RIGA TECHNICAL UNIVERSITY

Faculty of Computer Science and Information Technology

Institute of Applied Computer Systems

Himanhsu Prakashbhai Patel

Student of the academic master study programme „Computer Systems" (Student ID 151ADM047)

EVALUTION OF METHODOLOGIES FOR EFFICIENT MULTI-AGENT SYSTEM PROJECTS

Master Thesis

Scientific supervisor

Dr.sc.ing., professor

Jānis Grundspeņķis

Riga 2017

THESIS ACCOMPLISHMENT AND EVALUATION

The Master Thesis is developed in ***the Institute of Applied Computer Systems.***

I certify with my signature that all information sources used are included in the list of Bibliographies and the thesis is original.

The author of the thesis:

Stud: **H.P. Patel**...……………………….......……………………………......

(signature, date)

The Master thesis is recommended for defence:

Scientific supervisor:

Dr.habil.sc.ing., Prof: **Jānis Grundspeņķis**……...............………………………………...

(signature, date)

The thesis is approved for defence:

Head of the Institute of Applied Computer Systems:

Dr.sc.ing., Prof.:**J. Eiduks**..............…………………………………...

(signature, date)

The bachelor thesis was defended at the meeting of the graduation examination commission of the Institute of Applied Computer Systems in

……... ……………....…evaluated with the mark ( )…....…………………..

(year, month, date)

The secretary of the graduation examination commission of the Institute of Applied Computer Systems

……...................…………

(surname, signature)

ABSTRACT

The main aim of the thesis I am presenting here, is in investigating the stretch and nature of the methodologies that have evolved in areas of Multi agent Systems, focusing on the theory, analysis, methodologies, technologies and tools used for design and development of the MAS projects as applicable to the Industrial revolution that is happening in IT, Cloud, Artificial Intelligence, Industrial automation, robotics and space programs. My research will further add value of several findings that has not been compiled in one place earlier like applicability of these methodologies for project work and ability in choosing right methodology for a specific project.

The use of MAS projects is growing at an alarming rate and there is great potential for growth and advances in this area; this is the main purpose to write this thesis to showcase how the MAS evolution is taking place and what types of projects can be leveraged for the benefit of the Industrialist and Company owners .This research is based on the industry standards related to AI and robotic engineering and the methods used are as per the best practices specified for this course and university guidelines. The findings underline that there is an increased use of the MAS projects since several years when there was a need for autonomous, self-decision making systems, self-correction and less human intervention and feedback to measure, control and govern the system. My current thesis has demonstrated the evolution and the necessary data depicted in form of charts and graphs to highlight the evolution and the rate of growth in this sector. The key conclusion that can be drawn out of this work is that MAS projects are being adopted in many large-scale projects worldwide and for effective project implementation there is a need for improved awareness, better knowledge of tools, templates, methodologies that will help the projects in effective implementation, management and delivery of the return of investments.

This thesis has tried to highlight the need of effective methodologies and there is ample scope for conducting research and develop newer and better methodologies while working on MAS projects. Transition from theory to hypothesis and to project implementation should be smooth, well documented and mature enough to meet the needs of the modern industry.

The thesis contains 92 pages,54 figures and 9 tables, 53 references, and 0 appendices.

Key words: Agents, Multi Agent system(MAS), Agent development methodologies, Agent-oriented software engineering (AOSE), Agent types, features and benefits, Belief-Desire-Intention model(BDI), Agent Architectures, CommonKADS, AUML, Prometheus methodology

Tropos methodology, O-MaSE methodology, GAIA methodology, INGENIAS, MAS-CommonKADS, Methodology for BDI agents, Comparative analysis of the methodologies.

ANOTĀCIJA

Galveinais mērķis darbam, ko es prezentēju šeit, ir metodoloģijas, kas ir attīstījušās Multi Aģentu Sistēmu jomās (turpmāk tekstā - MAS), platības un dabas izpēte, fokusējot uzmanību uz teorijas, analīzes, metodoloģijas, tehnoloģijas un instrumentiem, kurus izmanto MAS projektu attīstībai un projektēšanai, kas piemērojami rūpniecības revolūcijai, kura notiek tādās jomās kā IT, Cloud, Mākslīgais intelekts, rūpnieciskā automatizācija, robotika un kosmosa programmas. Mani pētījummi vēl pievienos vērtību dažādiem secinājumiem, kas agrāk nebija apkopoti vienā vietā, piemēram šo metodoloģiju pielietojamība projektu darbam un prasme izvēlēties pareizo metodi konkrētam projektam.

MAS projektu izmantošana pieaug satraucošā ātrumā un šajā jomā ir liels izaugsmes un progresa potenciāls; rakstot šo tēzi, mans galvenais mērķis bija parādīt kā MAS attīstība notiek un kāda tipa projekti var būt izmantoti rūpnieku un uzņēmumu īpašnieku labā. Šis pētījums ir balstīts uz nozares standartiem, kas saistīti ar AI un robotu inženierzinātni un izmantotās metodes pilnīgi atbilst šī kursa universitātes prakses metodiskiem norādījumiem. Atklājumi pasvītro, ka ir palielinājusies MAS projektu izmantošana kopš tā laika, kad palika nepieciešamība autonomās, sevis lēmumu pieņemošās, sevis koriģējošās sistēmās un mazāk cilvēku iejaukšanās, kā arī palika nepieciešama atgriezeniskā saite, lai novērtēt, kontrolēt un regulēt sistēmu. Mans pašreizējais darbs parādīja attīstību un nepieciešamību attēlot datus diagrammās un grafikos lai izcelt attīstību un izaugsmes ātrumu šajā nozarē. Galvenais secinājums, ko var izdarīt pēc šī darba ir tas, ka MAS projekti tiek pieņemti daudzos liela mēroga projektos visā pasaulē, un efektīvai projektu īstenošanai ir nepieciešams: uzlabot izpratni, labākas zināšanas par rīkiem, veidnes, metodes, kas palīdzēs projektu efektīvai īstenošanai, darba organizācija un ieguldījumu atgūšanu piegāde.

Šajā tēze ir uzsvērta efektīvas metodoloģijas nepieciešamība, un ka pastāv plašas iespējas veikt izpēti un attīstīt jaunu un labāku metodiku, strādājot ar MAS projektiem. Pārejai no teorijas uz hipotēzi, un uz projekta īstenošanu, jābūt gludai, labi dokumentētai un pietiekami nobriedušai, lai apmierinātu mūsdienu nozares vajadzības.
Darbā ir 92 lappuses, 54 attēli,9 tabulas, 53 norādes un 0 pielikumi.

Atslēgas vārdi: Aģents, Multi Aģentu sistēmas (MAS), izstrādes metodoloģijas aģents, aģentu orientētas programmatūras inženierija (Agent-oriented software engineering - AOSE), aģentu tipi, īpašības un priekšrocības, Ticības- Vēlēšanās-Nodomu modelis (Belief-Desire-Intention model - BDI), Aģentu Arhitektūras, CommonKADS, AUML, Prometheus metodoloģija, Tropos metodoloģija, O-MaSE metodoloģija, GAIA metodoloģija, INGENIAS, MAS-CommonKADS, metodoloģija priekš BDI aģentiem, Salīdzinošā metodiku analīze.

ACRONYMS

MAS – Multi Agent System

AOSE- Agent oriented software Engineering

BDI – Belief Desire Intention

AI – Artificial Intelligence

UML – Unified Modeling language

AUML – Agent UML

OOP - Object Oriented Programming

AOP - Agent-Oriented Programming

OO - Object–Orientation

KE - Knowledge Engineering

RE - Requirements Engineering

PDT - Prometheus Design Tool

GMoDS - Goal Model for Dynamic Systems

SD - System Description

SRS - Systems Requirement Specification

AFIT - Air Force Institute of Technology

CA - Communicative Act

AIP - Agent Interaction Protocol

IDK - INGENIAS Development Kit

RUP - Rational Unified Process

MDD – Model Driven Development

IDP - INGENIAS Development Process

OMT - Object Modelling Technique

RRD - Responsibility Driven Design

UER - User-Environment-Responsibility

MSC - Message Sequence Charts

BDI – ASDP - BDI Agent Software Development Process

JADE – Java Agent Development Environment

JADEX. – Belief Desire Intention (BDI) reasoning engine

ECLIPSE – Java based IDE for Agent development

JACK - Belief Desire Intention (BDI) agent development environment

SDLC – Software Development Life Cycle

XML –Extensible Markup Language

DB- Database

CMS - Conference Management System

GUI – Graphical User Interface

IT – Information technology

FIPA - The Foundation for Intelligent Physical Agents

OMG - Object Management Group

CONTENTS

1 INTRODUCTION

The main inspiration in choosing the topic — "Evaluation of methodologies for efficient multi-agent system projects" was the challenge I faced when I set out to find how many companies use and believe in "Multi Agent System abbreviated as MAS." (Garcia-Ojeda, and DeLoach, 2017) I have attempted to find fortune companies that use MAS on a large scale and as a domestic service. To my surprise, I was not able to gather enough information on agent development by the Industry except for few military and space programs. I was not happy with the type of information and data available in the market. This failed my attempt to obtain information on the methodologies used for MAS.

1.1 Problem Statement

The problem that I am going to address in this thesis is that the challenges faced (theoretical, methodology, tools and proven AI project techniques) by IT industry in developing AI based MAS projects and how these can be overcome by way of systematic approach and methodologies.

Choosing a right MAS type and methodology is a challenge for each of the projects and this requires a systematic work from scratch. This thesis will try to resolve such issues in areas of MAS methodologies.

1.2 Background and Need

New challenges are being faced by the software industry and paradigm shifts that need to adapt to the newer mission critical business needs. Modern age businesses require applications that can operate autonomously and remain competitive. The enterprises need to align their structure more robustly to the enterprise vision with less human intervention, more automation and systems that can adapt to dynamic, agile environments, configurations and those that can interact with other enterprise applications for realistic solutions. As truly noted by Garcia "Multiagent system (MAS) technology is a promising approach to such new business requirements and commands a different way of designing and developing applications." (Garcia-Ojeda and DeLoach, 2017) "Its central notion – the intelligent agent –encapsulates all the characteristics (i.e., autonomy, proactive, reactivity, and interactivity) required to fulfill the requirements demanded by such newer applications." (Garcia-Ojeda and DeLoach, 2017)

Traditional application development methodologies (SDLC) have been leveraging the waterfall and iterative models for quite a long time. Moreover, the uses of custom home grown methodologies and non-standardized processes have made the project life cycle look like a long wavy road ahead. Conditions go bad as key sponsors give hard limits for timely, economic releases due to limited budgets and time of end clients. The need for standardized process and implementation methodology has been in great demand to control project expenses and deliver a quality product.

To match with the huge growing IT needs, projects need to be more realistic and it also applies to Artificial Intelligence domain. Apart from being a research area, Artificial Intelligence cannot remain stagnant and it needs to be utilized in the industrial and commercial applications for domestic purposes and projects. Hence there is a need for education and awareness of the MAS methodologies that help in resolving the issues currently faced by IT Industry

As Michael Wooldridge said, "Until researchers don't recognize that agent-based systems are all of computer science and software engineering more than AI, then within few years, we may be asking our self why agent technology suffered the same destiny as other AI ideas that looked good in theory." (Wooldridge, 1997) (Mohire, 2012)

As per Kephart and Chess "Within the modern age applications, many of them will be biologically inspired: self-organizing sensors based networks, allowing control of airborne vehicles, or of dangerous remote areas; and also, storage facilities, or operating systems facilities, that, work as human nervous system, control in functionalities transparently." (Kephart and Chess, 2003) (Mohire, 2012)

Looking at the above quotes, there is a need for clear understanding and better value creation of the AI and MAS in real products. This thesis will attempt to highlight the evolutions of methodologies used for MAS project and the pros and cons of these methodologies.

1.3 Aims of the study

The key aim of this thesis is in "identifying and evaluating the types of MAS and MAS methodologies used for MAS project development using systematic analysis, research and references". It focuses mainly on comparing the MAS types and methodologies, their origin, key strengths, limitations and purposes. Another aim is to "lay a strong foundation work that can be beneficial for MAS project team in gaining insight, confidence and become aware of the limitations involved in developing the MAS system". This thesis will help them to decide and use a methodology that best fits the project needs. This thesis can be considered as a baseline reference document before taking up MAS based projects and before demonstrating the features, benefits and values to win the contract.

The following objectives have been acknowledged as of top importance to achieve the best results:

1.4 The area of research

The area of research I have proposed is related to "Artificial Intelligence (AI) based Multi Agent System (MAS)" and their related methodologies.

In particular, my area of research is related to the methodologies, their evolution and how these methodologies provide value to project work and how these compare to each other.

1.4.1 The most appropriate previous finding in this area

During my MS term at Riga Technical University, I have researched and found that several agent oriented methodologies have been proposed by famous scholars based on a variety of concepts, techniques, notations, and methodological principles. Few of these rely on standard methods and or modelling languages like "CommonKADS and UML." (AUML, 2007) "The MAS CommonKADS and AUML." (AUML, 2007) extended "CommonKADS and UML" (AUML, 2007) respectively to meet the multi-agent systems. I also have researched on several articles written by scholars who have dedicated their time on similar thesis area, their details can be seen in the references sections under the names like "Wood, Wooldridge, Russell, Chess, Lin, Rumbaugh and Fausto" to name a few. Few of these have introductions to methodologies like TROPOS, Prometheus, GAIA, O-MaSE and AUML. More details of my research are provided in the following Chapters.

1.4.2 My research problem and why this is worthwhile studying

The research problem I am going to address in this thesis is that the challenges faced (theoretical, methodology, tools and proven MAS project techniques) by IT industry in developing AI based MAS projects. Choosing a right MAS type and methodology is a challenge for each of the projects in MAS that requires a systematic work from scratch. This thesis will try to resolve such issues in areas of MAS methodologies by way of compiling most popular methodologies and their pros and cons all documented in one document.

1.4.3 Target group

The target group of this thesis is a broad set of users like MAS project designer, architects and development team who wish to know about MAS and have a complete picture of the methodologies that they can rely upon and adopt them in their projects without spending much time doing self-research. This document can also be useful for other research scholars and IT staff who want to get a general awareness of the Agent development methodologies. This can be used as a general reference for Master thesis writing and project work. This is a useful document that can benefit the academic world, professionals and IT team who are interested to work in MAS.

1.5 Research method in brief

My research method has been guided by my professor and as per RIGA TECHNIUCAL UNIVERSITY framework. It includes the following points:

- Literature review - Study of earlier work done in this area of AI and MAS
- Reference the library books at university – Visit to the University library and reference relevant books from research scholars and professionals
- Online research using Internet and AI community groups – Conduct search using suitable keywords related to Agent, MAS, visit professor’s websites and publications
- Personal interaction with scholars who have earlier worked on similar thesis – Interact with the scholars in this area

Apart from these I have used my own combination of study, interaction and participating in the AI community, trying to gain the first-hand information from authors and getting their participating in guiding me in this thesis writing.

1.6 Structure of The Report

Chapter 1: Introduction

This chapter is the premise of the thesis and introduces the research area, current problems and issues faced that are being addressed and the research process used in resolving such issues and adding value to the process. This chapter is the executive summary that provides an overall view of the problem at hand in agent development and how the methodologies can be chosen to resolve ambiguities in choosing the right methodology.

Chapter 2: Mas Methodologies

In this chapter I will introduce to various agent methodologies, their origins and the base category they belong to. I will explain the path these methodologies have taken and how they evolved over time morphing into the modern methodology that addressed various earlier limitations present in the legacy methodologies. These newer modern methodologies can be used for MAS development that is proposed today. I will provide overview of the SDLC phases used by various agent models and agent interactions diagrams as applicable to a MAS. I will enlighten and bring the SDLC phases where the methodology is used, key deliverables, process steps and the analysis, design, and code related activities involved in the methodology**.**

Chapter 3: Case Study

In this chapter, I will describe the real examples that have been used by customers in developing their products by using a specific methodology in their development environment. This chapter briefs about different ways the methodology can be incorporated into a specific domain and project work as part of the SDLC. It highlights the key areas where the methodology is effective and generates value to an application under production environments and its limitations that are visible to the end user.

Chapter 4: Comparative Analyses of The Methodologies

This is the key chapter of my research and findings with regards to the MAS methodologies. In this chapter, I will compare and contrast seven popular MAS methodologies that I have described in earlier chapters. I will use an industry framework to compare the key features related to agent oriented software development. I will provide and personalise the metrics to compare and show the strengths and limitations of each methodology. This will allow the developer to know more about the individual concepts.

Chapter 5: Research Results and Discussion

In this chapter I will try to put forth my key findings and provide know-how of my finding that can be used in various MAS project work by choosing a right methodology. A brief discussion and my personal observations will be described.

Chapter 6 Conclusions and Future Work

This chapter provides the concluding remarks of my research and study and paves a way for further researches who wish to research in this area of MAS. This is the concluding chapter of the thesis that will be followed by other sections like References and appendix.

2 LITERATURE REVIEW

After conducting a thorough research using the university recommended practices I am presenting the following history and evidences along with the related references. My due respect to all those scholars who have relentlessly made progress in this area, and have been able to provide the modern MAS methodologies to the AI world.

2.1 Introduction to Agents and Multi agent system (MAS)

When people talk about Intelligence systems they think about the AI world and agents. The common man has some general awareness of artificial intelligence (AI), robotics and how robotics make life easy by redoing the same job without any issue several thousand times that is not an easy task for the human being. Moreover, artificial intelligence is about adding sense and human live features to the ordinary robots. The main application that handles the robot is an agent that is intelligence, autonomous and can handle jobs with less human intervention, able to decide on its own and able to continue work from a point. Moreover, these agents are in line to the laws of computer systems, control logic and work in a controlled window that is closely monitored by humans.

The notation and concept about agent and artificial intelligence was first introduced in the 1950's when science fiction was becoming an actual research area where in the idea to visualize design and start implementation of robots similar to human behavioral activities.

"In modern time, there is no specific definition about AI however the definitions of Wooldridge and Jennings is increasingly being adopted by many AI scholars". (Lee, 2006)

As per my study, an agent is an intelligent system like computer system that has a fixed environment and is capable of taking specific actions according to changes in environmental situation to ensure design goal or objectives are met. As per Wooldridge thoughts he "distinguishes an agent and an intelligent agent, and classifies them as reactive, proactive and social. An intelligent agent is characterized as being autonomous, situated, reactive, proactive, flexible, robust and social". (Wood, 2000) (Wooldridge, 1997) (Padgham and Winkoff, 2005)

In words of Wood "An agent is a computer system which is positioned in an environment, and which is capable of taking autonomous action in the environment in order to achieve its design objective". (Wood, 2000)

Following Russell and Nerving "an agent is anything that can be viewed as perceiving its environment using sensors and acting upon that environment using actuators". Few examples are humans, robots, or software agents that mimic the words. We often use the term autonomous to refer to an agent whose decision-making relies on its own perception than to the prior-knowledge given to it at design time, and that can complete an assigned task with less human intervention". (Russell and Norvig, 2003)

Agents are entities that have modules similar to human mind, thinking like a human. They have beliefs and capabilities like human mind and they are able to choose, and commit to a purpose. Agents are fragments of program codes which can complete tasks autonomously. Each agent has autonomy as one of its property. It means the agent can operate without direct human intervention. Moreover, agents are able to interact with humans or other agents. This feature is the in-built social capability of an agent.

Reactivity and pro-activeness are another two important properties of agents. Reactivity is to feel the environment and respond to environmental changes in a timely manner; and pro-activeness means exhibiting goal-directed behavior. (Salah and Ashabi, 2014)

About agent origins, they have been instantiated from Object Oriented Programming (OOP) paradigm and the related paradigm is called Agent-Oriented Programming (AOP). AOP looks at the system as a set of modules that communicate with one another by treating the incoming and outgoing messages and by setting up the mental state of the agents to contain relevant components like as capabilities, beliefs, and decisions. Various limitations are provided on the mental state of an agent and these match to the constraints as per their intellectual state. When implementing agents as a system, they are capable of achieving sophisticated goals autonomously until a solution is found that completes the assigned goal.

The potential of system capability based on agents is large that can be harnessed to a great extent to benefit the science projects. The thriving future of agent-oriented systems lies on a complete and accurate analysis of its conceptual base, principles of structuring, and techniques used to design and develop agent systems. Hence, using a methodology is highly desired to support requirement changes and adding new components.

