saminni et al., 2012) |
| **TROPOS** | • Suggests methods and tools for automating models transformation<br>• Focuses on preliminary stage of requirements<br>• Uses actors and goals as basic concepts (Hendreson-Sellers & Giorgini, 2005) | • It is not designed to support a certain type of software.<br>• Developed software agents designed for accomplishing special goals are not supported by Tropos methodology<br>• Does not use case scenarios (Sukhvir et al, 2012). | • Developing Adaptable and Open Service Systems: Application in Supply Chain Management (Wautelet et al., 2009)<br>• Using tropos to model agent based architectures for adaptive systems: a case study in ambient intelligence (Bresciani et al., 2005)<br>• Developing a decision support system for integrated production in agriculture (Perini & Susi, 2004) |

Table 9. Comparative Analysis of PASSI and O-MaSE

| | Strengths | Limitations | Application Domain |
|---|---|---|---|
| **PASSI** | • Suggests a complete lifecycle methodology from requirement-to-code methodology<br>• Integrates design models and concepts from both OO and MAS using UML notation<br>• Refers to the most diffused standards: UML, FIPA, JAVA, Rational Rose(Cossentino, 2005) | •.The need to refer simultaneously to various models in order to understand the system and the way it works and changes over time is a critical issue.<br>• Every model offers its own set of notation and special concepts, resulting in an abnormal complexity in terms of vocabulary.<br>• Does not support the environment model(Cossentino, 2005). | • A Domain Analysis Approach fo MASs Product Lines(Nunes et al., 2009)<br>• The Development of a Multi-agent based Middleware for RFID Asset Management System using the PASSI Methodology (Massawe et al., 2009)<br>• Patterns Reuse in the PASSI methodology (Cossentino et al., 2000) |
| **O-MaSE** | • Is comprehensive for building MAS.<br>• Is Step-by-step lifecycle methodology MAS.<br>• Provides guidance throughout the entire software development lifecycle.<br>• Open systems are considered, thus agents can be created, deleted or moved during implementation | • Some of the software applications are closed.<br>• Management, product distribution, and testing and assessment have been absent.<br>• There is only one-to-one connection among agents in the system<br>• It does not clearly define the usage case model (Sukhvir et al, 2012) | • The Multi-Agent Distributed Goal Satisfaction project (Saba& Santos, 2000)<br>• Agent-Based Mixed-Initiative Collaboration project (Cox et al., 2000)<br>• Developing a Multi-agent Conference Management System Using the O-MaSE Process Framework (Deloach , 2007)<br>• A multi-agent approach to a biologically based computer virus immune system (Harmer & Lamont, 2000) |

Table 10. Comparative Analysis of ADELFE and ASEME

| Methodology | Strengths | Limitations | Application Domain |
|---|---|---|---|
| **ADELFE** | • Can be used by a non-specialist in agent systems.<br>• It will also be easier to combine the parts from other approaches into ADELFE.<br>• Use the cooperation rules (Bernon et al., 2005).<br>•ADELFE allows different UML/AUML realizations. | • It is specialized, thus it cannot be used to design all existing applications or to model all kinds of agents.<br>• Some definitions of work still do not exist.<br>• Sometimes the developer may find the graphical modeling tool difficult to use (Bernon et al., 2005). | •Tools for Self-organizing applications engineering (Bernon et al.,2003).<br>• A sample application of ADELFE focusing on analysis and design the mechanical synthesis problem (Capera el al., 2005).<br>• A Tool for Generating Model Transformations By-Example in Multi-Agent Systems (Garcia-Magarino et al., 2009).<br>• ADELFE 3.0 Design, Building Adaptive Multi Agent Systems Based on Simulation a Case Study (Mefteh et al., 2015 |
| **ASEME** | • Supports of documentation of non-functional requirements.<br>• Provides the model transformations among the various development stages (Jorge et al, 2015). | • The guidelines are missing because the authors rely only on the paradigm shifts.<br>• The same thing happens with the tasks to be conducted in each stage of the development.<br>• Some processes need to be improved.<br>• In requirements, implementation and design, there are no guidelines for the production of models(Jorge et al, 2015). | •Automated Product Pricing Using Argumentation(Spanoudakis & Moraitis, 2009).<br>• Using ASEME methodology for model-driven agent systems development(Spanoudakis & Moraitis, 2011).<br>• Engineering an agent-based system for product pricing, automation(Spanoudakis & Moraitis, 2011). |

Table 11. Comparative Analysis of Prometheus methodologies

| Methodology | Strengths | Limitations | Application Domain |
|---|---|---|---|
| **Prometheus** | • Offers a complete lifecycle methodology from requirements specification to detailed design.<br>• Supports both dynamic and static models for individual agents.<br>• Provides elaborated guidance on how to carry out the different stages.<br>• Gives clear support to the environment concept. | • Less focus on initial requirements.<br>•Does not deal at all with mobile agents.<br>•Cannot be established on UML<br>•Support for the socialite side of agent focuses on the lowest common divisors: messages and protocols | •An open meteorological alerting system: Issues and solutions(Mathieson et al., 2004).<br>• Tool support for agent development using the Prometheus methodology in**:** Quality Software(Padgham et al.,2006)<br>• AO modeling and development of a person-following mobile robot (Gascueña ,& Fernández-Caballero, 2011) |

## 6. 2 Conclusion

This study conducted an analysis and evaluation of key AOSE methodologies, focusing on their features and limitations. With the increasing necessity for complicated systems in the industry, the intention to employ agent technologies to promote commercial and industrial software systems is rising rapidly as well, moving the industry from a form of product for a user to a manner of delegation, where it can also involve the user in order to make decisions and take actions accordingly. Thus, the availability of AO approach to support software engineers in designing agent-based projects is very significant. To this end, the presdent research carried out a comparison of seven prominent AO methodologies to comprehend the interactions among them.

In particular, our major objectives were: Firstly, to determine the resemblance and variations in terms of processes and models that are paramount in directing the development of agent-based projects, and secondly, to estimate each methodology's features, limitations, and range of usability. The assessment of the AOSE method reveals that most amount of attention has been paid to requirements, design, and implementation stages. However, progress is still needed in all stages of the software processes.

AO software engineering systems are often distinguished with intelligence, autonomy, and reasoning. The main purpose for these methodologies is to deliver the required processes, models, mechanisms and tools so that agent-based systems can be developed in a more formal and organized manner. As pointed out previously, a large number of AOSE methodologies with various backgrounds have been made available to experts in recent years.

However, assessment of these methodologies has continuously faced several difficulties; the completeness of various methodologies varies explicitly, and each methodology has different advantages and drawbacks with numerous individually-oriented features to support various aspects of their proposed application scopes. For these reasons, comparing agent methodologies is more difficult; as they cover different aspects or differ in their terminology.

The objective of this research study is to make an attempt toward a standardized approach. A collection of artifacts and ideas is made from different key AOSE techniques. Yet, it is crucial to comprehend the relationship among them including the strengths, weaknesses and applicability of each methodology. Also, we have compared the meta-models that are associated with available methodologies and unified them in one meta-model. Based on this comparative analysis, we proposed an initial unification of the assessment AO approaches by integrating their strong characteristics.

As mentioned in section 2.5.3, various evaluation techniques are available for methodologies' assessment. Two forms of feature analysis (case study and structural analysis approaches) and meta-model were selected. A structural analysis is performed to examine and identify the common core of all the models and processes as well as the various components of each methodology. This method is used so as to identify the common cores and components of the seven methodologies.

As a second step, a case study was conducted in which we designed for the same application GA: PCHIS using different approaches. The effective utilization of the seven agent approaches in designing the GA: PCHIS indicated that they are practical and usable. Additionally, meta-models are also utilized to get a better understanding of the agent notions as applied in AO approaches and to look for some agreement in the major components that can be utilized to identify MAS.

Several MAS meta-models have been offered alongside an initial proposal of integration based on the seven AO approaches (Gaia, Tropos, PASSI, O-MaSE, ADELFE, ASEME, and Prometheus). This was doen to analyze and compare the meta-models associated with these methodologies so as to learn about their commonalities and definitively determine the essential notions utilized in MAS to combine them into one MAS meta-model.

In our view, all of the seven AO methodologies are clearly agent-based, the structral is widely regarded as an agent methodology and most of them consider the social aspects. Generally, all seven approaches provide sufficient support for major AO

properties. The most social characteristic addressed in all seven approaches offer support for most social standards as communication, communication language and relationship.

However, there are numerous aspects of agent-based systems that were not sufficiently handled in most of the approaches. For example, except ASEME, none of the seven approaches provides backing registration of non-functional requirements, in the requirements analysis step. Regarding the environment percepts feature, ADELFE, Prometheus and O-MaSE provide support for modeling the environment of agents. As to architecture, all seven methodologies provide support with different degrees to agent architecture, while Prometheus introduces distinctive backing for belief, goal, and plan, the other methodologies seem to be weak in this respect.

The notation of the seven approaches is acceptable and most have a modeling language in terms of fulfilling different aspects such as systematic transitions, modularity and ease of comprehension. However, none of the approaches clearly offers mechanisms or models to back the design of reusable components. In addition, only ASEME provides techniques and adequate support for transition from one stage to another through typical transformations by three forms of conversion used for automation between the ASEME models.

Regarding the process lifecycle, all of the approaches reference the specification, analysis, design and detailed design to a certain degree. Additionally, all of them offer instances and heuristics to help designers in all methodology stages. Tropos and ADELFE make use of initial and final requirements, which is an important stage in these approaches. Moreover, they suppose full specification of requirements and do treat the requirement-grouping. This guides designers to the benefit of richer requirements produced by agent technologies. The seven methodologies also share several common features; for example Prometheus and Gaia share the use of the acquaintance technique. In addition, all of the seven methodologies provide models to cover the social system structure.

Distinguishing features also exist, such as in PASSI, from other participant methodologies. In case of PASSI, this lies in its strong focus on the deployment

model, which is regarded as one of the main models in UML. Using this activity, PASSI describes the system based on agent classes and their position on the available processing units in the diagrams. Additionally an important aspect which distinguishes Prometheus from other participant methodologies is about data coupling as it offers evident processes to cope with agent characterizations.

### 6.3 Critical Review

The survey method was within the suggested scope to make more efficient assessments by providing a wide range of perspectives, but the authors had time and faced other difficulties such as a lack of resources.

Also ,in proposed methodology the design stage have a limitation in represent the behavior of agent . Despite the different perspectives regarding agent conduct , none of competing is methodology better than the others and all are equally dissatisfactory and in sufficient. This stage is still in its infancy and needs further development. The implementation stage is still in its infancy and needs further development.

### 6.4 Future work

With regard to future work, it includes increasing the number of agent-oriented methodologies and adding other evaluation methods to the assessment. In doing so, the work can be improved by involving helpful models and techniques. Also, this work has looked for the commonalities and differences between participant methodologies in terms of process and models without the detailed techniques, which are an important issue for future work.

Also building a framework can be attempted to include the most commonly accepted properties of agents. However, as application complications increase, we anticipate future systems to have an increasingly large number of features to consider that may not be covered by our work. To provide assessment for such aspects, it must have particularized characteristics to cover various new sides.