Also, the changing market requirements raise the need for new production planning models. This requires newer approaches for production lines and intelligent machines that ensure stability, sustainability at an economical price. It is important for system to be agile and react quickly with minimum risk to the sudden and unpredictable changes in requirements. Adaptive, reconfigurable and modular production systems address these requirements allowing machines and plants to adapt themselves to changing demands and interact with each other to fulfill the overall production goal, as stated in the Manufacture Strategic Research Agenda. (Commission, 2007)

Multi-Agent Systems (MASs) have been widely recognized as enabling technologies for designing and implementing next-generation of distributed and intelligent industrial automation systems driven by standard MAS methodologies. “Self-diagnostics and robustness that allow efficient continuing in operation even if a part of the system is down are other important properties related to agents.” (Kadera, 2015) All these mean a standardized MAS methodology is the need of hour that can enhance and reduce investment risks in industrial projects.

2.2 Types of agents

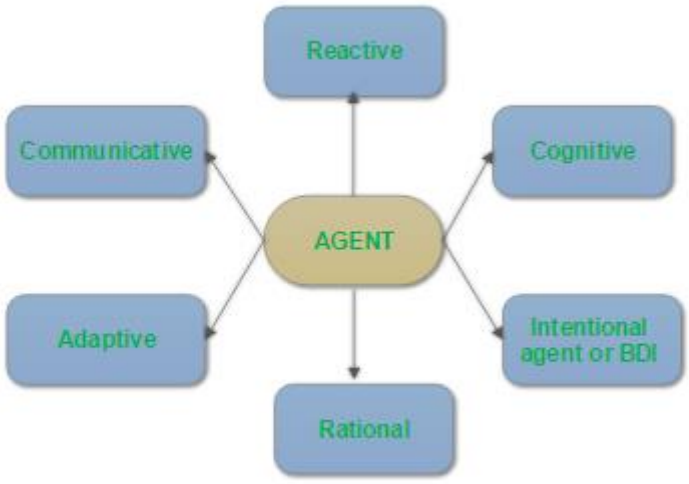


Figure 1 Types of agents (adopted from (Maalal and Malika, 2011))

I have researched and identified the following types of agents:

- “Reactive agent” (Maalal and Malika, 2011) which is a passive type of agent and often described as not being "clever". It is a very simple agent component that perceives the environment and acts based on an external input. It works only for stimulus-action which is a form of communication.
- “Cognitive agent” (Maalal and Malika, 2011) which is an active type of agent and is more intelligent than the reactive agent. It is primarily characterized by a figurative illustration of knowledge and/or mental theme. It has a fractional illustration of the environment and precise goals. It is capable of planning its behavior, remember its past actions, communicate by sending messages and negotiate with other agents.
- “Intentional agent or BDI (Belief, Desire and Intention) agent” (Maalal and Malika, 2011) which is an intelligent agent that uses human intelligence models and has similar human perspective of the world using mental concepts like knowledge, beliefs, intentions, desires, choices and commitments. Its behavior is explained by beliefs, desires and intentions.

- "Rational agent" (Maalal and Malika, 2011) which is an agent that takes action in a way that allows it to get the highest achievement in the task it was assigned. For this there should be a metric to measure the performance, and an objective associated with to a particular task against which the agent should run.

- "Adaptive agent" (Maalal and Malika, 2011) which is an agent that adapts as per changes to its environment. It is very intelligent because it is capable of changing its objectives and the knowledge base whilst it is required to adopt to its changing environment.

- "Communicative agent" (Maalal and Malika, 2011) which is an agent that is used to disseminate information to its surroundings. The information is usually data related to its own perceptions and related data transmitted by other agents.

Put in a different way I can say "An agent is anything that can create its own surroundings using sensors and takes actions in the its surroundings through effectors.

According to this definition I can say agents are:

- A human-like agent if I can interact like a human being and that has sensory parts like nose, tongue, skin and eyes.
- A robot which is a robotic agent that has so many advance functions like cameras and infrared range and motors and many electronic equipment.
- A software component which is also an agent that has preset bit strings as its programs and procedures.

In addition, as per my study agents can be further classified into three more type according to complexity.

Passive-agent (Kubera, Mathieu et al., 2010) that have no goals (like obstacle and destinations).

Active-agent **(**Kubera, Mathieu et al., 2010) with very simple goals (like in bird's flock, or sheep-wolf).

Cognitive-agent (Kubera, Mathieu et al., 2010) which contain complex calculations and codes and is the power house of knowledge and cognitive intelligence.

2.3 Agent Architecture and Models

Agents are architecture based on their types, intentions and key intention.

About Architectures, “Deliberative Architectures” (Wooldridge,1997) is one key architecture where agents retain an overt depiction of their world, that can be tailored by some form of representative reckoning.

“Reactive Architectures**”** (Georgeff and Lansky,1987) (Wooldridge,2009): The goal of this architecture is to build automatic mobile robots that can adopt several things from its environment, make them self-move freely in its environment without any internal connection. They can make decisions with very less amount of information. Decisions of such agents are based directly on the sensory inputs that are part of the agent part.

“Hybrid Architectures**”** (Ferguson, 1991) (Wooldridge, 2009): This is a combination of the classical and alternative approaches, and which combines the advantages of both of these kinds and also avoids the disadvantages.

“Layered Architectures” (Wooldridge, 2009): This architecture symbolizes the different subsystems that are arranged in a hierarchical layer, that involves the dissimilar decompositions within the agents.

About models, the “Belief-Desire-Intention (BDI) model is the most widespread model” (Rao and Georgeff, 1995). In simple words

Beliefs: Information about the agent environment.

Desires/Goals: Objectives or goal to be accomplished or finish.

Intentions: This is the currently chosen course of action as set by the agent.

Plans: This is the means of achieving certain future states related to the agent world.

Actions: This is the way an agent operates in the environment.

Few details of the BDI model in following section.

2.3.1 Belief-Desire-Intention (BDI) Model

The “Belief-Desire-Intention model is one important agent model. “It is based on human activity behavior, reasoning and intelligent decisions.” (Bratman, 1999) To ensure proper decision making by BDI agent there is a control mechanism to control intelligent actions. It is developed by Michael Bratman to describe future –directed intention. “The Belief-Desire-intention software model developed for intelligent agent coding and programming. A belief-desire-intention agent is a particular type of restricted rational software agent with specific architectural components” (Yang and Yokoo et al., 2009) shown below:

- Beliefs: Is the representation of the informational state of an agent, basically comprises of the agent believes and its understanding of the surrounding world.
- Desires: Is the representation of the motivational state of an agent. It represents the targets, objectives or situations the agent would like to accomplish or bring out. An agent might have a goal which can be a desire that is actively pursued by the agent.
- Intention: Is the representation of the deliberative state of an agent. It is what an agent chose to complete a goal or desire. It is also the sequence of actions according to the set plan.
- Events: Events are the decision or triggers for reactive activity performed by an agent. An event may change the beliefs, change and update goals or can be a decision or can trigger plans.

2.4 How agents Are Useful?

Agent reduces coupling and have loose coupling with other agents. Actually, coupling is possible through encapsulation provided by autonomy, and reactiveness and the robustness and pro-activeness of agents. The properties of an agent are stable and persist that is used in achieving a set goal or objective by trying other available approaches depending on effect of environmental change. “Active and reactive agents are similar to human way of dealing with problems”. (Lin and Winikoff, 2005)

Major advantages agents provide are:

- An agent can think independently and act on its own when it is not engaged with the rest of the surrounding system.
- An agent can be replaced or backed up by another agent; also, there can be multiple implementations of same agent using different languages that enhance fault tolerance.
- Agents can flexibly join or leave the system as necessary according to the demand (i.e., the system can be scaled and adapted automatically if necessary).
- Write once and run many time –a single instance of an agent can be reused multiple times in same or different environments.

Now it is time to provide the introduction to multi-agent-systems.

2.5 Multi Agent Systems

Definition for Multi agent system holds similarly to the definition for agent system.

From a software engineering perspective MAS is based on open architecture which allows new agents to dynamically join and leave the MAS system Agents act autonomously showing proactive behaviors not completely predictable a priori. (Henderson-Sellers, Brian et al., 2005)

Fig. 2 shows the elements of a multi-agent system. MAS comprises of two main elements: the "agents and the environment". (Alonso, Gómez et al., 2007) Agents exchange information based on certain organizational structure, patterns and rules. An agent can act alone

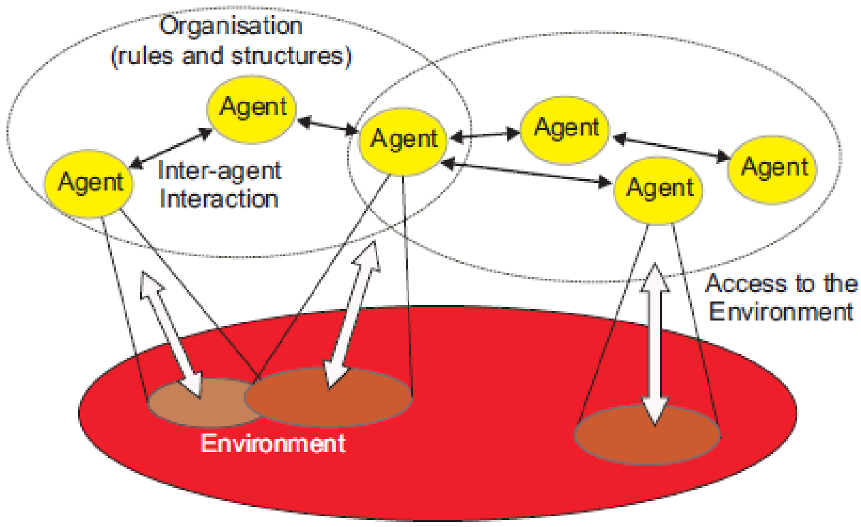


Figure 2 Elements and interactions in Multi-Agent System (adopted from (Alonso, Gomez et al., 2007))

without interaction with any other agent; also, agent can act in coalition with other agents ("by cooperating or negotiating or simply in a coordinated way". (Alonso, Gómez et al., 2007) in order to solve a problem from different parts. "Environment is where the agent lives" (Alonso, Gómez et al., 2007) (e.g. the Web, a database). Agents may or may not interact with the environment depending upon its goals and skills.

Fig 2. Elements and interactions in Multi-Agent System (adopted from Alonso, Gómez, et.al., 2007)

"A multi-agent system is a loosely coupled network of problem-solving entities (agents) that work together to find solutions to problems that are beyond the individual capabilities or knowledge of each entity (agent)." (Flores-Mendez, 2009)

In reality agents within a multi agent system work together as small cooperation among individual participating agents.

A MAS is considered as an independent system if all its individual agent completes its own goals alone without any communication with other agents. A MAS is said to be discrete if it is independent and if there are no issues in their goals or relationship with other agents and hence there is no cooperation required. "However, agents can cooperate with no intention of doing so and if this is the case then the cooperation is an emergent one. Cooperation is possible:

- by explicit design (the designer of the agents purposely designs the agent behaviors so that cooperation occurs).
- by adaptation (individual agents learn to cooperate).
- by evolution (individual agent's cooperation evolves through evolutionary process)." (Dr Glavic, 2006)

An agent is a computer structure that is located in a setting, and is capable of self-directed action in this setting to achieve its design objectives'. The work of Wooldridge points at the differences between agent and an intelligent agent. This is further categorized as be autonomous, reactive, proactive and social. The description of these as below:

- Autonomous**:** These agents are self-governing and formulate their own decisions without direct involvement of other agents or humans; such agents exerci**se** control over their actions and their internal state.
- Reactive**:** These agents are of reactive nature (not deliberate), responding in time to the changes in their environment.
- Proactive**:** This is an agent that seeks its goals over a period of time and takes the initiative when appropriate.
- Social**:** These agents frequently interact with other agents to finish their tasks and help other agents in achieving their goals. (Wooldridge, 1997)

One important concern in agent architecture is harmonizing reactiveness and reactiveness. On one side, an agent should be reactive so its plans and actions are influenced by environmental changes. On the other side, an agent plans and actions should be focused to its goals. The confront is to strike a stability between the two, that are often conflicting. If an agent is too reactive, it will be constantly making adjustments to its plans and hence likely not be successful in achieving its goals. Conversely, if the agent is not adequately reactive, it will then squander time by way of following plans that are not pertinent or applicable. "Agents tend to be used where the domain environment is challenging; more specifically, typical agent environments are dynamic, unpredictable and unreliable". (Padgham and Winkoff, 2005) In words of Padgham:

- Dynamic**:** These environments are of dynamic nature as they change rapidly. Therefore, an agent cannot presume that the environment shall remain static when it is trying to achieve a goal. (Padgham and Winkoff, 2005)

- Unpredictable: It is not improbable to predict the future states of the environment; as it is not probable for the agent to know the perfect and absolute information regarding their environment. (Padgham and Winkoff, 2005).
- Unreliable**:** The actions performed by agents might not succeed for reasons beyond an agent's control. (Padgham and Winkoff, 2005)

In conclusion, a multi-agent system indicates two or more agents interacting within an essential communication infrastructure that has no procedural control method; The individual agents are often distributed and are autonomous.

This concludes the chapter. Next I will focus on the Research on methodologies.

3 RESEARCH ON MAS METHODOLOGIES

3.1 Background

Despite the advancements in agent-oriented methods and technologies, nevertheless due to non-availability of sufficiently matured methodologies to steer the development processes, several cases have been observed where in a limited scale of adoption has happened to large

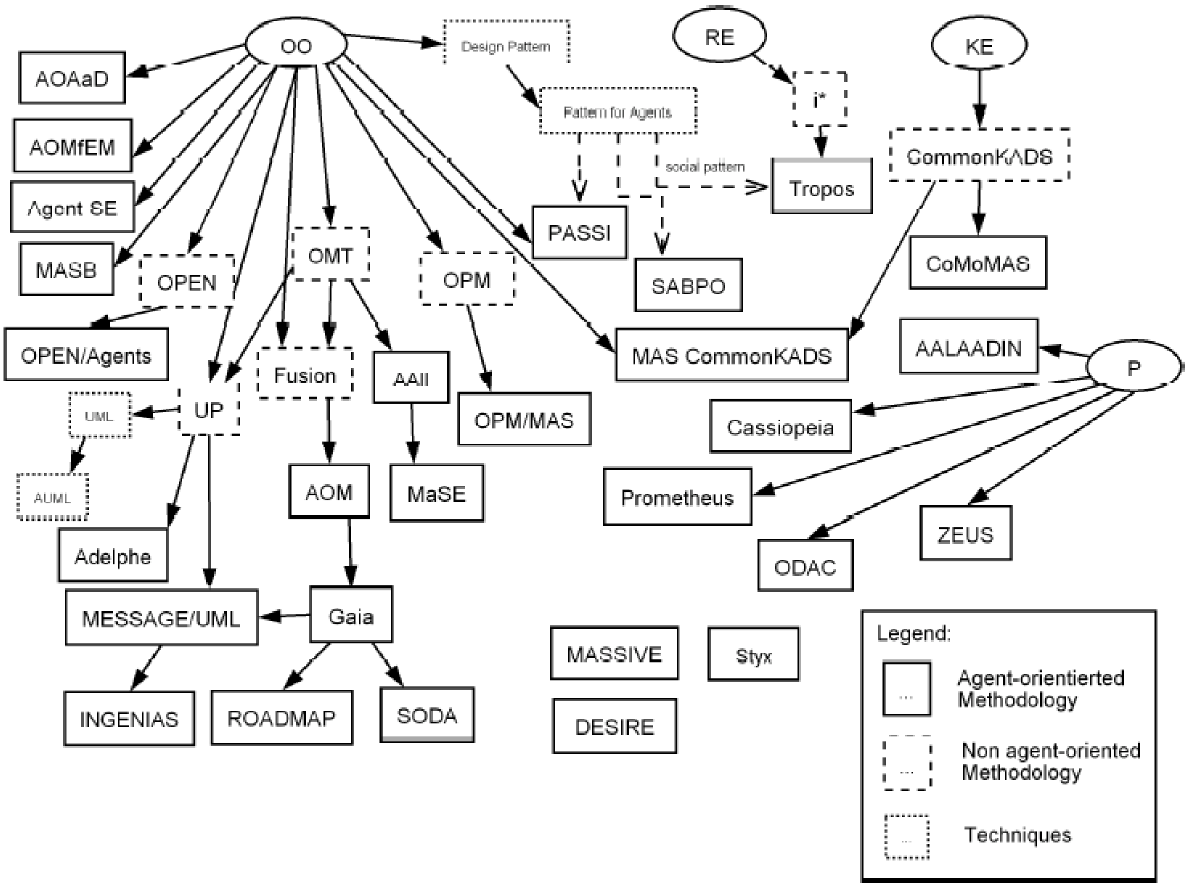


Figure 3 Genealogy of proposed methodologies (adopted from (Sudeikat, Braubach, et.al, 2005))

sized systems. So, the proper usage of agent-oriented methodologies plays a significant role in especially for large scale projects. Besides the need for reliable agent platform the dictate for a methodical development of applications is necessary. This can be addressed by a number of popular methodologies. "A methodology aids development through (1) guidance by a life cycle process, (2) a set of predefined techniques (guidelines, heuristics, etc.) and (3) allows modeling by providing a suitable notation." (Odell and Giorgini et al., 2005). These three elements have different advices meant for explicit purposes. This makes it complicated for organizations to choose a suitable one. The choice of the correct methodology is essential for the success of a project.

To address the above-mentioned issues diverse frameworks to compare has been proposed. These use aspect–based comparison of various criteria to identify the most suitable ones. However not any of the proposed frameworks consider the target platform. This is because different platforms mean different concepts in diverse situations.

My observations in this research bring forth interesting points on the origins of the agents. To portray these a few number of diverse methodologies have been showcased in a genealogy. This interesting genealogy is shown Figure 3. It illustrates the main influences on the individual methodologies that have evolved over time period and have been refined and improved by various scholars. My finding is that "Object–Orientation (OO), Knowledge Engineering (KE) and Requirements Engineering (RE) are the basic key ancestors." (Sudeikat and Braubach et al., 2005), which have been extended by agent programming abstractions.

Figure 3. Genealogy of proposed methodologies (adopted from (Sudeikat, Braubach et al., 2005))

3.2 Agent Development Methodologies

Numerous agent-oriented methodologies have been projected based on various notations, concepts, techniques and methodological guiding principles. A few of these have their base on standard modelling languages like the CommonKADS and UML. The "MAS CommonKADS

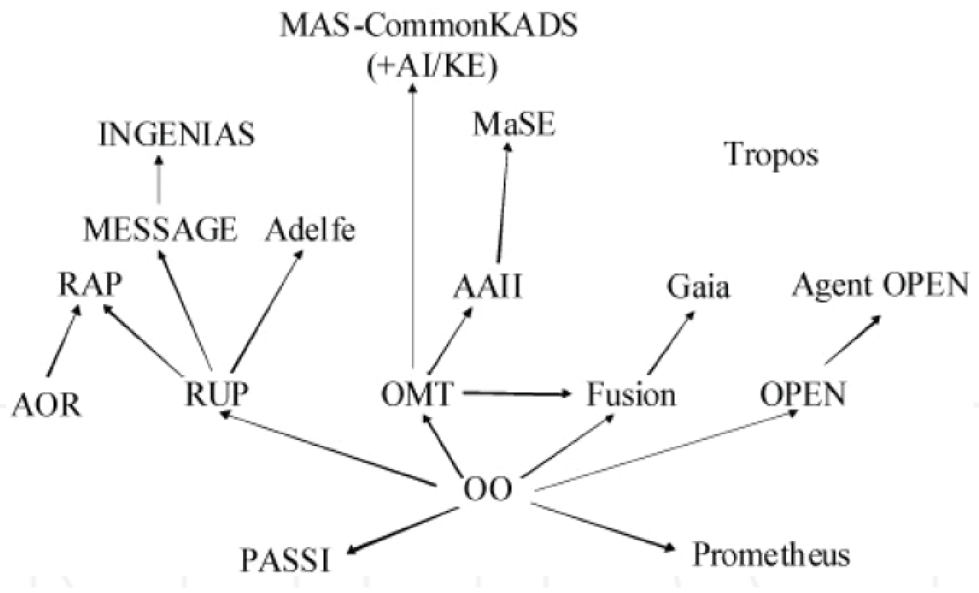


Figure 4 Influences of object-oriented methodologies on agent –oriented methodologies (adopted from (Werneck, Moreira Costa, et. al, 2011))

and AUML extended CommonKADS and UML respectively to meet the multi-agent systems." (AUML, 2007)

The agent-oriented methodologies have multiple roots (Figure 4). "Some are based on the idea of artificial intelligence coming from the knowledge engineering (KE). Other methods originate from software engineering and they are extensions of object-oriented (OO) paradigm.