## REFERENCES


Abdalla, R.&Mishra,A. (2017). Application of Agent Methodology in Healthcare Information Systems. *Tem Journal -Technology Education Management Informatics*. Vol. 6.Issue: 1.Pages: 147-152.

Alonso, F., Frutos, S. Martinez, L. & Montes, C. (2003). SONIA: A Methodology for Natural Agent Development.28660 *Boadilla del Monte (Madrid), Spain*.

Bawa,A. , Bhatia S. & Kaur Attri, V. (2015). A REVIEW ON AGENT ORIENTED SOFTWARE ENGINEERING. 5940 *International Journal of Advanced Research in Computer and Communication Engineering* Vol. 4, Issue 4, April 2015 Copyright to IJARCCE DOI 10.17148/IJARCCE.2015.4495 421.

Bernon ,C., M.-P., Gleizes ,G.& Glize P. P. (2002).The Adelfe Methodology for an Intranet System Design, *Fourth International Bi-Conference Workshop on Agent-Oriented Information Systems* at CAiSE'02, Toronto, Canada, P.

Bernon C., Cossentino. M, Gleizes ,M., Turci ,P.,& Zambonelli, F.(2004). A Study of some Multi-Agent Meta-Models. *5th International Workshop, AOSE 2004, New York,* NY, USA, July 19, 2004.

Bernon, C., Camps, V., Gleizes, M. P. & Picard, G. (2005). Multi-agent systems: the ADELFE methodology. *In Agent-Oriented Methodologies, Henderson-Sellers, B. & Giorgini, P. (eds), Idea Group Publishing*, Ch. VII. 172–202.

Bernon, C., Camps, V., Gleizes, M. P. & Picard, G. (2003). ADELFE: A Methodology for Adaptive Multi-agent Systems Engineering. In: *Petta P., Tolksdorf R., Zambonelli F. (eds) Engineering Societies in the Agents World* III. ESAW 2002. Lecture Notes in Computer Science, vol 2577.

Bernon, C, Camps,V., Gleizes, M. & Picard, G.(2003).Tools for Self-Organizing Applications Engineering. *1st International Workshop on Engineering Self-Organising Applications* (ESOA2003) Location: Melbourne, AUSTRALIA Date: JUL 15, 2003.

Booch, G.(1994). Object-Oriented Analysis and Design with Applications. Redwood City CA: *The Benjamin/Cummings Publishing Company*.

Booch, G. , Rumbaugh ,J. & Jacobson ,I.(1998). The Unified Modeling Language User Guide.Addison Wesley.

Bresciani, P.,Kuflik, T. & Busetta, P.(2005). Using tropos to model agent based architectures for adaptive systems: a case study in ambient intelligence.*IEEE International Conference on Software - Science, Technology and Engineering*, Proceedings.Pages: 37-46.DOI: 10.1109/SWSTE.2005.23.

Bresciani, P. Giorgini,P. Giunchiglia,F. Mylopoulos, J. Perini, A.(2004). Tropos: an agent-oriented software development methodology, *Journal of Autonomous Agents and Multi-Agent Systems* (JAAMAS) 8 (2004) 203–236.

Brazier, F. M.T., Dunin-Keplicz, B. M., Jennings, N. R., & Treur, J.(1997). DESIRE: Modelling multi-agent systems in a compositional formal framework. Int *Journal of Cooperative Information Systems*, 6(1):67-94.

Burmeister, B. (1996). Models and Methodology for Agent-Oriented Analysis and Design In: *Fischer, K. (ed): Working Notes of the KI'96 Workshop on Agent-Oriented programming and Distributed systems,Saarbrucken,Germany*.

-Brinkkemper, S., Hong, S. & van den Goor, G. (1993). A formal approach to the comparison of object-oriented analysis and design methodologies, in *Proc.of the Twenty-Sixth Hawaii Intl*. Conf. on, Vol. 4, pp. 689-698, Jan.

Bush,G.,Cranefied,S. & Purvis,M. (2001).The Styx Agent Methodologies, *university Otago ISSN 1172-6042 January* 2001.

Castro,J, Kolp,M.& Mylopoulos,J. Towards Requirements-Driven In-formation Systems Engineering: The Tropos Project. *Information Systems. Elsevier, Amsterdam, the Netherlands*, (to appear).

Caire, G., Leal, F.(2001). Project p907, deliverable 4: Recommendations on supporting tools. Technical Information Final version, *European Institute for Research and Strategic Studies in Telecommunications* (EURESCOM), July 2001.

Capera, D., Picard, G. Gleizes, MP& Glize, P.(2005). A sample application of ADELFE focusing on analysis and design the mechanical synthesis problem. Engineering Societies In *The Agents World Vbook Series: Lecture Notes In Artificial Intelligence.* Vol.: 3451Pages: 231-244.

Capera D., Georgé, JP., Gleizes M-P., and Glize, P.(2003). The amas theory for complex problem solving based on self-organizing cooperative agents. In *1st International workshop on Theory and Practice of Open Computational Systems (TAPOCS) at IEEE 12th International Workshop on Enabling Technologies*: Infrastructure for Collaborative Enterprises (WETICE 2003), pages 383–388. IEEE Computer Society, 9-11 June 2003.

Capera D., Georgé, JP., Gleizes M-P., and Glize, P.(2003). Emergence of organisations,emergence of functions. In FAISB’03 symposium on Adaptive Agents and Multi-Agent Systems, April 2003.

Cernuzzi, L.& Rossi, G.(2002). On the evaluation of agent oriented modeling methods. In *Proceedings of Agent Oriented Methodology Workshop, Seattle*, November 2002.