There are still those that use a mix of concepts based on these two areas and some are derived from other agent-oriented methodologies." (Werneck and Moreira Costa, et al., 2011)

After introducing the origins, I will provide details of each the methodologies in the following sections

3.2.1 Prometheus Methodology

Prometheus methodology is an Agent-Oriented Software Engineering methodology (AOSE)" (Padgham, Thangarajah et al., 2007) used for the development of intelligent agents with details and not mere black boxes. It has three phases which are: (I) System Specification, (II) Architectural Design, and (III) Detailed Design" (Padgham, Thangarajah et al., 2007) This is shown in Figure 5.

Prometheus has all the required phases of software engineering, from requirements specification to detailed design and implementation. Every phase contains numerous models that capture the dynamics of the subsystems along with the whole system structure and components. As per LIN the phases are described as

- System specification: In this phase, the goals of the system are identified, the interface between the agents and their surrounding environment is documented (in terms of actions and percepts), roles are defined, and the exhaustive scenarios having sequences of steps are developed. (Padgham, Thangarajah et al., 2007)
- High-level design: In this phase, the agent types are described by merging the roles, the overall structure of the system is defined using a overview diagram, and interaction protocols are used to document the dynamics of the system in expressions of message sequences. (Padgham, Thangarajah et al., 2007)
- Detailed design: In this phase, the internals of each agents are developed in terms of their capabilities, events, plans and composite data. (Padgham, Thangarajah et al., 2007)

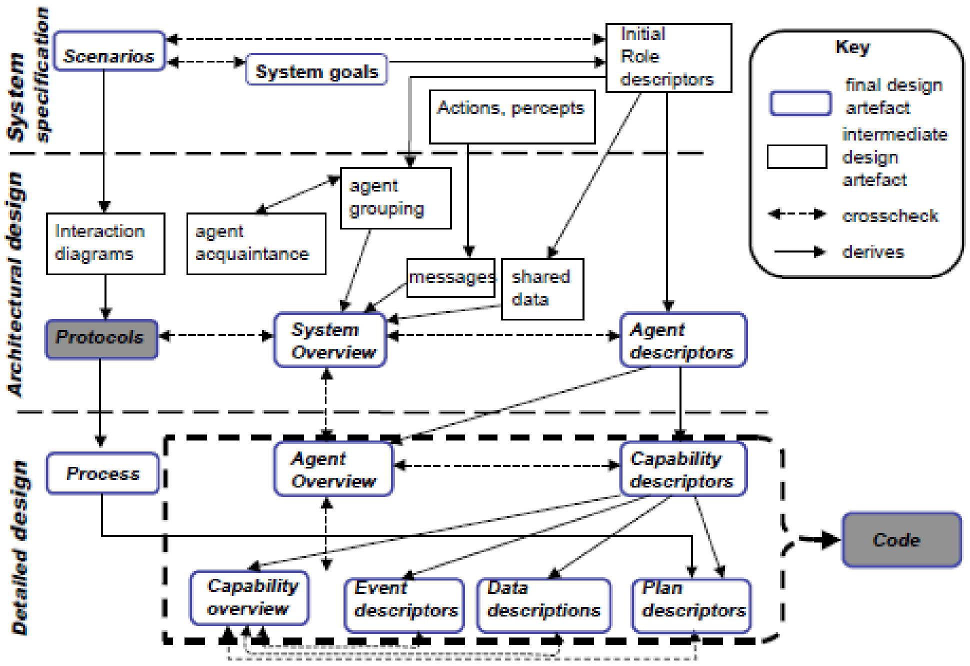


Figure 5 Prometheus Methodology (adopted from Padgham, Thangarajah, et.al 2007)

Prometheus Design Tool

The "Prometheus Design Tool – PDT" (Padgham, Thangarajah et al., 2007) is a freely downloadable tool, that runs under Java 1.5, which supports the Prometheus methodology. PDT has graphical interface for generating the design as per the methodological steps; With its help designer can edit diagrams and descriptors to generate entities and related artefacts. PDT enforces few constraints and checks the design for various consistency levels and conditions.

After the design has been developed, the design needs to be implemented. PDT has the feature to "generate skeleton code in the JACK agent-oriented programming language." (Padgham, Thangarajah et al., 2007). Also, the code generator extension of the PDT maintains synchronization among the code and design.

This concludes the brief overview of the Prometheus methodology. I will introduce Tropos.

Tropos Methodology

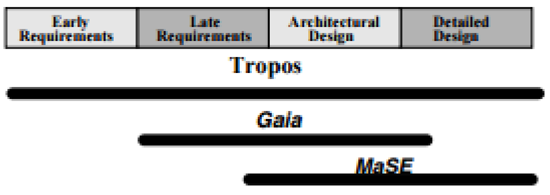


Figure 6 Comparison of Tropos with other software development methodologies (adopted from (Giunchiglia, Mylopoulos, et.al. 2003))

Tropos is a methodology that covers all the entire lifecycle of software development process (SDLC) activities, right from system analysis to system design and system implementation.

Tropos development process consists of five phases: (I) Early Requirements, (II) Late Requirements, (III) Architectural Design, (IV) Detailed Design and (V) Implementation. (Khalil and Ashabi, 2014)

The Requirements analysis of the Tropos methodology is split into two key phases: 1) Early Requirements and 2) Late Requirements analysis. (BRESCIANI, PERINI et al., 2004) Both share the same conceptual and methodological approach. Almost all of the ideas generated during early requirements analysis are used in late requirements phase. More precisely, during the first phase, the domain stakeholders are identified and are modeled as social actors, that depends on each other for goals that are to be achieved, plans that are to be performed, and resources. By precisely stating these dependencies, it is possible to define the what, why, and how, for the system functionalities and, finally, to verify how the intended implementation matches with the initial requirements. During the Late Requirements analysis, the conceptual model designed earlier is extended by including a new actor, that represents the system, and also the dependencies with other actors. The functional and non-functional requirements are defined by these dependencies.

The architectural and detailed design phases provide details of the system specification, from earlier phases. "Architectural Design describes the system's global architecture in vocabulary of the constituent sub-systems which are interconnected through data and control flows" (BRESCIANI, PERINI et al., 2004). In the model the sub-systems are denoted as actors and the data and control connections are denoted as dependencies. Along with this within the architectural design there is a mapping provided for the actors to a collection of software agents. "The Detailed Design phase is meant at specifying the agent capabilities and their interactions" (BRESCIANI, PERINI et al., 2004). At this point, the implementation platform should have been finalized; this can be incorporated to complete a detailed design which will be translated directly to the target code.

Development Steps

The development steps are made up of goal analysis with regards to the different actors, and is described by a non-deterministic concurrent algorithm. Initially, the process begins with a set number of actors, each associated with a list of root goals (including soft goals). Each root goal is analyzed from the viewpoint of its respective actor, and as sub-goals are generated, they are either delegated to other actors, or the actor takes on the responsibility of dealing with them all by itself. (Giunchiglia, Mylopoulos et al., 2003). This analysis is done in parallel for each root goal. Occasionally the process needs injecting new actors that will be delegated with goals or tasks. This process is assumed done when all the goals have achieved the satisfaction level of the actors. It is assumed that actorList has a finite set of actors, and the goals list for actor is saved in "goal List(actor)" (Giunchiglia, Mylopoulos et al., 2003). Also, "agenda(actor") (Giunchiglia, Mylopoulos et al., 2003) has the goals list for actor that has been undertaken independently, besides the plan selected for each goal. Initially, agenda(actor) list has no value. "dependencyList includes a collection of dependencies between the actors, whereas capability List(actor) has <goal, plan >pairs indicating the way by which the actor can achieve particular goals." (Giunchiglia, Mylopoulos et.al., 2003)

The goalGraph saves an illustration of the goal graph that was generated in the design process. To start with, the goalGraph has all the root goals for all the actors with no links.

"The procedure rootGoal Analysis performs goal analysis in parallel for each root goal" (Giunchiglia, Mylopoulos et.al., 2003) the pseudo code is shown as below.

```
"global actorList,goalList,agenda,dependencyList,
        capabilityList,goalGraph;
procedure rootGoalAnalysis(actorList,goalList,
        goalGraph)
begin
rootGoalList = nil;
for actor in actorList do
for rootGoal in goalList(actor) do
        rootGoalList = add(rootGoal, rootGoalList);
        rootGoal.actor = actor;
end ;
end ;
end ;
concurrent for
rootGoal in rootGoalList do
goalAnalysis(rootGoal,actorList)
end concurrent for;
if not[satis f ied(rootGoalList,goalGraph)]
        then f ail;
```

***end procedure*”** (*Giunchiglia, Mylopoulos et al., 2003*)

3.3 O-Mase Methodology

O-MaSE is a new loom for analyzing and designing agent-based systems. O-MaSE is designed from a set of method fragments that will be used in a “method engineering framework.” (Garcia-Ojeda, Deloach et al., 2007) The main purpose of O-MaSE is to let developers to create customized agent-oriented software development processes.

As per Ojeda “O-MaSE is made up of three basic structures: (1) a metamodel, (2) a set of methods fragments, and (3) a set of guidelines (Garcia-Ojeda, Deloach et. al 2007). The metamodel defines the key concepts that are needed to design and implement a system. The method fragments are errands that are run to generate a basket of deliverables, that might include code, models, or documentation. Guidelines are referred to illustrate integration of the method fragments in order to obtain “O-MaSE compliant processes.” (Garcia-Ojeda, Deloach et al., 2007)

3.3.1 Metamodel

The O-MaSE metamodel” (Garcia-Ojeda, Deloach et al., 2007) defines the main concepts that is used to define a system. “The metamodel encapsulates the rules or grammar of the notation and depicts them graphically using object-oriented concepts like the class and inheritance relation.” (Garcia-Ojeda, Deloach et.al., 2007) The O-MaSE metamodel is derived upon an organizational approach. As shown in Fig.8, the “The Organization is a collection of five entities which are: 1) Goals, 2) Roles, 3) Agents, 4) Domain Model, and 5) Policies. (Garcia-Ojeda, Deloach et.al., 2007)

A Goal describes the overall purpose of the organization; A Role describes a position within an organization whose behavior is likely to achieve a particular goal or a set of goals.

Agents are like human nature or are artificial (can be either hardware or software based) entities that perceive their environment and perform their actions upon the environment. In order to observe and act within an environment, agents own capabilities, which define the procedures or actions. Capabilities can be soft one (i.e., algorithms or plans) or hard one (i.e., hardware related actions). Plans have algorithms that the agents use for conducting explicit tasks, while Actions allows agents to recognize or sense objects in the environment.

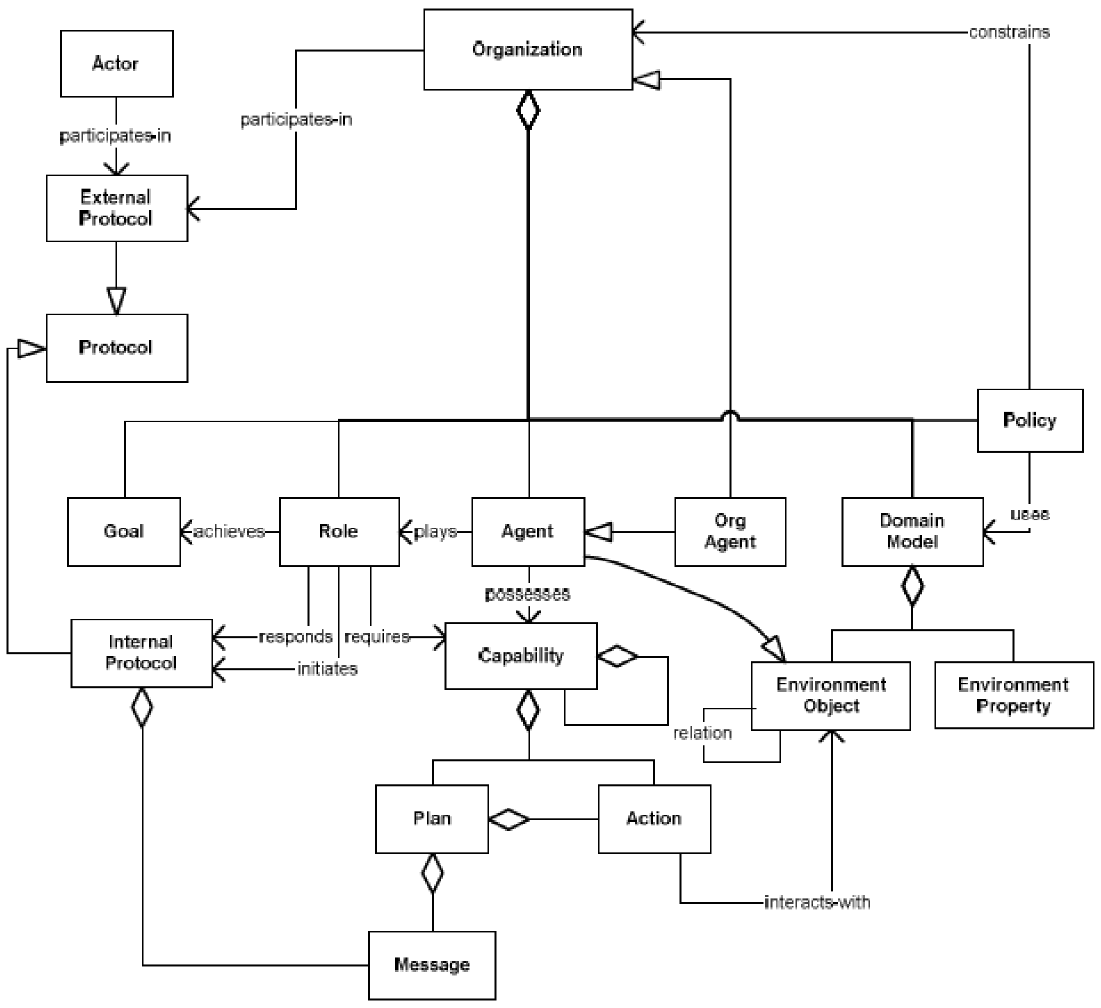


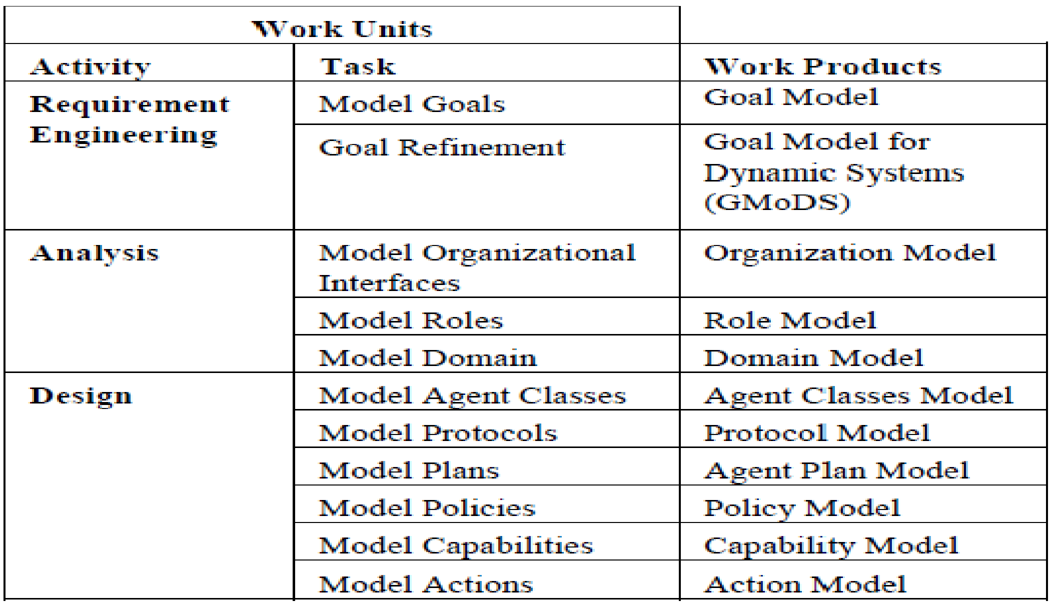
Figure 7 O-MaSE metamodel (adopted from (Garcia-Ojeda, Deloach, et.al. 2007))

The environment is defined using the Domain Model, which describes the types of objects present in the environment and the relations among them. Finally, a Policy describes how an agent might or might not behave in a specific situation." (Garcia-Ojeda, Deloach et al., 2007)

3.3.2 Method Fragments

The initial set of method fragments are derived from an extended version of the MaSE methodology. (Garcia-Ojeda, Deloach et al., 2007) O-MaSE is based on an iterative cycle across all phases with each successive iteration adding details to the models until a complete detailed design is developed.

Table 1 O-MaSE method fragments (adopted from (Mohire, 2012))



Work Units		
Activity	**Task**	**Work Products**
Requirement Engineering	Model Goals	Goal Model
	Goal Refinement	Goal Model for Dynamic Systems (GMoDS)
Analysis	Model Organizational Interfaces	Organization Model
	Model Roles	Role Model
	Model Domain	Domain Model
Design	Model Agent Classes	Agent Classes Model
	Model Protocols	Protocol Model
	Model Plans	Agent Plan Model
	Model Policies	Policy Model
	Model Capabilities	Capability Model
	Model Actions	Action Model

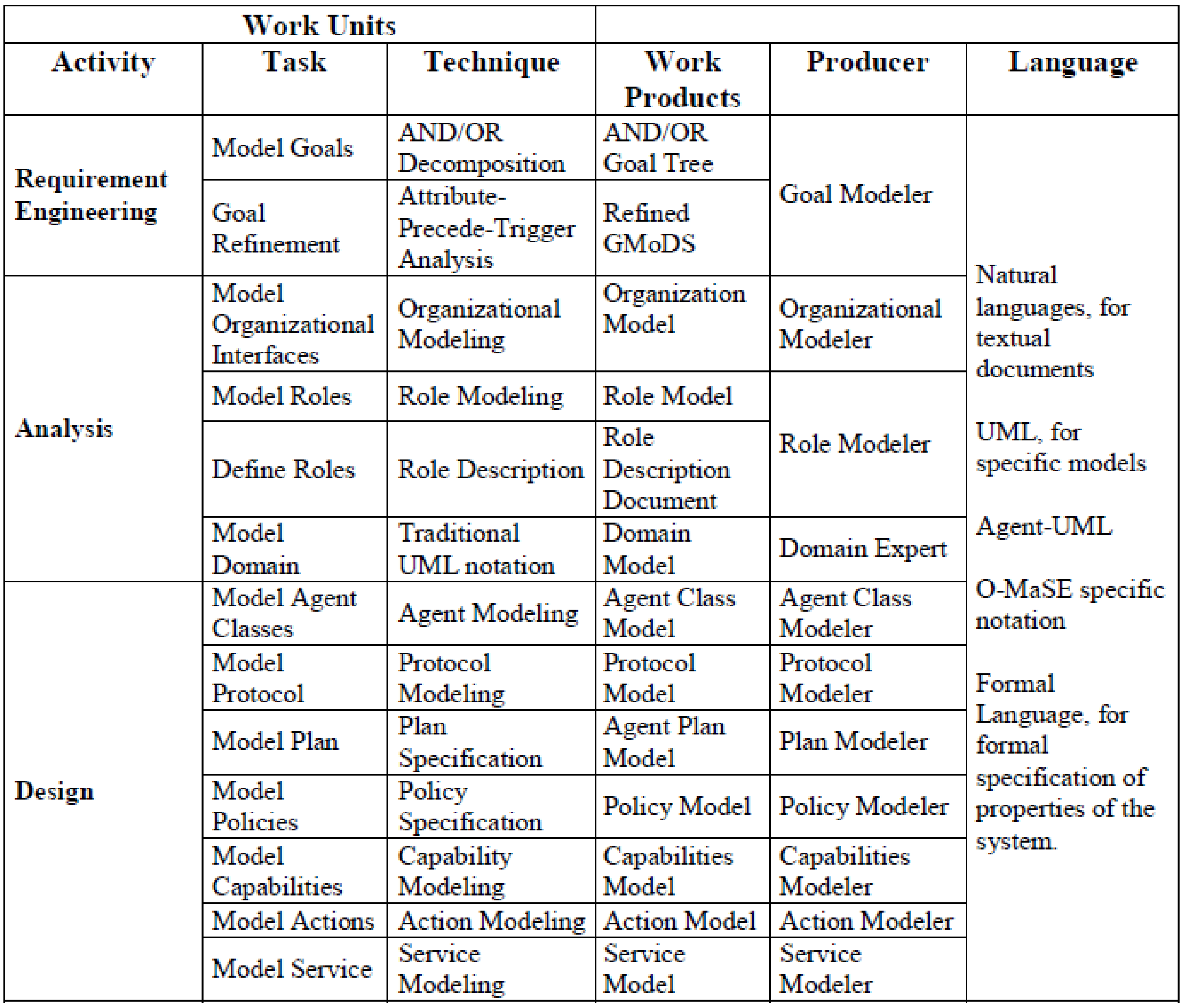
O-MaSE, has three main activities: (1) requirements engineering, (2) analysis, and (3) design. (Garcia-Ojeda, Deloach, et al., 2007) As shown in Table 1, each Activity is decomposed into a collection of Tasks and these identify a group of Techniques which can be leveraged to realize each of the tasks.

During the Requirement Engineering phase, systems requirement is translated into system level goals by defining "Model Goals and Goal Refinement" (Garcia-Ojeda, Deloach et al., 2007). The Model Goals is used for transforming the requirements into a goal tree while the Goal Refinement refines the relationships and attributes for the defined goals. "The goal tree is captured as a Goal Model for Dynamic Systems (GMoDS) The Goal Modeler should be able to: (A) use AND/OR Decomposition and Attribute-Precede-Trigger Analysis (APT) techniques, (B) understand the System's Description- SD or System's Requirements Specifications- SRS, and (C) interact with domain experts and customers. The outcome of Model Goals and Goal refinement are an AND/OR Goal Tree and GMoDS tree" (Garcia-Ojeda, Deloach et al., 2007).

3.3.3 Guidelines (Garcia-Ojeda, Deloach, et.al.,2007)

Guidelines are used to describe the integration of the method fragments to obtain O-MaSE compliant processes. These guidelines are specified as a set of constraints associated to Work Units and Work Products, which are specified as Work Unit preconditions and postconditions. (Garcia-Ojeda, Deloach et al., 2007). In my words guidelines are the best practices that should be followed to generate a O-MaSE complaint deliverables.

Table 2 O-MaSE Method Fragments (adopted from (Garcia-Ojeda, Deloach, Robby et.al. 2007))



Work Units					
Activity	**Task**	**Technique**	**Work Products**	**Producer**	**Language**
Requirement Engineering	Model Goals	AND/OR Decomposition	AND/OR Goal Tree	Goal Modeler	Natural languages, for textual documents UML, for specific models Agent-UML O-MaSE specific notation Formal Language, for formal specification of properties of the system.
	Goal Refinement	Attribute-Precede-Trigger Analysis	Refined GMoDS		
Analysis	Model Organizational Interfaces	Organizational Modeling	Organization Model	Organizational Modeler	
	Model Roles	Role Modeling	Role Model	Role Modeler	
	Define Roles	Role Description	Role Description Document		
	Model Domain	Traditional UML notation	Domain Model	Domain Expert	
Design	Model Agent Classes	Agent Modeling	Agent Class Model	Agent Class Modeler	
	Model Protocol	Protocol Modeling	Protocol Model	Protocol Modeler	
	Model Plan	Plan Specification	Agent Plan Model	Plan Modeler	
	Model Policies	Policy Specification	Policy Model	Policy Modeler	
	Model Capabilities	Capability Modeling	Capabilities Model	Capabilities Modeler	
	Model Actions	Action Modeling	Action Model	Action Modeler	
	Model Service	Service Modeling	Service Model	Service Modeler	

3.4 MaSE Methodology / process

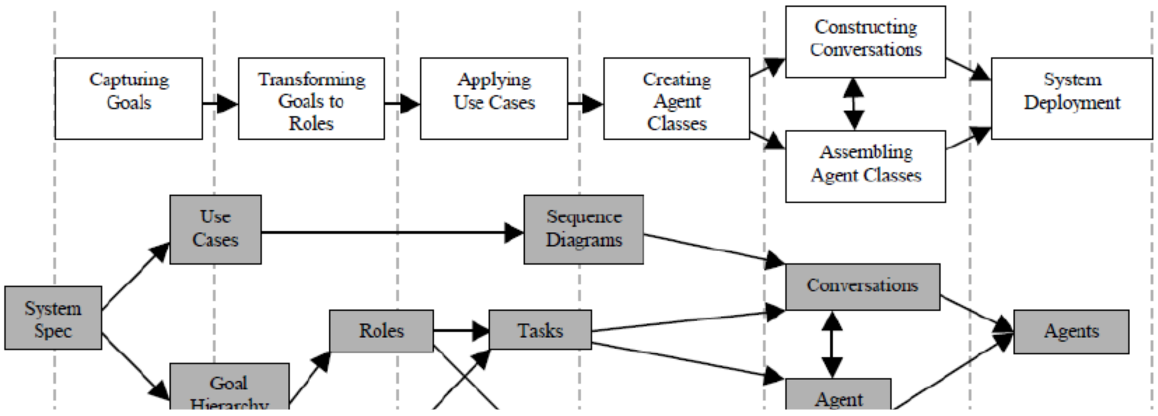


Figure 8 The entire MASE methodology (adopted from Mohire, 2012)

MaSE is a legendary software engineering methodology that has two key phases which are Analysis and Design along with several activities.

The MaSE Analysis phase has three steps: 1) Capturing Goals, 2) Applying Use Cases, and 3) Refining Roles. The Design phase has four activities: 1) Creating Agent Classes, 2) Constructing Conversations, 3) Assembling Agent Classes and 4) System Design (Werneck, Moreira Costa et al., 2011)

The initial step of the MaSE analysis phase is in capturing goals which provide information about the required system achievement. The captured goals are supposed to be stable throughout the Analysis and Design phases. The decomposition of goals in a hierarchy way is defined by the MaSE goal representation. Once the goals are defined, next the functional requirements are identified and represented by using the use case diagrams. Use Cases describe the behavior of agents for every agent condition. For the Applying Use Cases step, conditions of the initial requirements are expressed as UML Use Cases Diagrams, Descriptions, and Sequence Diagrams.

The Sequence Diagrams express the sequences of roles, events, the desired system behavior and its sequences of events. The ultimate step of the Analysis phase is to denote set of roles (by using Role Diagram) that are leveraged in achieving the desired goals of the system. The role is an abstract narrative behavior of an agent that helps in achieving the system goals. The role is defined by tasks that are "finite-state models -like the Concurrent Tasks Diagram" (Werneck, Moreira Costa et al., 2011). This completes the development work of the Analysis phase.

Four diagrams are proposed in the MaSE Design phase which are 1) Agent Classes, 2) Conversations, 3) Agent Architecture and 4) Deployment Diagram. (Werneck, Moreira Costa et al., 2011)

The initial step in the design process is the definition of each agent class by using an Agent Class Diagram. "The system developer then maps each defined role in the Roles Diagram to at least one Agent Class" (Werneck, Moreira Costa et al., 2011). This is done as this guarantees that the goals will be implemented and there exists at least one agent class that is responsible for achieving the goal. The agent classes can be assumed as templates that define agent roles and the protocols used

The subsequently step for the design process is to define the communication among the agent classes and to define a coordination protocol among the agents. "A dialogue consists of two Communication Class Diagrams that symbolize the initiator and the responder. This Communication class diagram is a finite state automation defining the conversation states of the agent classes using a similar syntax that was used in the analysis phase". (Werneck, Moreira Costa et al., 2011)

The third step in the Design process is related to definition of the agent architecture, for which the steps are: "(1) definition of the agent architecture and (2) its components" (Werneck, Moreira Costa et al., 2011). The developer now can opt an agent architecture, such as Reactive, Knowledge Base or Belief-Desire-Intention (BDI).

The last activity in the Design process is the definition of "Deployment Diagram" (Werneck, Moreira Costa et al., 2011). In system, this diagram denotes the number of agents, types of agents and deployment location of the agents in the system. The diagram describes the system based on agent classes, which is similar to UML Deployment Diagram. This diagram defines main and alternative agents' configurations on different platforms that helps in maximize the processing power and network bandwidth

3.4.1 MaSE TOOL

MaSE can be developed using the AgenTool that is developed by the Air Force Institute of Technology (AFIT). AgenTool helps the system developer in creating a series of models, from higher level goals definition to an automatic verification, a semi-automatic generation design and finally code generation. (Werneck, Moreira Costa et al., 2011)

In agentTool, out of the seven phases of MaSE has currently realized three phases, as depicted by the outlined box in Figure 10 below. The planned future of agentTool includes expanding it to other phases of the MaSE methodology. The main feature and goal of agentTool is intended to allow system developers to define the structure and behavior of a multiagent system and using semi-automation produce systems that comply with the requirements.

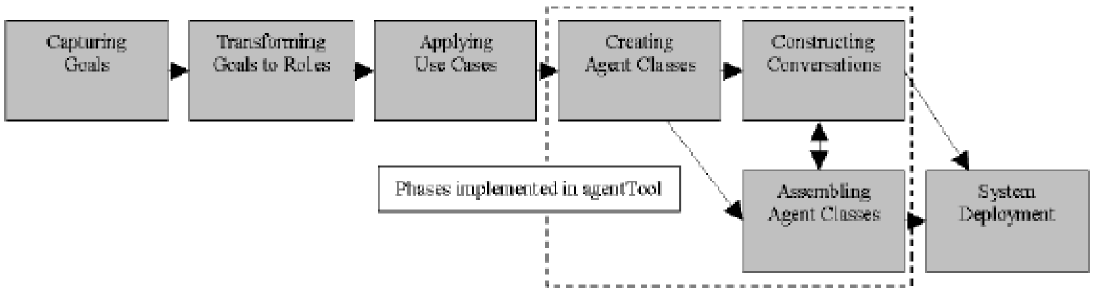


Figure 9 MaSE in agentTool (adopted from (Mohire, 2012))

Gaia

Gaia is intended to allow a developer to proceed from a statement of requirements to a design phase which is sufficiently detailed that ensures the implementation occurs directly. The requirements phase is independent of the analysis and design. In applying Gaia, the developer refines from abstract to more concrete concepts in steps. Each successive step introduces greater implementation scope, and reduces and refines the scope of systems that can be used to implement to meet the requirements statement. The main models used of Gaia are depicted in

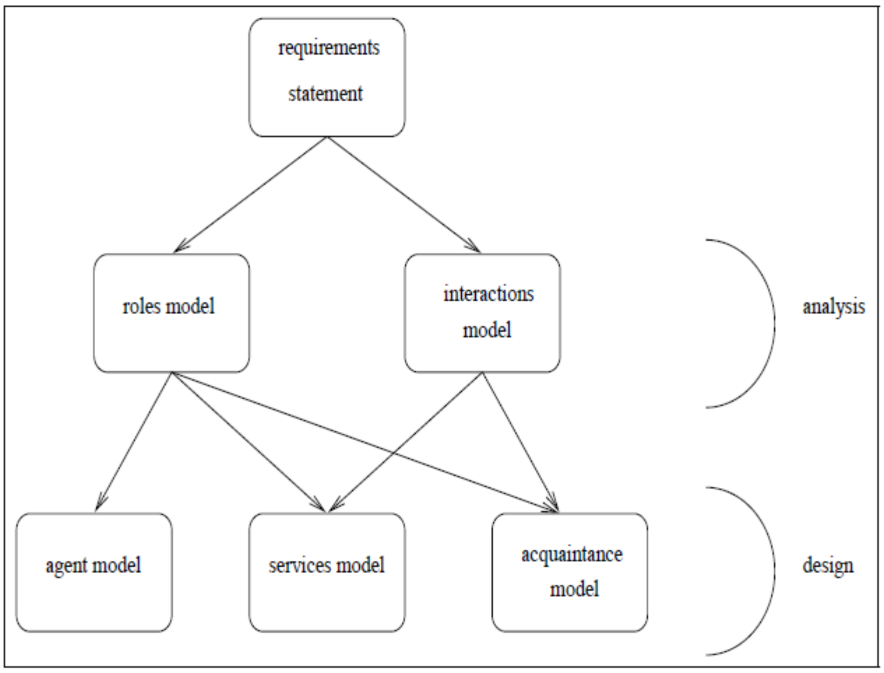
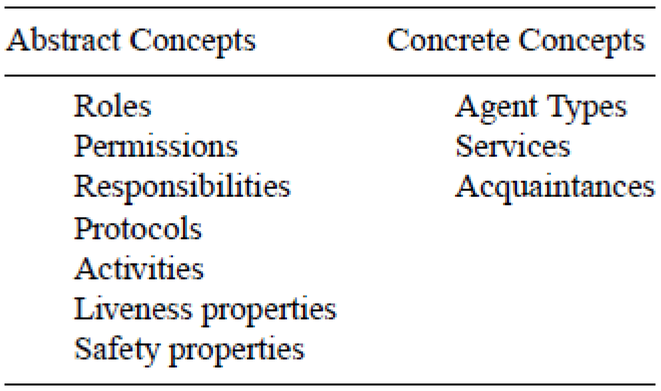


Figure 10 Relationships between Gaia's models (adopted from (Wooldridge, Jennings, et.al. 2017))

Figure 11.

The key Gaian concepts are divided into two categories: 1) abstract and 2) concrete; (Wooldridge, Jennings et al., 2017) these two concepts are summarized in Table 3. Abstract entities are used in analysis phase to conceptualize the agent system; however, these may not have any direct realization within the system. Concrete entities, differ; these are used in the design process, and will have direct counterparts in the production system.

Table 3 Abstract and concrete concepts in Gaia (adopted from (Wooldridge, Jennings, et.al. 2017))



Abstract Concepts	Concrete Concepts
Roles	Agent Types
Permissions	Services
Responsibilities	Acquaintances
Protocols	
Activities	
Liveness properties	
Safety properties	

In the Gaia process is a series of models are constructed, that describe both the "macro (societal) aspects and the micro (intra-agent) aspects" (Wooldridge, Jennings, et.al, 2017) of a multi-agent system (MAS). Figure 12 shows the analysis and design phases that are related to the GAIA methodology.

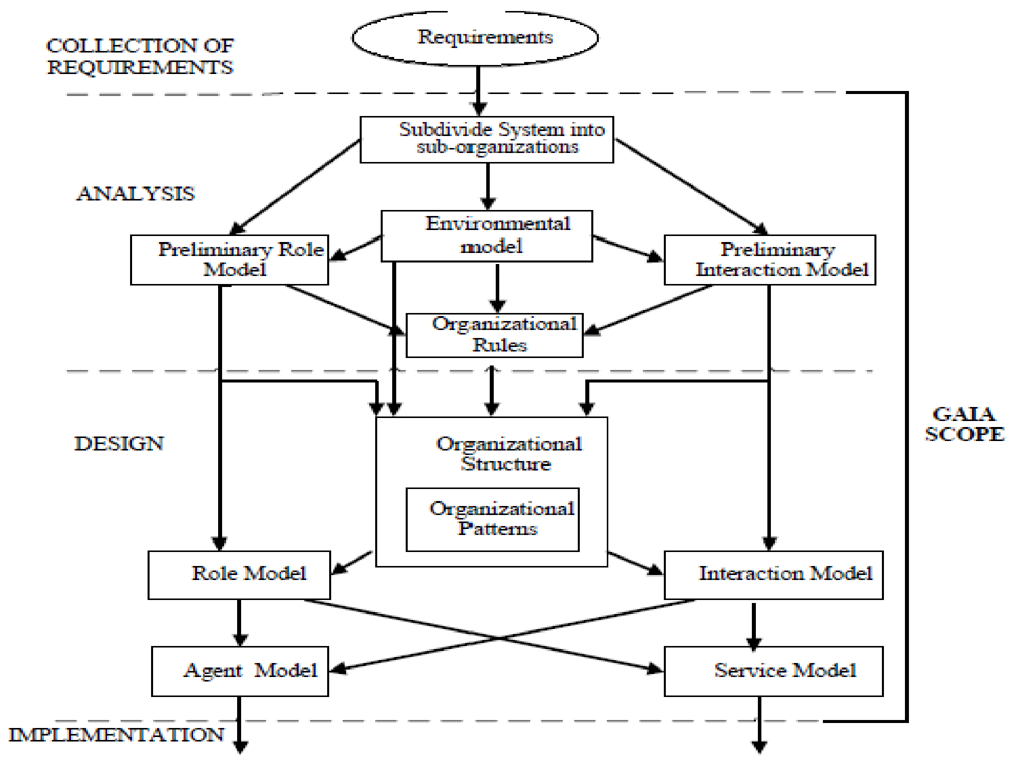


Figure 11 Models of Gaia v.2 and their Relationship (adopted from (Cernuzzi, Juan, et.al.2017))

In the analysis phase, the initial role model and the interaction models are developed, that bring into the MAS system a set of abstract roles. These two models then become inputs to the design stage. In the design model other models are developed which are role model, agent model, services model, and an interaction / acquaintance model. Outputs from the design model becomes the input to the implementation phase. I will describe these phases.

3.4.2 Analysis Phase

The key aim of the Gaia analysis process is to identify the roles and the model the interactions between these roles. Roles consist of four attributes: responsibilities, permissions, activates, and protocols (Yang, Yokoo et al., 2009) Responsibilities are the key attribute associated with a role and they identify its functionality. Responsibilities are classified as of two types: _LIVELINESS properties and SAFETY properties. LIVELINESS – is that which adds goodness to the system, and SAFETY properties – is that which must prevent and disallow that something bad that can occur to the system. Liveliness describes the tasks that an agent must fulfill for certain given environmental conditions and safety that ensures an acceptable state of action is maintained during the entire execution cycle. For responsibilities, a role has a set of permissions assigned to it. Permissions represent the activities the role is allowed to do and in particular, the information resources that it is allowed to access. Activities can be defined as independent tasks that an agent performs without interacting with other agents. Lastly, protocols are the ones that provide precise patterns of communication, e.g. a buyer role can support different purchasing protocols. Gaia has prescribed templates and operators for representing agent roles and its attribute. Gaia as well has schemas that may be used for the depiction of interactions between the various roles of a system.

3.4.3 Design Phase

In the Gaia design process the initial step is to map roles with the agent types and to generate the right quantity of agent instances for every type. An agent type can be composed aggregation of one or more agent roles. The next step is to determine the SERVICE model that is needed to fulfill a role for one or multiple agents. A service is as a function of the agent and can be obtained from a list of protocols, activities, responsibilities and the liveliness properties of a role. Final step, is developing the ACQUAINTANCE model for the depiction of dialogue between agents. The acquaintance model is not to define the actual messages that are exchanged between the agents; however, it is a simple graph that represents the communication pathways between the different agent types.

The implementation phase is not part of the GAIA methodology and hence not included here. It is left to the developer to choose the best product that fits this methodology.

3.5 AUML

The current version of UML is not agent oriented and is not enough to model agents and agent-based MAS systems. Also, no matured paradigm yet exists that sufficiently specify agent-based design and development in totality. To incorporate agent-based design and development in projects, a specification technique should support the entire software engineering process paradigm—starting from planning, through analysis design, and finally to system development, transition, and maintenance. “Both FIPA and the OMG Agent Work Group are exploring and recommending extensions to UML” (Odell, Parunak et al., 2000)

“UML” (Booch and Rumbaugh, 1999) offers many advantages than other paradigms as it is widely used in industrial grade software projects and many popular tools are readily available in the market to suit the needs of a mature development environment. Moreover, non-IT business designers can easily design use cases that fit the business need. “As Odell and colleagues indicated, starting all over with a new modeling language for agents by reinventing the wheel, it would be neither useful nor productive. Instead, multiagent systems would benefit from an incremental development of prevailing trusted methods. Agent UML is one that provides such a solution. The initiative behind AUML is to utilize UML extension capabilities such as stereotypes and constraints" (Odell, Parunak et al., 2000). AUML makes it easy for the growing concern of agent-based modeling representations; it lets IT and non-IT developers smoothly transit from software development to agent development without much relearning.