Cernuzzi & Zambonelli, F. (2004). Experiencing AUML in the GAIA methodology, *6th International Conference on Enterprise Information Systems*, pp. 283–288, Porto,Portugal, April.

Chia-En L., Krishna M., Frederick T. & Sheldon and Thomas E.(2005). A Methodology to Evaluate Agent Oriented Software Engineering Techniques. *The International Journal of Multiagent and Grid Systems*, Nov. 2005

Chung L. K., Nixon, B. Yu, A. E., and Mylopoulos, J.(2000). Non Functional Requirements in Software Engineering, Kluwer Publishing, 2000.

Collinot, A., Drogoul, A. & Benhamou, P.(1996). Agent oriented design of a soccer robot team. In *Proceedings of the Second International Conference on Multi-Agent Systems, Kyoto, Japan*, December 1996.

Cossentino, M. & Potts, C.(2002). A case tool supported methodology for the design of multi-agent systems, *International Conference on Software Engineering Research and Practice*, Las Vegas, USA, June 2002.

Cossentino, M., Sabatucci, L., Sorace, S., & Chella, A.(2000). Pattern reuse in the PASSI methodology, ESAW'03, *Imperial College London*, UK, October 2000.

Cossentino, M. (2005). From requirements to code with the PASSI methodology. In Agent-Oriented Methodologies.

Cox, M., Kerkez, B., Srinivas, C., Edwin, G.& Archer, W.(2000). Toward Agent-Based Mixed-Initiative Interfaces. In Proceedings of the 2000 International Conference on Artificial Intelligence. CSREA Press (2000).

Cribbs, J., Roe, C. & Moon, S.(1992). An Evaluation of Object-Oriented Analysis and Design Methodologies. *SIGS Books, New York*, 1992.

Dam, K.H, Winikoff, M.,and Padgham, L.(2006). An agent-oriented approach to change propagation in software evolution. *Australian Software Engineering Conference*, ASWEC 2006 (pp. 309-318). Australia: IEEE.

Dam, K.H. & Winikoff, M.(2004). Comparing agent-oriented methodologies, in: *P. Giorgini, B. Henderson-Sellers, M. Winikoff (Eds.), Agent-Oriented Information Systems, AOIS*, in: *Lecture Notes in Computer Science, vol. 3030*, Springer, 2004, pp. 78–93.

Dam,H.(2003). Evaluating and Comparing Agent-Oriented Software Engineering Methodologies. *MSC Thesis, RMIT University*, Australia.

Dam, H. & Winikoff, M.(20013). Towards a next-generation AOSE methodology, Science of Computer Programming, 78 (2013)684-694. doi:10.1016/j.scico.2011.12.005 (2013).

Dalpiaz, F., Molesini, A. & Puviani, M.(2007).:Towards filling the gap between AOSE methodologies and infrastructures: requirements and meta-model.

Debenham, J., Henderson-Sellers, B.(2002). Full lifecycle methodologies for agent oriented systems - the extended OPEN process framework. In *Proceedings of Agent-Oriented In- formation Systems* (AOIS-2002) at CAiSE'02, Toronto, May 2002.

DeLoach, S.(1999). Multiagent systems engineering: a methodology and language for designing agent systems, *Agent Oriented Information Systems*, May.

DeLoach, S.A., Garcia-Ojeda, J.C., (2010). O-MaSE: a customizable approach to developing multi-agent development processes, *International Journal of Agent-Oriented Software Engineering* 4 244–280. doi:10.1504/IJAOSE.2010.036984.

DeLoach, S.A., Garcia-Ojeda, J.C., (2014). The O-MaSE Methodology. *Chapter from book Handbook on agent-oriented design processes* (pp.253-285).

DeLoach, S.A., Miller, M.(2010). A goal model for adaptive complex systems. *International Journal of Computational Intelligence: Theory and Practice*. 5, 83–92 (2010).

DeLoach, S. A., Wood M. F.(2000). Multiagent Systems Engineering: the Analysis Phase. *Technical Report, Air Force Institute of Technology*, AFIT/EN-TR-00-02, June 2000.

DeLoach, S. A. & Kumar, M. (2005). Multi-agent systems engineering: an overview and case study. In *Agent-Oriented Methodologies, Henderson-Sellers, B. & Giorgini, P. (eds), Idea Group Publishing*, Ch. XI. 317–340.

DeLoach, S.A. , Valenzuela, J.L.(2007).An agent-environment interaction model', in *Padgham, L. and Zambonelli, F. (Eds.):* AOSE VII/AOSE 2006, LNCS Vol. 4405, Springer, Heidelberg.

DeLoach, S.A., Oyenan, W. and Matson, E.T. (2008) 'A capabilities based model for artificial organizations', *Journal of Autonomous Agents and Multiagent Systems*, Vol. 16, No. 1, pp.13–56.

Dumke, R., & Foltin, E.(1998). Metrics-based evaluation of object-oriented software development methods. In *Proceedings of the 2nd Euromicro Conference on Software Maintenance and Reengineering* (CSMR'98), pages 193-196, Florence, Italy, March 8-11 1998.

El Hajj, F., El Hajj, A.& Chehade, RA.(2016). Multi-agent system vulnerability detector for a secured E-Ieaming environment.*IEEE2016 SiXTH International Conference On Digital Information Processing and Communications*. Pages: 113-118.

Ericsson, N.(2004). TECHNICAL DESCRIPTION, Integrated Health Care Information System. Croatia, April 2004.