AUML is not exactly a methodology as it is a notational language used by several methodologies like RUP that has been adopted by several development tools. AUML is referenced here to make it clear that methodologies like RUP can be used to design agent domain specific artefacts

AUML helps design by way use of features like decomposition, abstraction and organization characteristics, for reducing the development complexity.

In AUML decomposition, it decomposes a system into smaller parts like the objects, models, use-case and classes, along with related operational actions.

About abstraction, it provides a generalized abstract view of the model entities (such as class, use-case, diagram, interface etc.). It creates a collection of conditions and semantics for the infrastructure and operations services.

Lastly, the agent–based organization defines an array of elements and notations as a set of requirements specification for domain modeling. It provides a model and an internal architecture of an agent system. It offers a set of frameworks (class, diagram, interface, etc.) which demonstrate how agents should be developed in the agent system. The modeling focuses on one aspect or view and provides the ability to understand complex issues.

The foundational parts of AUML are a set of mechanisms to model exchange patterns for multi-agent interaction. "This is possible by introducing a new category of diagrams into UML which is the protocol diagrams. These diagrams provide extensions to existing UML state and sequence diagrams" (Odell, Parunak et al., 2000). Specific extensions comprise of multithreaded lifelines, agent specific roles, parameterized nested protocols, extended message semantics, and protocol templates to quote a few.

What we need apart from the object oriented design paradigm, is to consider agent as a special case of object. Figure 13 illustrates our own view on the relationship between an agent and an object.

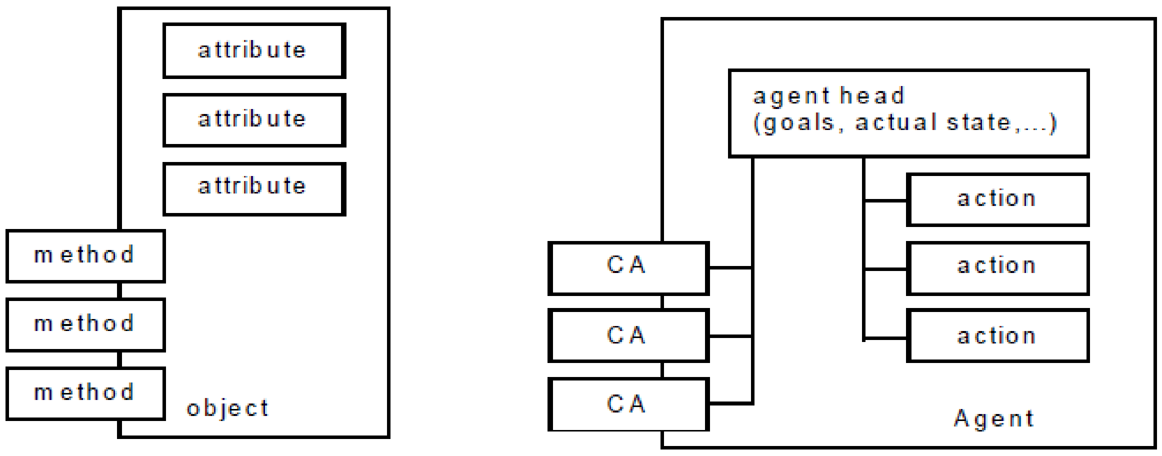


Figure 12 object vs agent (adopted from (Bauer, Müller, et.al, 2001))

Agents have autonomy, pro- and re-activity, and communication that is based on speech act theory (communicative act, CA for short). The internal state is more than fields with imperative datatypes, and additional features. All these concepts have to be supported by a class diagram for agents. (Bauer, Müllerv et al., 2001)

In the agent, oriented programming paradigm there is a distinct difference between an agent class and individual class. The Agent class defines the type and blueprint of an individual agent, and individual agents which is an instance of an agent class. Hence, I specify the schema of an agent class which is later used in programs as instantiated agents.

Relating classes as to agent technology raises the question about class in the context of agents. My answer is that in the same sense in which a class is a blueprint for objects, in our context an agent class is a blueprint for agents. This can be either an instance of the agent or a collection of agents satisfying some special role or behavior.

Few of the common characteristics that are common to agents are:

Identifier:this uniquely identifies an agent in a multi-agent system

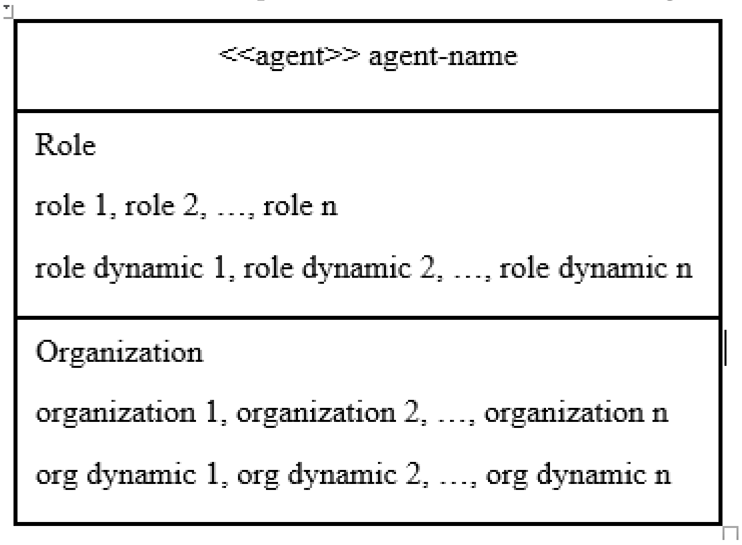
Role:this describes the activities of an agent within the society (es., Buyer and Seller)

Organization:this describes the relationships among the roles (like animal or human organizations that are markets, hierarchies, herds or groups of interest)

Capability:this defines the agent's actions that it is able to perform under a given condition

Service : this describes an action that an agent can execute to other agents. (Bauer, Müller, et.al, 2001)

The practical diagrams used by AUML are described below. UML Class Diagram can be used to depict the standalone view of an agent.


<<agent>> agent-name

Role

role 1, role 2, …, role n

role dynamic 1, role dynamic 2, …, role dynamic n

Organization

organization 1, organization 2, …, organization n

org dynamic 1, org dynamic 2, …, org dynamic n

Figure 13 Static View of agent

3.5.1 AUML Diagrams

I have reproduced the AUML diagrams as per my research.

AUML Use Case Diagram

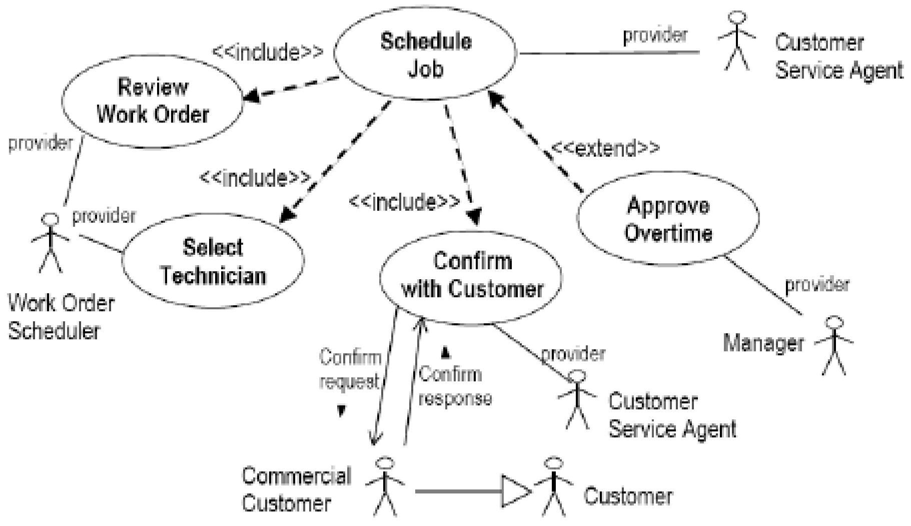


Figure 14 AUML Use Case Diagram (adopted from (Gatti, Staa et al., 2007))

As shown in figure, AUML Use case captures goal oriented interactions between agents with specified roles and the software system. These are a set of usage patterns within the system, each with its own discrete goal.

AUML Class Diagram

AUML Class Diagram describes the different types of agents within the system and their static relationships like one to one or one to many.

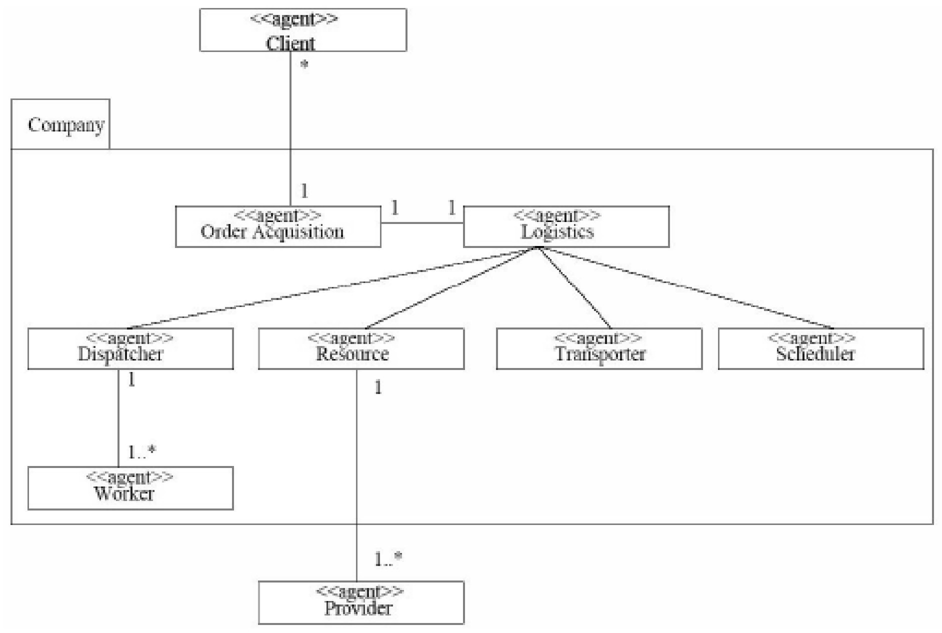


Figure 15 AUML Class Diagram (adopted from (Gatti, Staa et al., 2007))

AUML Sequence Diagram

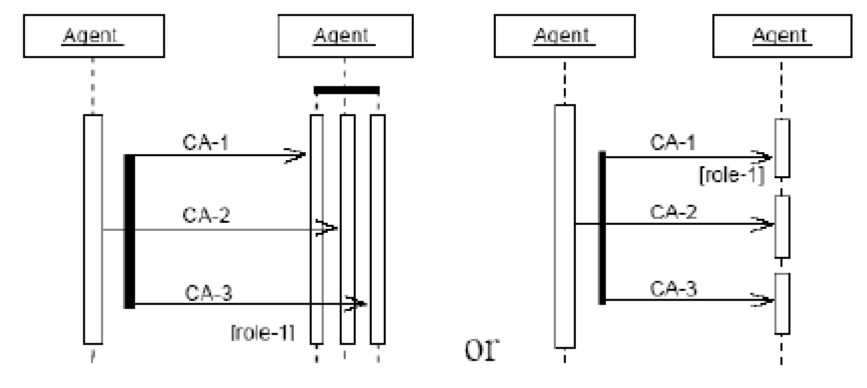


Figure 16 AUML Sequence Diagram (adopted from (Gatti, Staa et al., 2007))

Sequence diagrams are used to show the interactions among agents. Agent interaction protocol (AIP) describes the message blueprint, with a permissible sequence of messages among agents. These messages encompass constraints for the content of the messages, and has semantics that is consistent with the communicative acts (CAs) within a communication pattern (Gatti, Staa et al., 2007)

AUML State Diagram

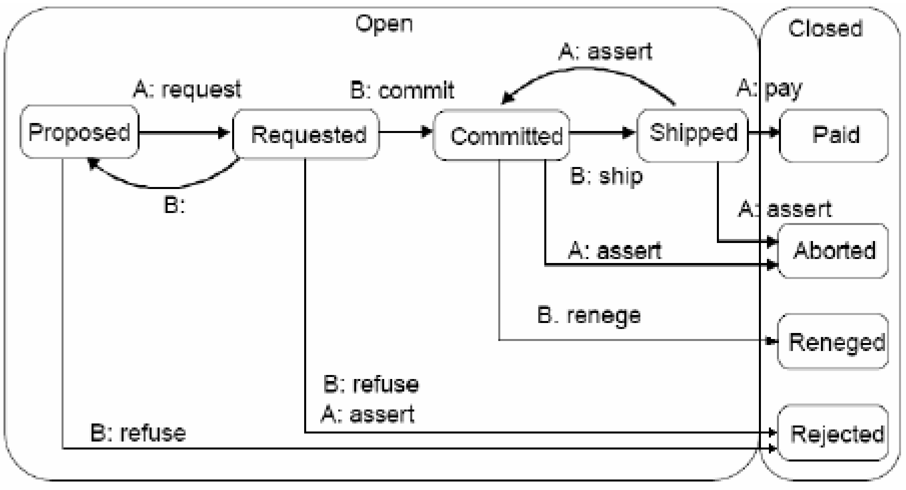


Figure 17 AUML State Diagram (adopted from (Gatti, Staa et al., 2007))

State diagrams show the transitions from one state to another. It also shows the states and protocols used by the agents playing different role.

AUML Activity Diagram

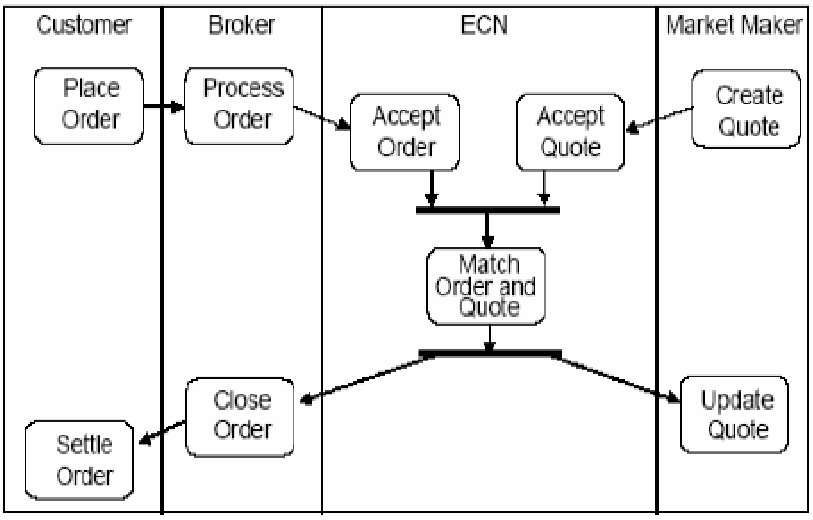


Figure 18 AUML Activity Diagram (adopted from (Gatti, Staa et al., 2007))

These show the activities of a protocol or of a role. These are usually arranged in swim lanes with activities arranged as per Role in these swim lanes.

3.6 INGENIAS

INGENIAS (Engineering for Software Agents) is an open-source software framework used to analyze, design and implement the MAS. (Pavon Gomez-Sanz et al., 2003)

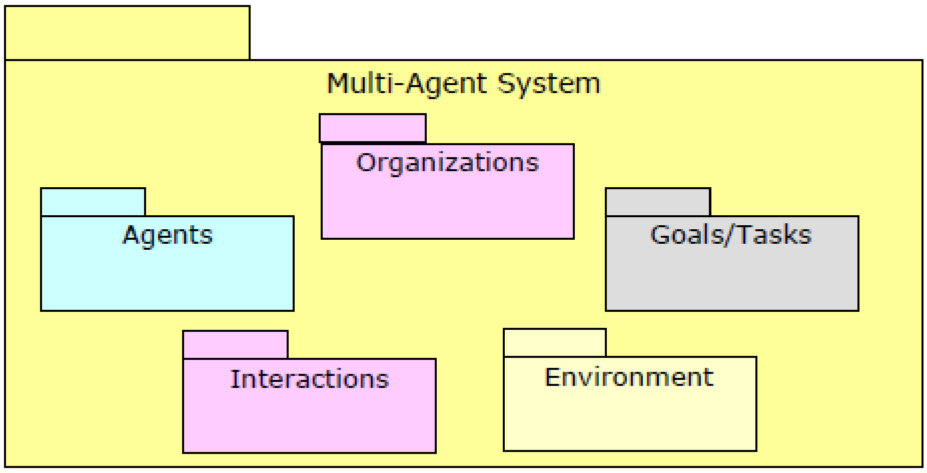


Figure 19 INGENIAS viewpoints (adopted from (Pavón, Gómez et al., 2006))

(Fig. 19) represent the INGENIAS viewpoints from which a MAS can be constructed. These viewpoints are: organization, agent, goals/tasks, interactions, and environment (Pavón, Gómez et al., 2006)

Agents are considered as intentional entities that pursue the set goals and they play roles to achieve the goals within the MAS organization. Considering the mental state (a set of facts, goals, believes) the agent decides which action (tasks) it will try to perform. "The mechanisms to make such decision are encapsulated in a Mental State Processor(P) and the

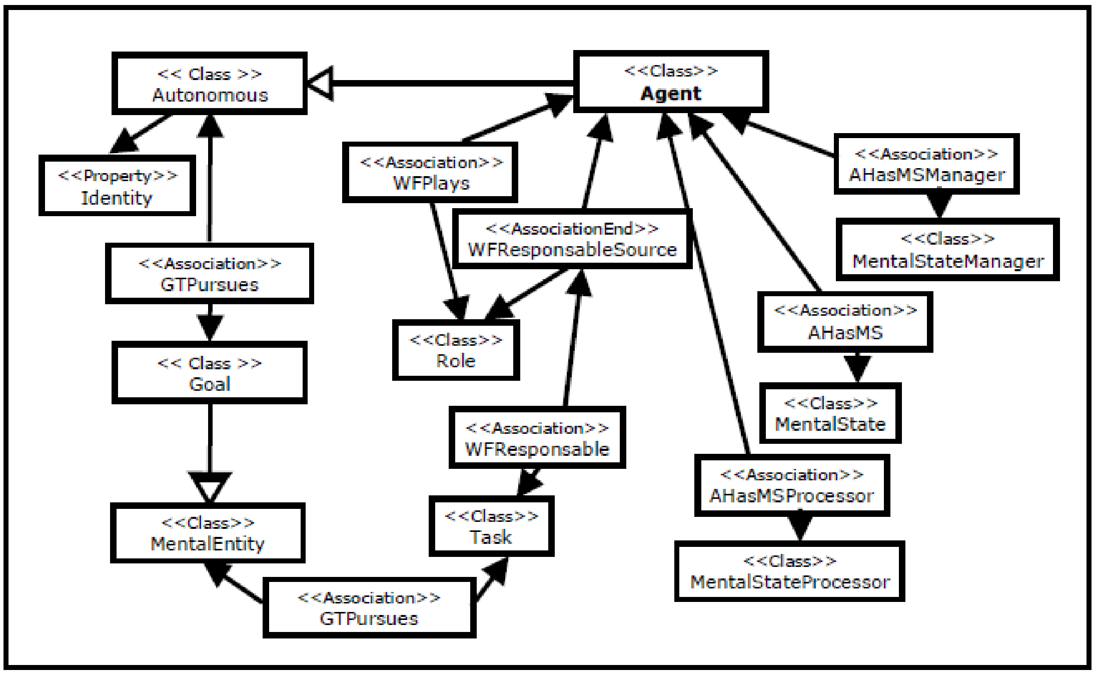

<< Class >> Autonomous
<<Class>> Agent
<<Property>> Identity
<<Association>> WFPlays
<<Association>> AHasMSManager
<<AssociationEnd>> WFResponsableSource
<<Class>> MentalStateManager
<<Association>> GTPursues
<<Association>> AHasMS
<< Class >> Goal
<<Class>> Role
<<Class>> MentalState
<<Association>> WFResponsable
<<Association>> AHasMSProcessor
<<Class>> MentalEntity
<<Class>> Task
<<Class>> MentalStateProcessor
<<Association>> GTPursues