Erol, K., Lang, J. & Levy, R. (2000). Designing Agents from Reusable Components. In *Proc. of the fourth international conference on Autonomous agents*, pages 76–77,

Fabiano, D, Ambra, M.& Mariachiara, P.(200). Towards filling the gap between AOSE methodologies and infrastructures: Requirements and metamodel, *in: M. Baldoni, M. Cossentino, F. De Paoli, V. Seidita (Eds.), 9th Workshop From Objects to Agents, WOA 2007, Seneca Edizioni, Palermo, Italy*, 2007, pp. 115–121. URL: http://www.pa.icar.cnr.it/woa08/materiali/Proceedings.pdf.

Frank ,U. (1998). Evaluating modelling languages: relevant issues, epistemological challenges and a preliminary research framework. Technical Report 15, Arbetsberichte des Instituts fuer Wirtshaftsinformatik (Universitt Koblenz-Landau).

Frank, U.(1993). A comparison of two outstanding methodologies for object-oriented design. Technical Report No. 779, Arbeitspapiere der GMD, Sankt Augustin, 1993.

Frutos, S. (2003). Modelo de Diseno duna Arquitectura Multi -Agente Basado en un Modelo de Sociedad de Agents " , *Ph.D Thesis. Universidad Politecnica de Madrid, Spain*.

Giacomo C., Letizia L., Mariachiara P.(2007). Service-Oriented Agent Methodologies. *16th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises* (WETICE 2007) 18-20 June 2007.

Garcia-Magarino, I., Rougemaille, S., Fernandez, RF., Migeon, F., Gleizes, MP.& Gomez-Sanz, J.(2009). A Tool for Generating Model Transformations By-Example in Multi-Agent Systems. *The International Conference ON Practical Applications OF Agents and Multi-Agent Systems* (PAAMS 2009).Book Series: Advances in Intelligent and Soft Computing Volume: 55.

Gascueña , M. and Fernández-Caballero,A. (2011) .Agent-oriented modeling and development of a person-following mobile robot. In *Expert Systems with Applications* Volume 38, Issue 4, April 2011, Pages 4280-429.

Gervais, M. (2003). ODAC: An Agent oriented Methodology Based on ODP.*Journal of Autonomous Agent and Multi-Agent systems*.

Giorgini, P., Kolp, M., Mylopoulos, J. & Castro, J. (2005). Tropos: a requirements-driven methodology for agent oriented software. In *Agent-Oriented Methodologies, Henderson-Sellers, B. & Giorgini, P. (eds), Idea Group Publishing*, Ch. II. 20–45.

Giunchiglia, F., Mylopoulos ,J. & Perini, A. (2002). The Tropos software development methodology: Processes, Models and Diagrams. In *Third International Workshop on Agent-Oriented Software Engineering*, July.

Glaser, N. (1996). Contribution to knowledge modelling in a multi-agent framework (the CoMo- MAS approach). *PhD Thesis, L'Universite Henri Poincare*.

Glaser, N.(1997).The CoMoMAS Methodology and Environment for Multi Agent System Development. In:*Zhang,C.,Lukose,D. (eds.): Multi Agent Systems-Methodologies and Applications,*LNAI1286. Springer-Verlag,Berlin.

Gleizes, F. M-P, Zambonelli, F. (2004).Methodologies and Software Engineering for Agent Systems, Kluwer Academic Publishers.

Gomez-Sanz J.J., Bot ,J., Serrano, E., Pavón, J.(2008). Testing and debugging of MAS interactions with INGENIAS, in *J.J. Gomez-Sanz, M. Luck (Eds.), Ninth International Workshop on Agent-Oriented Software Engineering,* AOSE, 2008, pp. 133–144.

Harel, D. & Kugler, H.(2004). The rhapsody semantics of statecharts (or, on the executablecore of the uml) - preliminary version. In: Ehrig, H., Damm, W., Desel, J., Groe-Rhode, M., Reif, W., Schnieder, E., Westk¨amper, E. (eds.) INT 2004. LNCS, vol. 3147, pp. 325–354. Springer, Heidelberg (2004).

Henderson-Sellers, B. & Giorgini, P. (2005).Agent Oriented Methodology (eds), Idea *Group Publishing*, Ch. IV. 79–106.

Hoa,.D & Michael,W.(2013). Towards a next-generation AOSE methodology, *Science of Computer Programming*, 78 684-694. doi:10.1016/j.scico.2011.12.005.

Hoa, D. & Winikoff, M.(2004). Comparing agent-oriented methodologies., *in: P. Giorgini, B. Henderson-Sellers, M. Winikoff (Eds.), Agent-Oriented InformationSystems (AOIS), Vol. 3030 of Lecture Notes in Computer Science*, *Springer*, pp. 78–93.

Hong, S., Van den Goor, G. and Brinkkemper, S.(1993). A formal approach to the comparison of object-oriented analysis and design methodologies. In *The Twenty Sixth Annual Hawaii International Conference on System Sciences*, pages 689-699, Hawaii.

Horling, B. and Lesser, V. (2004). A survey of multi-agent organizational paradigms', *Knowedgel. Engineering. Review*., Vol. 19, No. 4, pp.281–316.

IEEE Standards Collection: Software Engineering, IEEE Standard 610.12-1990, IEEE, (1993).

Iglesias, C., Garijo M., Gonzlez, J. & Velasco J.(1998). A methodological proposal for multi-agent systems development extending CommonKADS. In *Proceedings of the Tenth Knowledge Acquisition for Knowledge-Based Systems Workshop*.

Iglesias, C., Garijo, M. & Gonzalez, J.(1999). survey of Agent-oriented Methodologies.In: *Muller, J.P., Singh, M.P., Rao, A. (eds.): Intelligent Agents* V (Atal'98), LNAI 1555. Springer-Verlag, Berlin.