Figure 20 Fragment of the INGENIAS Agent viewpoint meta-model. Stereotypes indicate the intended use of each entity at model level (adopted from Pavón, Gómez et al., 2006)

management of mental state entities (creation, modification, deletion) is encapsulated in a Mental State Manager (G). "(Pavón, Gómez et al., 2006) These concepts and their dependencies are represented in meta-classes of the meta-model (see Fig 20). They may be assigned with a graphical representation for the modelling language, which can be specific to the INGENIAS tools or UML. Fig. 21 Correspondence of meta-model entities with graphical representation in

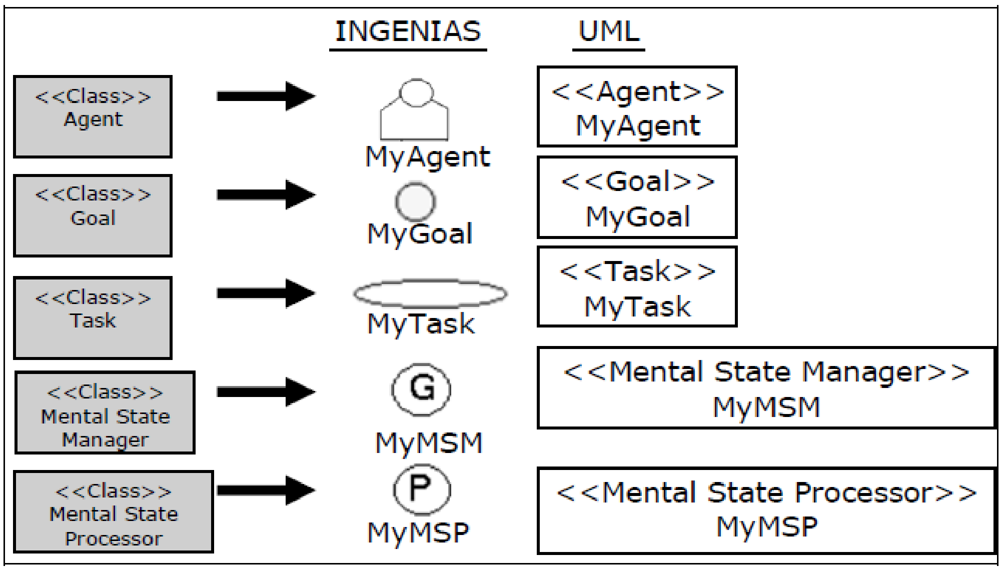


Figure 21 Correspondence of meta-model entities with graphical representation in the model with INGENIAS notation and UML (adopted from (Gatti, Cerqueira et al., 2007))

the model with INGENIAS notation and UML (adopted from Gatti, Cerqueira et al., 2007)

The current INGENIAS MAS meta-model is quite complex (at least in size). This complexity arises due to the required level of details needed to produce complete models that are to be transformed into an executable code. However, this is a decision that is left for the developer to use the meta-model to a certain degree of depth. In the initial specification of the INGENIAS methodology there are guidelines, based on the Unified Process, that guide about the elements and diagrams that should be produced in each stage of the development

INGENIAS provides a notation for modeling multi-agent systems (MAS) and a well-defined collection of activities to guide the development process of an MAS in the tasks of analysis, design, verification, and code generation, supported by an integrated set of tools—the INGENIAS Development Kit (IDK). These tools, and the INGENIAS notation, are based on five meta-models that define the different views and concepts that make up the multi-agent system. Meta-models provide flexibility for evolving the methodology and adopting changes to the notation (Gómez-Sanz and Fuentes, 2005). One of the most important purposes of this methodology is to amalgamate progressive developments in the agent technologies, towards a standard framework for agent based systems modeling. This may well facilitate the adoption of the agent approach by the software industry.

IDK is the development tool that relies on defining metamodels (Figure 22) that describe the elements that form MAS from several viewpoints; these viewpoints allows developer to define a specification language for the intended MAS using the Rational Unified Process (RUP). “These viewpoints are five in number, they are: 1) agent model (definition, control and management of agent mental states), 2) interaction model,3) organizational model, 4) environmental model, and 5) goals/tasks model”. (Mohire, 2012)

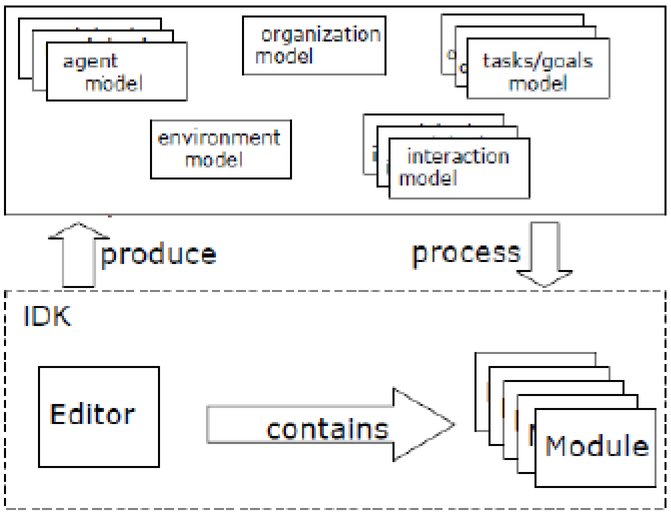


Figure 23 Relationship between the IDK, models and components (adopted from (Mohire, 2012))

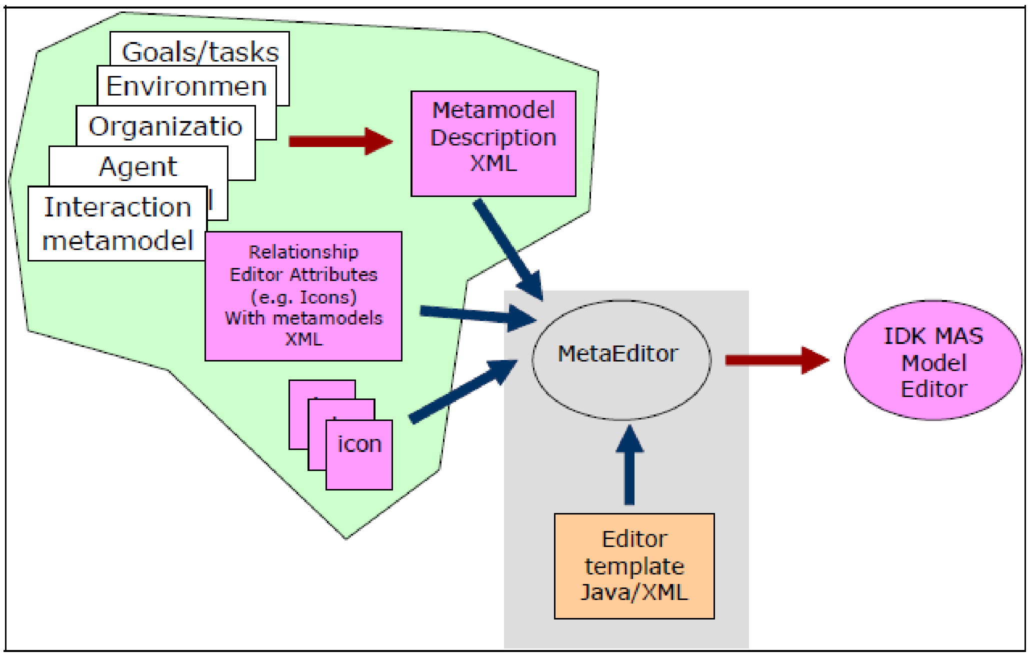


Figure 22 Generation of the IDK MAS Model Editor from meta-model specifications (adopted from (Pavón, Gómez et al., 2006))

The MAS Model editor is the key tool for the system developer. However, a MDD approach also requires verification and validation tools, along with transformation engines to generate code, documentation, or other models in the target platform. This is supported in the IDK by modules. Modules (or plugins) in INGENIAS are programs that process specifications and produce some output

INGENIAS Dev Kit has five meta-models that describe system views of the MAS. Each view, being the instance of a meta-model, is named as a model. To describe a MAS, one must develop the following models:

1) Agent model: This model defines single agents, their tasks, goals, initial mental state, and played roles. In addition, agent models are used to describe intermediate states of agents. These states are represented using goals, facts, tasks, or any such system entity that provides state description. (Juan and Gómez-Sanz, 2017)

2**)** Interaction model: This model defines the interaction among agents. Each interaction declaration includes concerned actors, goals pursued, and a description of the protocol used during the agent interaction. (Juan and Gómez-Sanz, 2017)

3) Tasks and goals model: This model describes relationships among goals and tasks, goal structures, and task structures. This is used to express the inputs and outputs of the tasks and their effects on the environment or on an agent's mental state. (Juan and Gómez-Sanz, 2017)

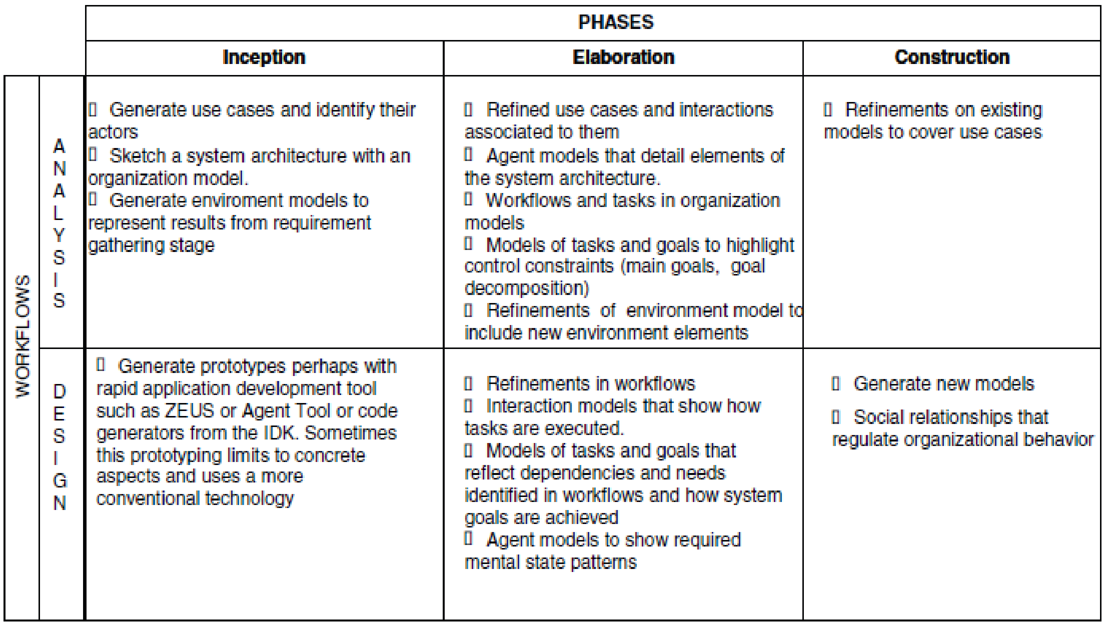
4) Organization model: This model describes the way system components (agents, roles, resources, and applications) are grouped together, the tasks that are executed in common, goals they share, and the constraints that exists during the agent interaction. These constraints are expressed in form of subordination and client-server relationships. (Juan and Gómez-Sanz, 2017)

5) Environment model: This model defines agent's perception in terms of existing elements of the system. It also identifies system resources and persons/entities responsible for their management. (Juan and Gómez-Sanz, 2017)

3.6.1 INGENIAS Development Process

The development process of INGENIAS is, the “INGENIAS Development Process (IDP)” (Mohire, 2012). By leveraging the IDP, an analyst can design the full specification of agent system. To follow the IDP analyst uses activity diagrams which describes the nature of expected results. Each meta-model can be built by conducting a set of activities as defined in the software development process which sets the path to the final MAS specification. To begin with, activities are organized and symbolized with UML activity diagrams that illustrate dependencies among them. Figure 24 below summarizes the deliverables for each phase of the IDP.

Similar to UML in object oriented applications, metamodels are used for the specification language of the multiagent system. (Mohire, 2012)



		PHASES		
		Inception	Elaboration	Construction
WORKFLOWS	ANALYSIS	▯ Generate use cases and identify their actors ▯ Sketch a system architecture with an organization model. ▯ Generate enviroment models to represent results from requirement gathering stage	▯ Refined use cases and interactions associated to them ▯ Agent models that detail elements of the system architecture. ▯ Workflows and tasks in organization models ▯ Models of tasks and goals to highlight control constraints (main goals, goal decomposition) ▯ Refinements of environment model to include new environment elements	▯ Refinements on existing models to cover use cases
	DESIGN	▯ Generate prototypes perhaps with rapid application development tool such as ZEUS or Agent Tool or code generators from the IDK. Sometimes this prototyping limits to concrete aspects and uses a more conventional technology	▯ Refinements in workflows ▯ Interaction models that show how tasks are executed. ▯ Models of tasks and goals that reflect dependencies and needs identified in workflows and how system goals are achieved ▯ Agent models to show required mental state patterns	▯ Generate new models ▯ Social relationships that regulate organizational behavior

Figure 24 Results of the analysis and design phases of the IDP (adopted from (Mohire, 2012))

Below figure shows how code is generated using the IDP/ IDK development methodology.

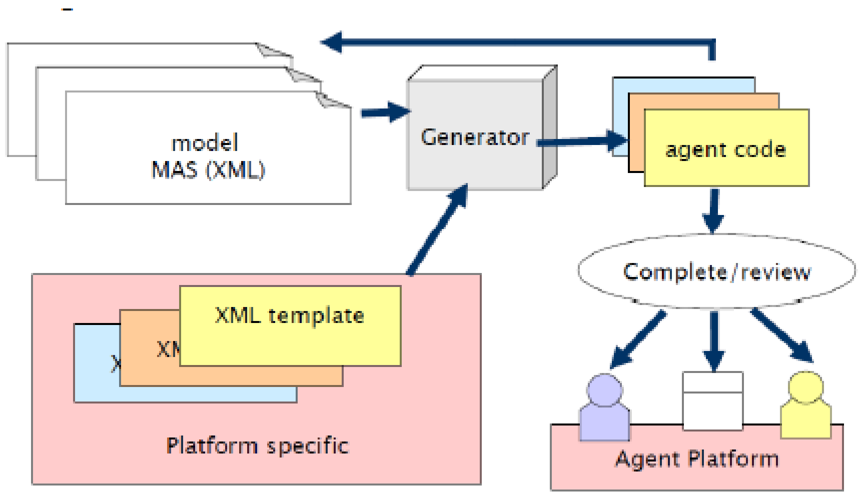


Figure 25 IDK code generation process (adopted from (Mohire, 2012))

3.7 MAS-CommonKADS

MAS-CommonKADS comes from CommonKADS a popular knowledge-engineering methodology" (Schreiber 1999), and derived "from object oriented methodologies such as Object Modelling Technique -OMT" (Rumbaugh, Blaha et al., 1991), "Object-oriented Software Engineering -OOSE" (Jacobson, Christerson et.al., 1992) and "Responsibility Driven Design - RRD (Wirfs-Brock, Wilkerson et al., 1990).

MAS-CommonKADS methodology comes from agent-oriented software engineering and provides guidance in the analysis and design of systems. Further MAS-CommonKADS is anchored on the models of CommonKADS that are extended and personalized to agent modelling. MAS CommonKADS provides a collection of models that are "1) Agent Model, 2) Task Model, 3) Expertise Model, 4) Coordination Model, 5) Communication Model, 6) Organization Model, and 7) Design Model" (Iglesias and Garijo, 2005) that describe a model of the problem that is to be solved. Model components are generic in nature to ensure reusability. The methodology defines the following models (Figure 26) along with short note.

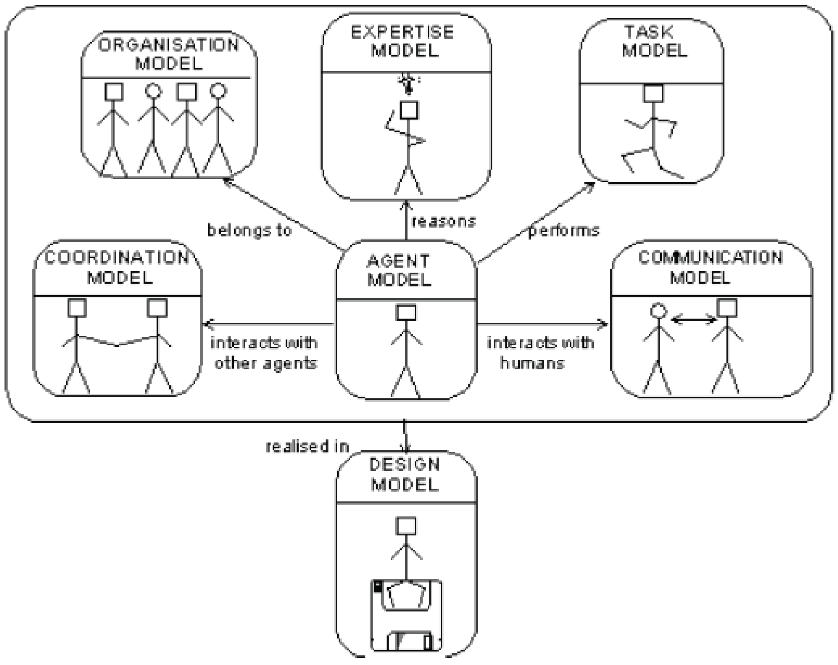


Figure 26 Models of MAS-CommonKADS (adopted from (Iglesias, and Garijo,2005))

- **Agent model:** This model specifies the agent characteristics like reasoning abilities, sensor or effector skills, agent groups, services "and hierarchies (modelled in the organization model) (Iglesias and Garijo, 2005).
- **Task model:** This model describes the tasks that agents can carry out like goals, decompositions**,** ingredients, problem-solving methods. (Iglesias and Garijo, 2005)

- **Expertise model:** This model describes the knowledge needed by agents in achieving their goals (Iglesias and Garijo, 2005)
- **Organization model:** This model defines the organization where the MAS will be introduced and also defines the social organization of the agent society (Iglesias and Garijo, 2005)
- **Coordination model:** This model describes the conversations among agents, their interactions, protocols, and the required capabilities used for the interactions (Iglesias and Garijo, 2005)
- **Communication model:** This model defines the human- agent interactions and the human factors that are involved in designing such a user interface. This model uses standard Industry prescribed techniques in developing the interfaces. (Iglesias and Garijo, 2005)
- **Design model:** This model has three sub models which are : 1) network design sub model, used for defining the required features of the agent network infrastructure (required network, knowledge and telematic facilities); 2) agent design sub model, that is used for dividing or composing the agents for analysis, as per pragmatic criteria and selecting the most appropriate architecture for every agent; and 3) platform design sub model, that is used in choosing the agent development platform for each defined agent architecture.(Iglesias, and Garijo.2005)

The software development life cycle in MAS-CommonKADS has the following phases:

- **Conceptualization**. This phase is the elicitation phase to obtain a first-hand description of the problem by defining a set of use cases that help in understanding the system and the way to test it. Two general techniques are used in MAS-CommonKADS to realize an agent-based system: Firstly, the "User-Environment-Responsibility (UER) cases technique" (Iglesias and Garijo, 2005), deal with the discovery of goals, use, and reaction, of an agent system. Secondly the "enhanced Class-Collaboration-Responsibility Cards technique" (Iglesias and Garijo, 2005) that deals with the identification of responsibilities, plans, and collaborations of an agent.

Use cases are defined using object-oriented notation and the communications are dignified with "MSC (Message Sequence Charts)" (Iglesias and Garijo, 2005). Use case interactions are dignified using MSC as a set of notations as displayed in Fig. 28. Here, two message transaction alternatives are united with the "alternative alt operator" (Iglesias and

Garijo, 2005). “A formal MSC contains the description of the asynchronous communication between entities called instances, and has primitive methods for regional action, timer (for setting, resetting and timeout), creation of processes, stopping of processes, co-regions, and inline operator expressions for event structures (parallel composition, alternative, iteration, exception and optional regions).”(Iglesias and Garijo, 2005) The rationale of the conceptualization phase is to avail insights of the agent communications, that will be cultured later on in the coordination model, by specifying the data and knowledge exchanged and the talking act of every interaction.