Iglesias, C., Garijo, M. & Gonzalez, J. (1992). A Survey of Agent-Oriented Methodologies. *In Intelligent Agents V - Proceedings of the Fifth International Workshop on Agent Theories, Architectures, and Languages (ATAL-98), Lecture Notes in Artificial Intelligence*. Springer-Verlag, Heidelberg.

Jacobson, I., Christerson, M., Jonsson, P., & Overgaard, G.(1992). Object-Oriented Software Engineering: A Use Case Driven Approach. *Addison-Wesley*.

Jacobson, G. Booch & J. Rumbaugh .(1999).The Unified Software Development Process – Addison-Wesley.

Jayaratna, N.(1994). Understanding and Evaluating Methodologies: NIMSAD a Systematic Framework. *McGraw-Hill, New York*, 2nd edition, 1994.

Jennings, N.R. 2000. Agent Oriented Software Engineering. Proc. 12th Int Conf on Industrial andEngineering Applications of AI, Cairo, Egypt.

Jennings, N.R., Wooldridge, M. J.(1999). Agent-Oriented Software Engineering. In *Handbook of Agent Technology*.1999.

Jorge, J. Gomez-sanz &Ruben, F.(2015). Understanding Agent-Oriented Software Engineering methodologies. *The Knowledge Engineering Review*, Vol. 30:4, 375–393. © Cambridge University Press, 2015 doi:10.1017/S0269888915000053.

Juan, A. , Carles, S. , Josepli, A, Maite, L. & Inmaculada, R.(2015). Towards next generation coordination infrastructures. *The Knowledge Engineering Review*, Vol. 30:4, 435–453.© Cambridge University Press, doi:10.1017/S0269888915000090.

Juan, T., Pearce, A. & Sterling, L. (2002). Extending the Gaia Methodology for Complex Open Systems, *Proceedings of the 2002 Autonomous Agents and Multi-Agent Systems*, Bologna, Italy, July 2002.

Khanh H. D., (2003). Evaluating agent-oriented software engineering methodologies, *master's thesis, School of Computer Science and Information Technology*, RMIT University, Melbourne, Australia.

Kendall, E. A., Malkoun, M. T. and Jiang, C. H. (1995). A methodology for developingagent based systems. In *Chengqi Zhang and Dickson Lukose, editors, First Australian Workshop on Distributed Artificial Intelligence*.

Kendall, E. A.(2000). Agent Software Engineering with Role Modelling. *Part of the Lecture Notes in Computer Science book* series (LNCS, volume 1957) - Springer In this volume (2000).

- Kinny, D., Georgeff, M., Rao, A.(1996). A Methodology and Modelling Technique for Systemsof BDI Agents. Agents Breaking Away: *Proceedings of the Seventh European Workshop on Modelling Autonomous Agents in a Multi-Agent World, MAAMAW '96. Lecture Notes in Artificial Intelligence*, Vol. 1038. Springer-Verlag, Berlin Heidelberg (1996) 56-71.

Kinny, D., George, M.(1996). Modelling and design of multi-agent systems. *In Intelligent Agents III: Proceedings of the Third International Workshop on Agent Theories.*

Kitchenham, B.(1996). DESMET: a method for evaluating software engineering methods and tools. *Technical Report TR96-09, University of Keele, U.K.,* August 1996.

Kruchten, P.(2000). The Rational Unified Process: An Introduction. Addison-Wesley Pub Co, 2nd edition.

Law,D. (1988). Methods for Comparing Methods: Techniques in Software Development, NCC Publications. multi-agent systems *The Knowledge Engineering Review*, Vol. 30:4, 394–418. © Cambridge University Press, 2015 doi:10.1017/S0269888915000077.

Lind ,J. (1999). MASSIVE: Software Engineering for Multi-agent Systems. *PhD thesis*, University of Saarbrcken, Germany, 1999.

Luck, M., McBurney, P. &Preist, C.(2003). Agent technology: Enabling next generation computing: A roadmap for agent-based computing. Agent Link report, *available from* [www.agentlink.org/roadmap](www.agentlink.org/roadmap), 2003. (Date Accessed 20-2-2008).

Maes, P. (1995). Artificial Life Meets Entertainment: Life Like Autonomous Agents. *Communications of ACM*, Vol. 38, No. 11, pp. 108-114, 1995.

Massawe, LV., Aghdasi, F., & Kinyua, J.(2009). The Development of a Multi-agent based Middleware for RFID Asset Management System using the PASSI Methodology.*IEEE Computer Society Proceedings OF The 2009 Sixth International Conference ON Information Technology: New Generations*, VOL. 1-3.Pages: 1042-1048.

Mathieson,I., Dance,S., Padgham,L., Gorman,M. & Winikoff,M.(2004). An open meteorological alerting system: Issues and solutions. In *Vladimir Estivill-Castro, editor, Proceedings of the 27th Australasian Computer Science Conference*, pages 351–358, Dunedin, New Zealand,2004.

Mefteh, W., Migeon, F., Gleizes, MP.& Gargouri, F.(2015). ADELFE 3.0 Design, Building Adaptive Multi Agent Systems Based on Simulation a Case Study. *Computational Collective Intelligence (ICCCI 2015), PT I Book Series: Lecture Notes in Artificial Intelligence* Volume: 9329 Pages: 19-28. Muller, P.1997. Instant UML, Wrox Press, Birmingham,UK, 1997.

Moulin, B., Cloutier, L.(1994). Collaborative Work Based on Multi-Agent Architectures: A methodology Perspective. In: *Aminzadeh, F., Jamshidi, M. (ed.): Soft Computing: Fuzzy logic,Neural Networks and Distributed Artificial Intelligent.* Prentice-Hall, N.J., USA ,1994.

Munroe,S.,Miller,T.Belecheanu,R.A.Pˇechouˇcek,M.,McBurney,P.&Luck,M.(2006). Crossing the agent technology chasm: Lessons, experiences and challenges *in commercial applications of agents, Knowledge Engineering Review* 21 (4) (2006) 345–392. doi:10.1017/S0269888906001020.

Mylopoulos, J., Castro, J, & Kolp, M.(2000). Tropos: Toward agent-oriented information systems engineering. In *Second International Bi-Conference Workshop on Agent-Oriented Information Systems* (AOIS2000), June 2000.

Nguyen, C.D., Perini,A., Tonella, P. (2008). Experimental evaluation of ontology-based test generation for multi-agent systems, in: J.J. Gomez-Sanz, M., Luck (Eds.), *Ninth International Workshop on Agent-Oriented Software Engineering*, AOSE, 2008, pp. 165–176.

Nicholas, R., Jennings, N.R. & Wooldridge, M. J.(1998). Applications of intelligent agents. In *Nicholas R. Jennings and Michael J. Wooldridge, editors, Agent Technology: Foundations, Applications, and Markets*, pages 3-28. Springer- Verlag Heidelberg, Germany, 1998.

Nunes, I., Kulesza, U., Nunes, C., de Lucena, CJ.P.&Cirilo, E.(2009). A Domain Analysis Approach for Multi-agent Systems Product Lines. *Enterprise Information Systems-BKBook Series: Lecture Notes in Business Information* ProcessingVol.: 24. Pages: 716.

Nwana, H.S.1996 .Software Agents: An Overview, *Knowledge Engineering Review*, Vol. 11, No. 3, pp. 205-244.

Odell, J., Parunak, H. and Bauer, B. (2000) 'Representing agent interaction protocols in UML', Proc. of the *1st International Workshop on Agent-Oriented Software Engineering* (AOSE 2000), pp.121–140.

Odell, J., Parunak, H. and Bauer, B. (2001) 'Extending UML for agents', in Wagner, G., Lesperance, Y. and Yu, E. (Eds.): *Proc. of the Agent-Oriented Information Systems Workshop* (AOIS), Austin, 2000, pp.3–17.

Odell, J.(2002). Objects and agents compared. *Journal of Object Technology*, 1(1):41-53.

-OMG. (2003). Unified Modeling Language Specification. Version 1.5.

O'Malley, S. & DeLoach, S.(2001). Determining when to use an agent-oriented software engineering methodology. In *Proceedings of the Second International*

*Workshop On Agent- Oriented Software Engineering (AOSE-2001)*, pages 188-205, Montreal, May 2001.

-Omicini,A.( 2001). SODA:Societeis and Infrastructures in the Analysis and Design of Agent-Based Systems. In: Ciancarini, P., Wooldridge, M. (eds.): Agent Oriented Software Engineering,LNAI. Springer-Verlag. Berlin.

Padgham,L. & Winikoff, M.(2002). Prometheus: A methodology for developing intelligen agents. In *Third International Workshop on Agent-Oriented Software Engineering*, July 2002.

Padgham, L. & Winikoff, M. (2005). Prometheus: a practical agent-oriented methodology. In *Agent-Oriented Methodologies, Henderson-Sellers*, B. & Giorgini, P. (eds), Idea Group Publishing, Ch. V. 107–135.

Padgham, L., Thangarajah,J.& Winikoff, M. (2006). Tool support for agent development using the Prometheus methodology *in: Quality Software, 2005. (QSIC 2005). Fifth International Conference on 23 January.*

Padgham, L., Winikoff ,M., DeLoach, S., Cossentino, M.(2008). A unified graphical notation for AOSE, in: M. Luck, J.J. Gomez-Sanz (Eds.), *Proceedings of the Ninth International Workshop on Agent Oriented Software Engineering*, Estoril, Portugal, 2008, pp. 61–72.

Pavón, J., Gómez-Sanz, J. J. & Fuentes, R. (2005). The INGENIAS methodology and tools. In *Agent-Oriented Methodologies, Henderson-Sellers*, B. & Giorgini, P. (eds), Idea Group Publishing, Ch. IX. 236–276.

Perini, A.& Susi, A.(2004). Developing a decision support system for integrated production. *in agriculture. Environmental Modeling & Software*. Vol. 19.Issue: 9.Pages: 821-829.

Philippe,K.(2000). The Rational Unified Process: An Introduction. Addison-Wesley PubCo, 2nd edition.

Russell, S.J. and Norvig, P. (2003) Artificial Intelligence: A Moder Approach, *2nd ed., Prentice-Hall, Upper Saddle River*, NJ.

-Russell, S.J., Norvig, P.(2002). Artificial Intelligence: A Modern Approach. (*2nd ed.). Prentice Hall: Upper Saddle* River, NJ.

Saba, G.M., Santos, E.(2000).The Multi-Agent Distributed Goal Satisfaction System. *Submitted to International ICSC Symposium on Multi-Agents and Mobile Agents in Virtual Organizations and E-Commerce* (MAMA'2000).

Sabas, A., Badri, M., & Delisle, S. (2002). A multidimensional framework for the evaluation of multi-agent system methodologies. *Paper presented at the 6th World Multi-conference on Systemics, Cybernetics and Informatics* (SCI-2002), Orlando, Florida.

Saminni, FD., Tang,WH. & Wu, QH. (2012). Implementation of Gaia Methodology for Multi-agent based Transformer Condition Monitoring. *IEEE PES Innovative Smart Grid Technologies Conference Europe*.