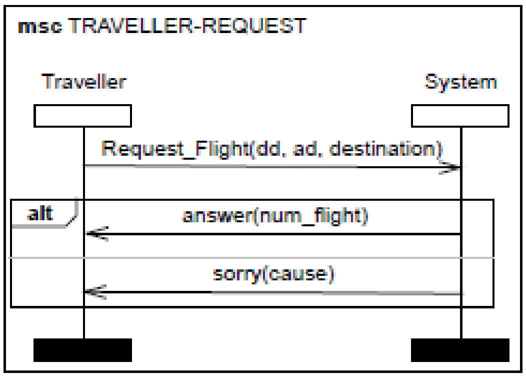


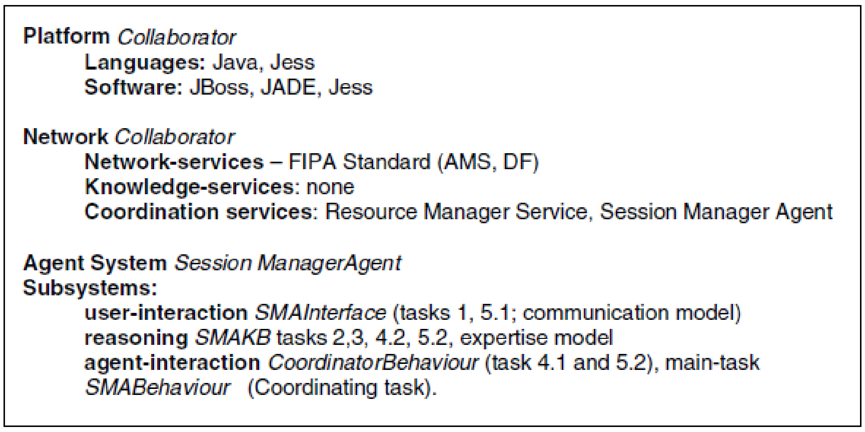
Figure 27 MSC Traveller request use case diagram (adopted from (Iglesias, Garijo et al.,1998))

Analysis. The analysis phase is used for describing the functional requirements of the system by developing a set of related models. Inputs from Conceptual phase are used in developing the various models as defined in Fig 27 (Iglesias, Garijo et al., 1998).

Design. The design phase takes an input from the analysis models, which are then transformed into specifications (the design model). “During design phase the agent internal architecture and th**e** system network architecture are determined” (Iglesias, Garijo et al., 1998). The design model defines the components that meet the requirements of the earlier analysis models. “There are three main decisions to be carried out in this model: 1) platform design, 2) agent network design, and 3) agent design” (Iglesias, Garijo et. al 1998) that are composed in textual template as shown in Figure 28 (Iglesias, Garijo et al., 1998).

Development and **testing**. This phase relates to coding and testing activities of the earlier defined agents. (Iglesias, Garijo et al., 1998)

Operation. Maintenance and operation of the system (Iglesias, Garijo et al., 1998).



Platform *Collaborator*
  Languages: Java, Jess
  Software: JBoss, JADE, Jess

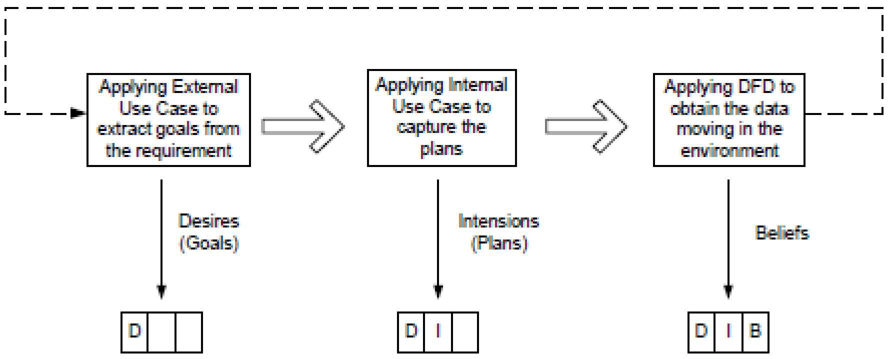
Network *Collaborator*
  Network-services – FIPA Standard (AMS, DF)
  Knowledge-services: none
  Coordination services: Resource Manager Service, Session Manager Agent

Agent System *Session ManagerAgent*
Subsystems:
  user-interaction *SMAInterface* (tasks 1, 5.1; communication model)
  reasoning *SMAKB* tasks 2,3, 4.2, 5.2, expertise model
  agent-interaction *CoordinatorBehaviour* (task 4.1 and 5.2), main-task *SMABehaviour* (Coordinating task).

Figure 28 Design textual templates for Session Manager Agent (adopted from (Iglesias and Garijo, 2005))

The key strengths of the MAS-CommonKADS methodology are its simplicity, and its ability to extend the standard software engineering practices. In addition, MAS-CommonKADS has reusability feature in the models, making it easy for reuse of entities produced from the analyses and designs phases. The main weakness of the methodology is its limited support in design, testing, and coding. Two tools are capable of developing entities based on MAS-CommonKADS methodology, these are: a “Java CASE tool and a Plug-In for Rational Rose” (Iglesias and Garijo, 2005)

3.8Methodology for BDI agents



Applying External Use Case to extract goals from the requirement
Applying Internal Use Case to capture the plans
Applying DFD to obtain the data moving in the environment
Desires (Goals)
Intensions (Plans)
Beliefs
D
D I
D I B

Figure 29 A new Approach to the BDI Discovery (adopted from CHANG, CHEN et al., 2004)

Figure 29 shows a new approach to discover Beliefs Desires and Intentions from the requirements phase. The desire is the first one found from the system requirements and then its intention and corresponding beliefs are found. The central idea originates from the natural way of doing in the real world. In reality, people initially establish the goals in the mind. To achieve these goals, people work out correspondent plans. For the sake of accomplishing the plans, people perform some actions. These requires people to communicate with their environment, or in other worlds, people need to know the data to describe the world

By following the natural way of human thought: goal-plan-data, our approach will first extract the desires from the requirement, and then create proper intentions. Finally, beliefs are found.

Next step, is using the “Agent Models (namely Role Model such as External Use Cases) to extract goals (belief) from the requirements, the Dynamic Model (namely Coordination Model - Communication and Negotiation Model such as Internal Use Cases, Sequence Diagrams, and Activity Diagrams) to capture the plans (intention), and the Data Model (such as Data Flow Diagrams) to obtain the data (belief) moving in the environment”. The high-level view of developing BDI agent is depicted in Figure below.

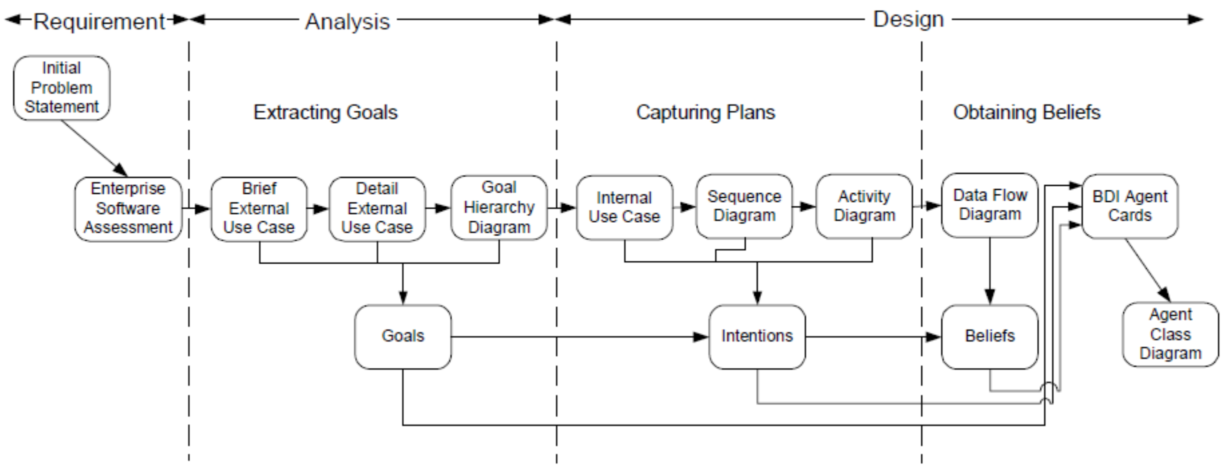


Figure 30 BDI Agent Development Process (adopted from (CHANG, CHEN et al., 2004))

3.8.1 BDI Agent-based Software Process (BDI ASP)

In the analysis phase Fig 30 above, I use “External Use Case to extract the desires from the external point of view” (CHANG, CHEN et al., 2004). This is used to fully understand the requirement and prepare for the design phase. In, the design phase, I utilize the Internal Use Case to capture the intentions and Data Flow Diagram to find the beliefs. After I discover the beliefs, desires, and intentions, I bring them together to “form the BDI Agent Cards” (CHANG, CHEN et al.,2004).

The BDI Agent Software Development Process (BDI-ASDP) consists of ten steps which are: 1) Initial Problem Statement,2) Enterprise Software Assessment, 3) Brief External Use Case, 4) Detailed External Use Case, 5) Structuring Goals, 6) Internal Use Case, 7) Sequence Diagram, 8) Agent Activity Diagram, 9) Data Flow Diagram, and 10) BDI Agent Cards. (CHANG, CHEN et al., 2004). The BDI-ASPD is a specialization of traditional software engineering methodologies customized for agent development.

Alternatively, as shown in Fig 31, looking at the above model from another BDI perspective, a MAS system consists of "cooperative and communicating agents" (Hyun Jo, Chang et al., 2005). An agent is described by a collection of BDI. Two steps are used to model the system by using the BDI agents, which are: "(I) Analysis: In this step, various artifacts are used including use cases to find BDIs and agents in the system; (II) Design: In this step BDIs are assigned to appropriate agents in the system"

Figure 31 shows a high-level overview of BDI which is used in the agent-based development process. It describes a general approach to discover agent BDI attributes in the BDI ASP agent software development process.

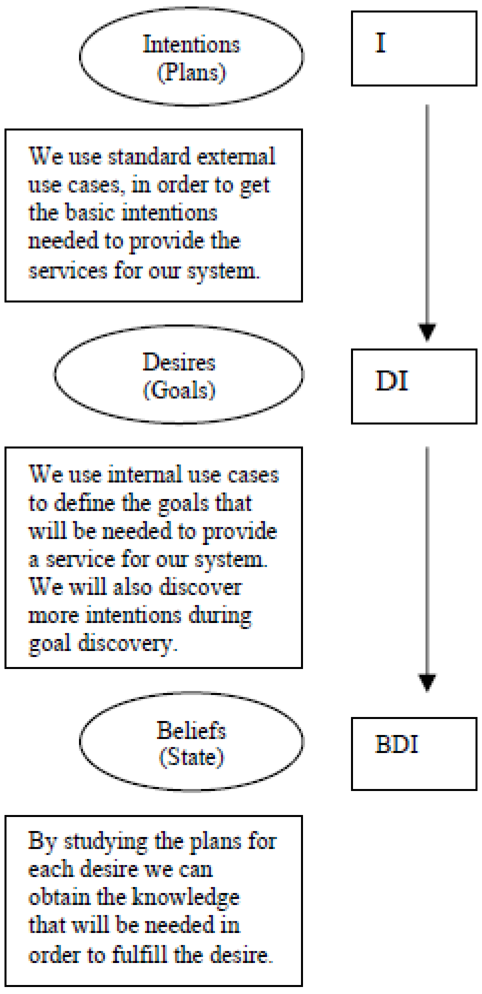

Intentions (Plans)
I
We use standard external use cases, in order to get the basic intentions needed to provide the services for our system.
Desires (Goals)
DI
We use internal use cases to define the goals that will be needed to provide a service for our system. We will also discover more intentions during goal discovery.
Beliefs (State)
BDI
By studying the plans for each desire we can obtain the knowledge that will be needed in order to fulfill the desire.

Figure 31 BDI Discovery in the BDI Agent Software Process (adopted from (Hyun Jo, Chang et al., 2005))

In the initial stage of the development process, external use cases are developed. This is a common strategy that indicates how a specific service provides an external point of view. These are later on refined into goals using earlier designed internal use cases. "The internal use cases decompose a service into one or more goals" (Hyun Jo, Chang et al.,2005). "The internal use cases provide a more precise description of each goal and its corresponding plan. Once a goal has been discovered and a plan described for each goal the beliefs are discovered. The beliefs are determined for each goal by analyzing each goal's plans and determining what beliefs is

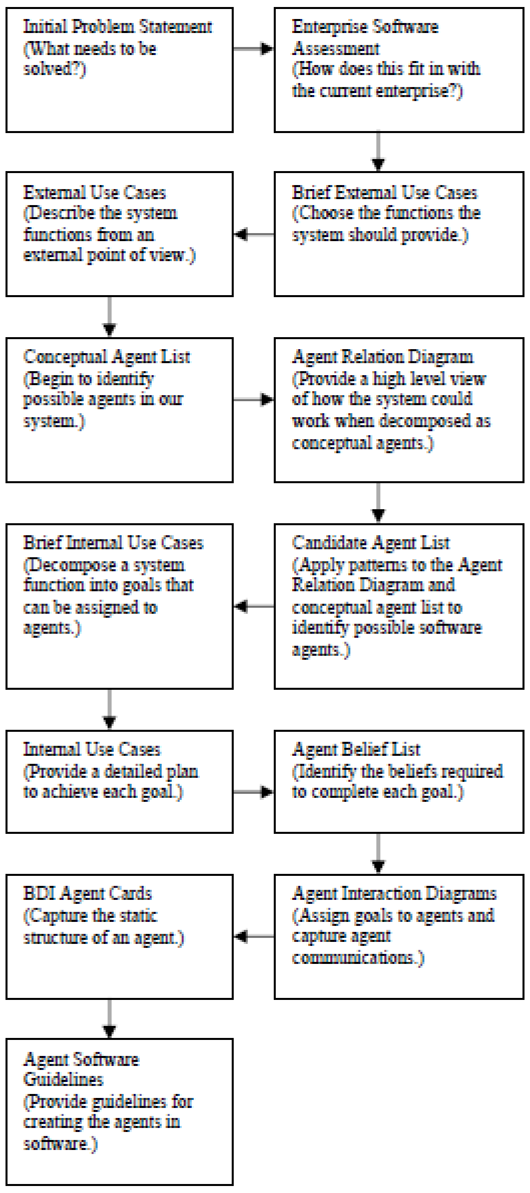

Initial Problem Statement (What needs to be solved?)
Enterprise Software Assessment (How does this fit in with the current enterprise?)
Brief External Use Cases (Choose the functions the system should provide.)
External Use Cases (Describe the system functions from an external point of view.)
Conceptual Agent List (Begin to identify possible agents in our system.)
Agent Relation Diagram (Provide a high level view of how the system could work when decomposed as conceptual agents.)
Candidate Agent List (Apply patterns to the Agent Relation Diagram and conceptual agent list to identify possible software agents.)
Brief Internal Use Cases (Decompose a system function into goals that can be assigned to agents.)
Internal Use Cases (Provide a detailed plan to achieve each goal.)
Agent Belief List (Identify the beliefs required to complete each goal.)
Agent Interaction Diagrams (Assign goals to agents and capture agent communications.)
BDI Agent Cards (Capture the static structure of an agent.)
Agent Software Guidelines (Provide guidelines for creating the agents in software.)

Figure 32 BDI Agent Software Development Process (adopted from (Hyun Jo, Chang et al., 2005))

necessary for its completion" (Hyun Jo, Chang et al., 2005). Once a complete BDI is described, it is assigned to an agent. Figure 32 shows the artifacts that are created during the "BDI agent development process" (Hyun Jo, Chang et al., 2005). The displayed arrows illustrate the broad order of construction of the artifacts in this process. There is no rule of the order of steps for creating artifacts. The depicted arrows represent an unconfirmed order that is recommended for artifact creation.

4 CASE STUDIES

In this chapter, I will provide few case studies for few important methodologies as shown in figure 33 and table 4. Below is the brief of the case studies that I will detail in the following

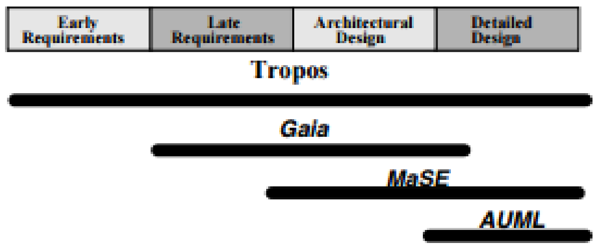


Figure 33 Comparison of Tropos with other software development methodologies (adopted from (Giunchiglia, Mylopoulos et al., 2003))

Table 4 : Case studies listing for methodologies

Sl. No	**Methodology / Case study name**	**Associated content of case study**
1	TROPOS / The Conference Management System Case Study (Morandini, Nguyen et al., 2017).	Tropos and related tools like TAOM4e is used for developing the Conference management system that has actors, goals and related relationships
2	GAIA / Airport services (Sánchez-Pi, Carbó et al., 2010)	GAIA approach is used to address the problem of using context-aware information to provide customized services to airport users in "Barajas-Madrid Airport and also for services offered to passengers, clients, crew and staff of IBERIA airline" (Sánchez-Pi, Carbó et al., 2010)
3	MaSE / Mobile Search System (MSS) (Self and DeLoach, 2003)	MaSE is used for developing agents that determine when it needs to take the next move. Although there might be request from other agents, or the agent platform itself, for the movement, however the autonomous nature of agents makes it possible to determine whether it will actually move or not.

Details of each case study is described in the following sections.

4.1 Case study one: TROPOS / The Conference Management System (CMS) Case Study

The CMS development process is model-driven, meaning, requirements and design models are core entities. These are developed from a conceptual modelling language, derived from the "i* framework" (Morandini, Nguyen et.al., 2017). This modelling activity, called Agent- Oriented (AO) modelling as shown in Fig. 35. This figure shows the first four phases of the software development process. "The entities of the modelling language are actors, goals, plan and dependency that help in goal achievement. AO modelling is conducted using the TAOM4e modelling tool This tool has extension that provides automatic code generation from the Tropos requirement to JADE or Jadex. This is made possible, by a mapping from the Tropos meta-model

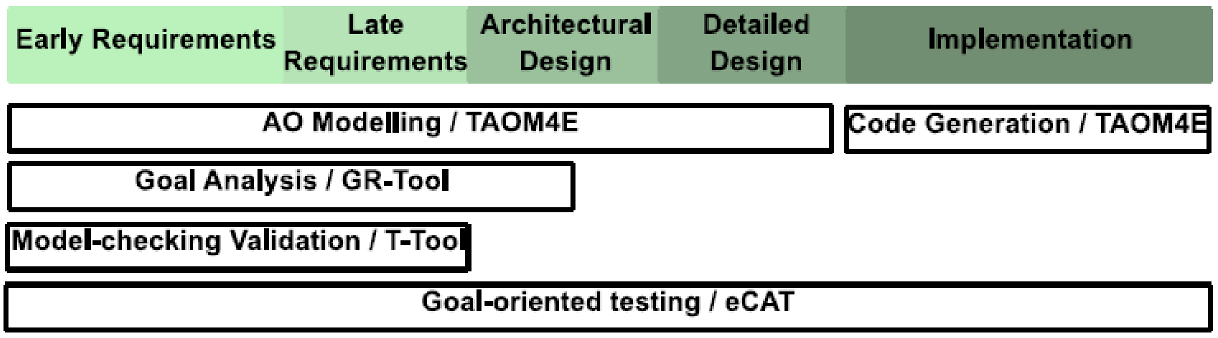


Figure 34 Development Process Phases: Activities and Supporting Tools (adopted from (Morandini, Nguyen et al., 2007))

concepts to the target languages constructs." (Morandini, Nguyen et al., 2017)

"TAOM4e is a framework for graphical Tropos modeling" (Morandini, Nguyen et al., 2017). This supports modeling for all the phases of the Tropos process. It is a plug-in extension available as part of the Eclipse framework along with other plug-ins, as shown in Fig. 36. EMF plug-in provides a modelling framework and code generation feature for developing tools. All plug-ins are available at Eclipse website.