-Shehory,O. & Sturm,A. (2001). Evaluation of modeling techniques for agent-based systems.In *Jörg P. Müller, Elisabeth Andre, Sandip Sen, and Claude Frasson, editors, Proceedings of the Fifth International Conference on Autonomous Agents,* pages 624-631. ACM Press, May 2001.

Siau, K., Cao, Q.(2001). Unified Modeling Language: A Complexity Analysis, *Journal of Database Management*, Vol. 12, No. 1, pp. 26-34, March 2001.

Siau, K., Rossi, M.(1998). Evaluation of Information Modeling Methods – A Review, in *Proc. 31 Annual Hawaii International Conference on System Science*, pp. 314-322, January 1998.

Spanoudakis, N. & Moraitis, P.(2009). Engineering an agent-based system for product pricing automation. *View ResearcherID and ORCID Engineering Intelligent Systems For Electrical Engineering and Communications* Vol.: 17 Issue: 2-3Pages: 139-151.

Spanoudakis, N.& Moraitis, P.(2009). Automated Product Pricing Using Argumentation. *View ResearcherID and ORCID. Artificial Intelligence Applications and Innovations III. Book Series*: International Federation for Information Processing. Pages: 321.

Spanoudakis N., Moraitis P.(2007). The Agent Systems Methodology (ASEME): A Preliminary Report. at: *https://www.researchgate.net/publication/249662978*.

Spanoudakis N.(2011). A Method Fragment for Transforming Gaia or ASEME Liveness Formulas to BPMN Models for Simulation. *International Workshop on Engineering Multi-Agent Systems. Turkey, May 5, 2011, Revised, Selected, and Invited Papers* .

Spanoudakis, N.I., Moraitis, P.(2008). The agent modeling language (amola).*In: Dochev, D., Pistore, M., Traverso, P. (eds.) AIMSA* 2008. LNCS (LNAI), vol. 5253, pp. 32– 44. Springer, Heidelberg (2008).

- Spanoudakis, N.: The Agent Systems Engineering Methodology (ASEME).(2009). *Ph.D. thesis, Paris Descartes University* (2009).

-Spanoudakis, N. I. & Moraitis, P. (2011). Using ASEME methodology for model-driven agent systems development. In *AOSE XI, Weyns, D., Gleizes, M. P. (eds), LNCS, 6788, 106–127*. Springer.

Sukhvir,S.,Prachi& Richa,S.(2012). Evaluation of Agent Oriented Software Engineering (AOSE) Methodologies-A review. *International Journal of Latest Research in Science and Technology* Vol.1,Issue 2 :Page No.94-97 , July .August.

Surya, B., Ralph, R., Akemi,T. & Rajkumar, M.(1988). Comparison of analysis techniques for information requirement determination. *Communications of the ACM*, 31(9):1090-1097, 1988.

-Szolovits, P., Doyle, J. & Long, W.J.(1994) Guardian Angel:Patient-Centered Health Information Systems, Technical Report MIT/LCS/TR-604,1994. http://www.ga.org/ga/manifesto/GAtr.html.

The National Science Foundation under grant #IIS-0083127.(2005). Agent-Oriented Software Engineering"Research Area Examination".

Tran, Q. , Low, G. & Williams, M.(2003). A Feature Analysis Framework for Evaluating Multi-agent System Development Methodologies, *School of Information Systems,Technology and Management The University of New South Wales New South Wales, Australia*.

Tveit, A. ( 2001). A survey of Agent-oriented software engineering. First NTNU CSGS.

Vliet, H. (2000). Software Engineering: Principles and Practice. John Wiley & Sons,second edition.

Walker, I.(1992). Requirements of an object-oriented design method. *Software Engineering Journal*, pages 102-113, March 1992.

Wand, Y., Weber, R.(1990). An Ontological Model of an Information System, *IEEE Trans. on Software Engineering*, Vol. 16, No. 11, pp. 1282-1292, November 1990.

Wautelet, Y.,Achbany, Y., Lange, JC. & Kolp, M.(2009). *A Process for Developing Adaptable and Open Service Systems: Application in Supply Chain Management*. Enterprise Information Systems-BK. Lecture Notes in Business Information Processing.Vol.24. Pages: 564-576.

Weiss, G.(2002). Agent oriented Software engineering. *Knowledge Engineering review*, Vol. 16(4).

Wooldridge, M.& Jennings, N. R.(1995). Intelligent agents: Theory and practice. *The Knowledge Engineering Review*, 10(2):115-152,1995.

Wooldridge, M., Jennings, N.R.,& Kinny, D.(2000) The Gaia methodology for agent-oriented analysis and design. *Autonomous Agents and Multi-Agent Systems*, 3(3), 2000.

Wooldridge, M.& Jennings, N. R. (1999). .A methodology for agent-oriented analysis and design.. AGENTS '99Proceedings of the third annual conference on Autonomous Agents.

Wood ,M. F. & DeLoach, S. A.(2000). An Overview of the Multi-agent Systems Engineering Methodology, *First International Workshop on Agent-Oriented Software Engineering* (AOSE-2000), Limerick, Ireland, June.

Wood, M. F.: Multiagent Systems Engineering: A Methodology for Analysis and Design of Multiagent Systems.(2000). MS thesis, AFIT/GCS/ENG/00M-26. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB Ohio, USA (2000).

Yu, E, (1995). Modelling Strategic Relationships for Process Reengineering. *PhD thesis, University of Toronto, Department of Computer Science*.

Zambonelli, F., Jennings, N. R. & Wooldridge, M. (2005). Multi-agent systems as computational organizations: the Gaia methodology. In *Agent-Oriented Methodologies, Henderson-Sellers, B. & Giorgini, P. (eds), Idea Group Publishing*, Ch. VI. 136–171.