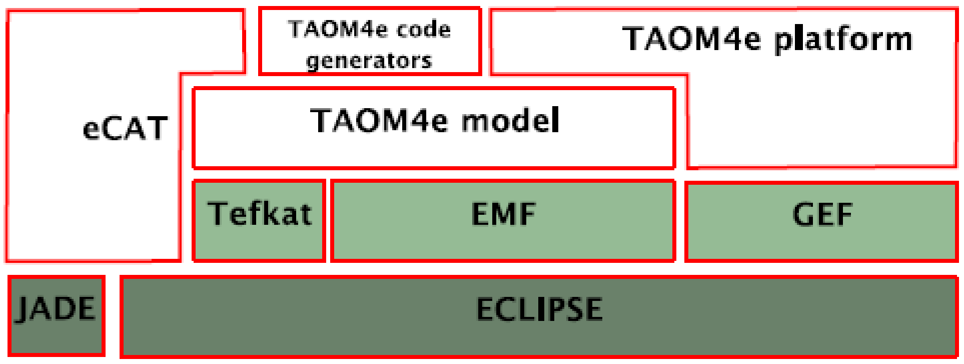


Figure 35 Architecture of TAOM4e and eCAT (adopted from (Morandini, Nguyen et al., 2007))

The Graphical Editing Framework (GEF) plug-in makes it possible to create a graphical editor from a previously created application model; the Tefkat plug-in has a rule-based language that helps to develop cross-model transformation. (Morandini, Nguyen et.al., 2017)

As shown in Fig 36, the Tropos metamodel is placed above EMF (TAOM4e model). GEF that is a sibling to EMF, is used to develop the graphical representation of the model along with various views on top of it (TAOM4e platform). The Tefkat plug-in which is also a sibling to EMF transforms the top-level plans and their decompositions to UML artefacts which are activity diagrams. The resulting diagrams are compatible with most UML editors and can be further enhanced with sequence diagrams, that provide communication protocols for agents. Fig. 37 denotes the display screen of the "TAOM4e GUI" (Morandini, Nguyen et al., 2007).

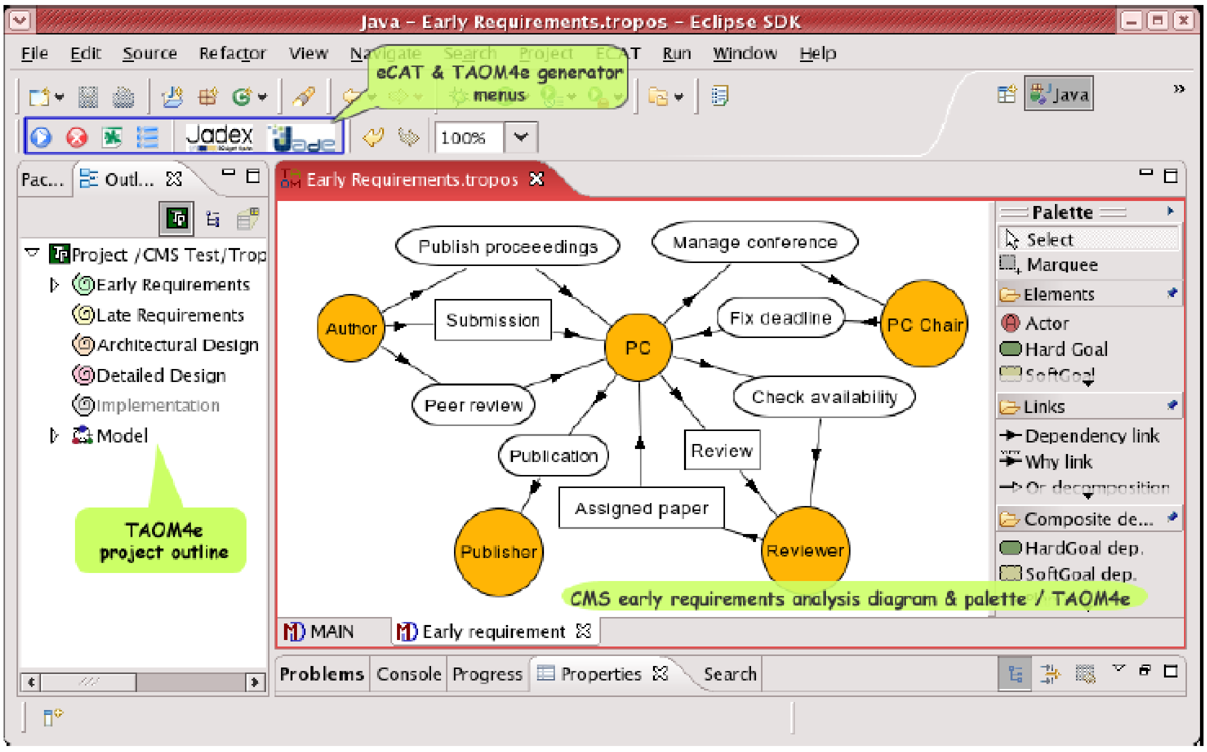


Figure 36 A snap view of the "TAOM4e's GUI" (adopted from (Morandini, Nguyen et al., 2007))

4.1.1 Requirements Analysis

The requirements analysis phases as per Tropos has two models which are Early Requirements and Late Requirements.

The Conference Management System is modeled in terms of its key stakeholders (a.k.a. actors). The actors are identified as authors, papers, PC Chair actors, paper reviewers, the conference's program committee and its chair, and the proceedings publisher. Once the actors are identified then the stakeholders' goals are identified. (Morandini, Nguyen et al., 2017) After this for each goal, the developer, on the basis of the domain, can decide if the goal is achievable or if there is a need to delegate it to another actor, which shows the dependency relationship between the two actors. An example for this is the dependency between Author and PC to achieve the goal Publish proceedings

Fig. 38, shows an Early Requirements goal diagram. This diagram denotes a (partial) view of the model. This diagram shows two actors, namely PC and PC Chair. These are represented with two goal dependencies which are Manage conference and Decide deadlines.

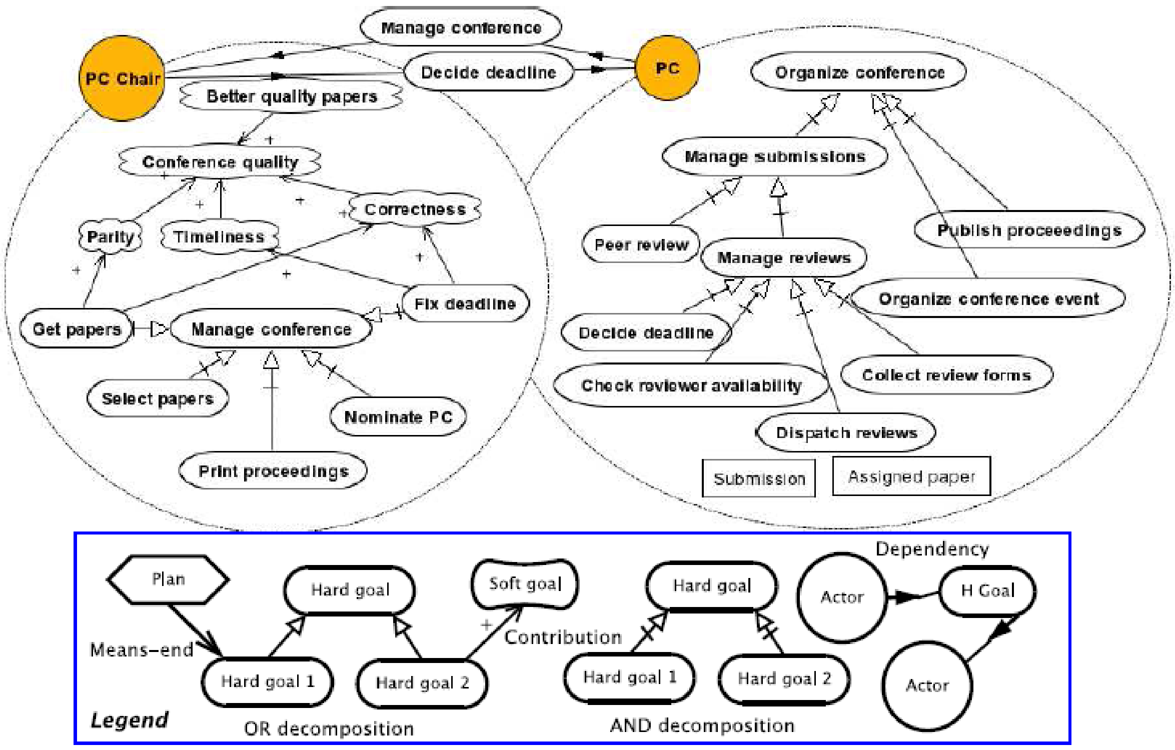


Figure 38 Early Requirements of CMS: Goal Diagram (adopted from (Morandini, Nguyen et al., 2007))

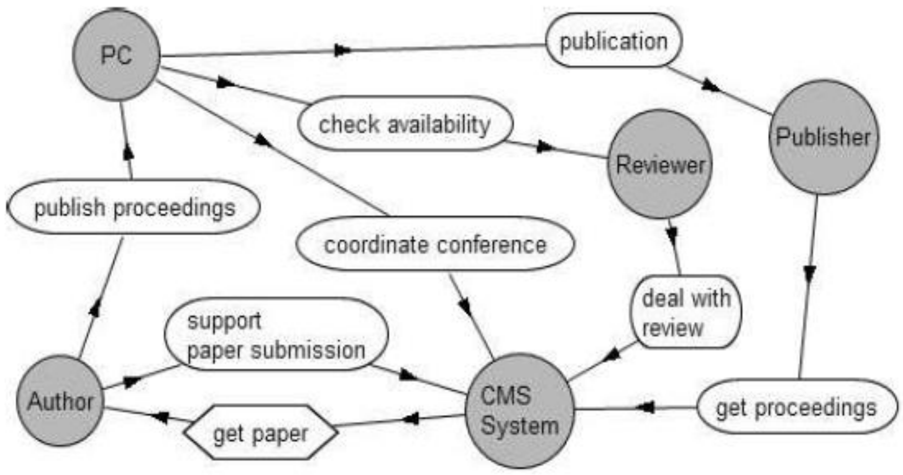


Figure 37 Late Requirements of CMS: Actor Diagram (adopted from (Morandini, Nguyen et al., 2007))

The Late Requirements phase is meant to obtain the changes in the CMS domain caused by the addition of the target system and the definite properties of the system. A partial view of the resulting model for CMS is shown in Fig. 37.

4.1.2 Design

The system's overall structure is captured in the Architectural Design artifact. It is represented by its sub-systems and their inter-dependencies. TAOM4e has the capability to generate an "Architectural Design diagram for every system actor designed in the Late Requirements Analysis" (from Morandini, Nguyen et al., 2007).

Fig. 38 illustrates the ensuing architectural design diagram for the System actor. By analyzing the actor's goal model (see fig. 37 and 38), the developer should be able to extract a proper decomposition into sub-actors.

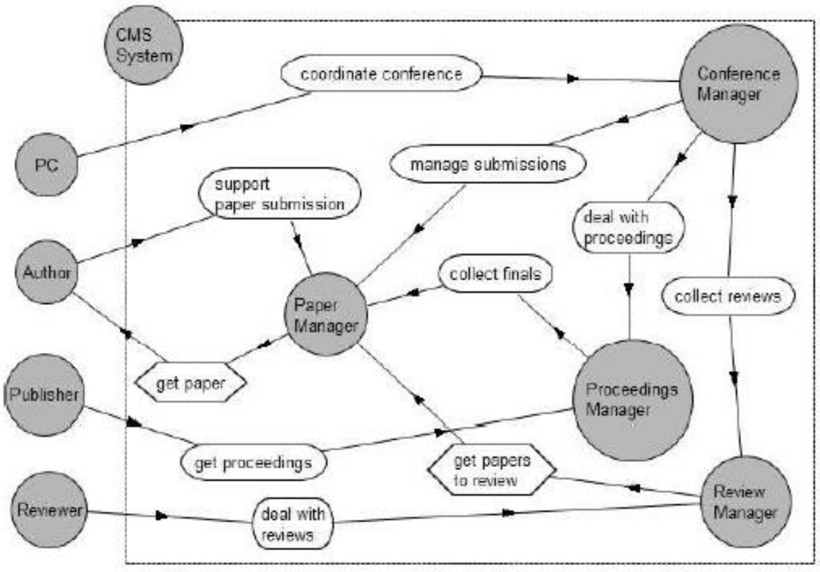
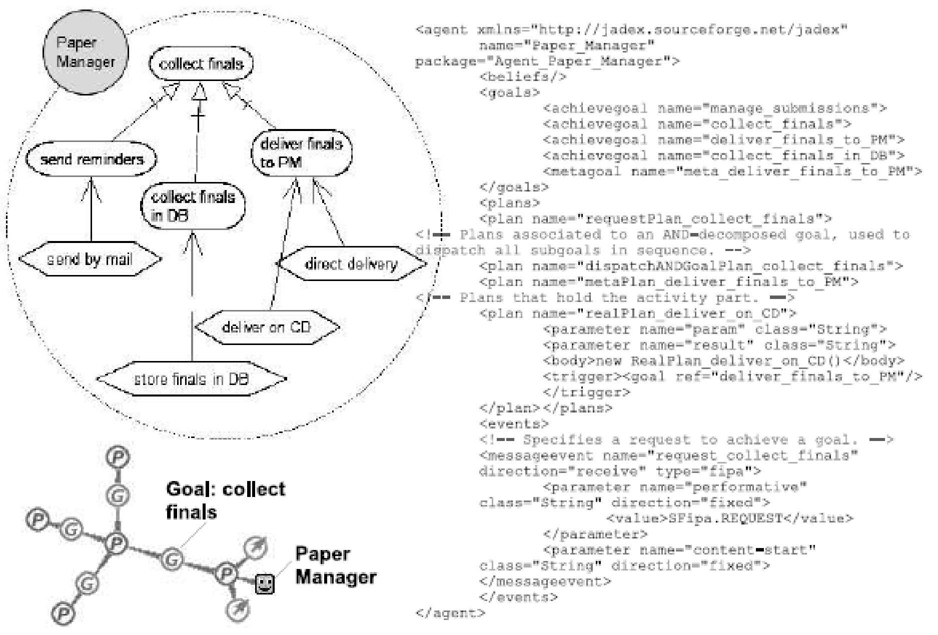


Figure 39 Architectural Design: CMS System Decomposition into Sub-actors (adopted from (Morandini, Nguyen et al., 2007))

4.1.3 Code and Test Suites Generation

The desired software agents form the basis for the goal models created in the design phase. T2x tool provides features to generate the target code, in this case it is "Jadex agent definition files" (Morandini, Nguyen et al., 2017). These definition files can be generated by choosing a system agent in the GM and using menu to start the automatic file generation process

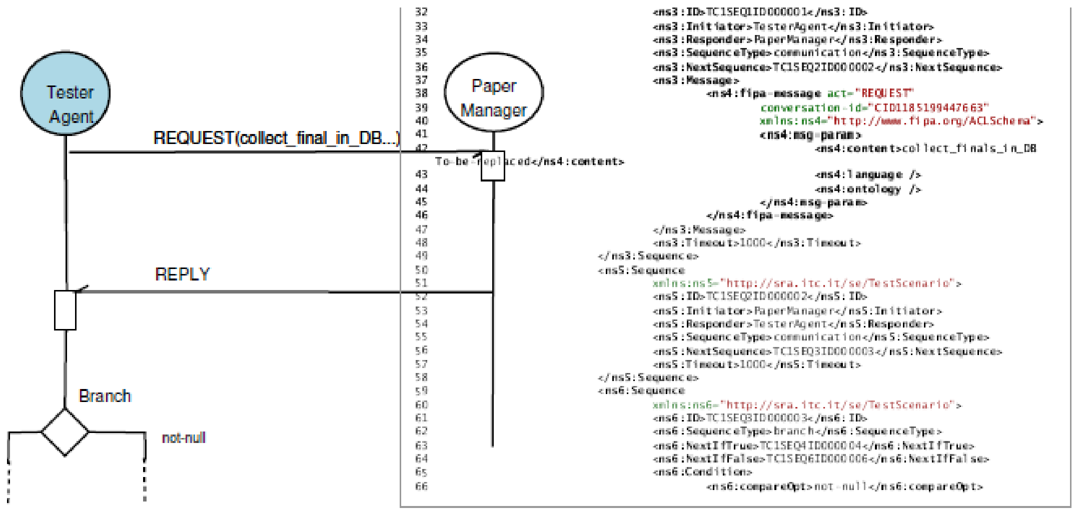


```
<agent xmlns="http://jadex.sourceforge.net/jadex"
         name="Paper_Manager"
package="Agent_Paper_Manager">
         <beliefs/>
         <goals>
                  <achievegoal name="manage_submissions">
                  <achievegoal name="collect_finals">
                  <achievegoal name="deliver_finals_to_PM">
                  <achievegoal name="collect_finals_in_DB">
                  <metagoal name="meta_deliver_finals_to_PM">
         </goals>
         <plans>
         <plan name="requestPlan_collect_finals">
<!-- Plans associated to an AND-decomposed goal, used to
dispatch all subgoals in sequence. -->
         <plan name="dispatchANDGoalPlan_collect_finals">
         <plan name="metaPlan_deliver_finals_to_PM">
<!-- Plans that hold the activity part. -->
         <plan name="realPlan_deliver_on_CD">
                  <parameter name="param" class="String">
                  <parameter name="result" class="String">
                  <body>new RealPlan_deliver_on_CD()</body>
                  <trigger><goal ref="deliver_finals_to_PM"/>
                  </trigger>
         </plan></plans>
         <events>
         <!-- Specifies a request to achieve a goal. -->
         <messageevent name="request_collect_finals"
         direction="receive" type="fipa">
                  <parameter name="performative"
         class="String" direction="fixed">
                           <value>SFipa.REQUEST</value>
                  </parameter>
                  <parameter name="content-start"
         class="String" direction="fixed">
         </messageevent>
         </events>
</agent>
```

Figure 40 Goal diagram and Jadex code for CMS System (adopted from (Morandini, Nguyen et al., 2007))

For example, Fig. 40 shows the generated Jadex code, for agent Paper Manager which is in XML format (Morandini, Nguyen et.al., 2017)

For test example, Fig. 41 depicts a test suite that verifies if the agent Paper Manager is able to accomplish the goal collect finals in DB. Figure 42 shows the test logic in XML. "For executing the test, the Autonomous Tester Agent sends a request that has "REQUEST" as its test function with parameters with the name of the goal collect finals in DB as message content to Paper Manager" (Morandini, Nguyen et.al 2017)., It will then wait for a reply and based on the return value will make a decision whether to stop the test or to carry on with other requests.



```
        <ns3:ID>TC1SEQ1ID000001</ns3:ID>
        <ns3:Initiator>TesterAgent</ns3:Initiator>
        <ns3:Responder>PaperManager</ns3:Responder>
        <ns3:SequenceType>communication</ns3:SequenceType>
        <ns3:NextSequence>TC1SEQ2ID0000002</ns3:NextSequence>
        <ns3:Message>
            <ns4:fipa-message act="REQUEST"
                conversation-id="CID118519944766 3"
                xmlns:ns4="http://www.fipa.org/ACLSchema">
                <ns4:msg-param>
                    <ns4:content>collect_finals_in_DB
To be replaced</ns4:content>
                    <ns4:language />
                    <ns4:ontology />
                </ns4:msg-param>
            </ns4:fipa-message>
        </ns3:Message>
        <ns3:Timeout>1000</ns3:Timeout>
    </ns3:Sequence>
    <ns5:Sequence
        xmlns:ns5="http://sra.itc.it/se/TestScenario">
        <ns5:ID>TC1SEQ2ID000002</ns5:ID>
        <ns5:Initiator>PaperManager</ns5:Initiator>
        <ns5:Responder>TesterAgent</ns5:Responder>
        <ns5:SequenceType>communication</ns5:SequenceType>
        <ns5:NextSequence>TC1SEQ3ID0000003</ns5:NextSequence>
        <ns5:Timeout>1000</ns5:Timeout>
    </ns5:Sequence>
    <ns6:Sequence
        xmlns:ns6="http://sra.itc.it/se/TestScenario">
        <ns6:ID>TC1SEQ3ID000003</ns6:ID>
        <ns6:SequenceType>branch</ns6:SequenceType>
        <ns6:NextIfTrue>TC1SEQ4ID000004</ns6:NextIfTrue>
        <ns6:NextIfFalse>TC1SEQ6ID0000006</ns6:NextIfFalse>
        <ns6:Condition>
            <ns6:compareOpt>not-null</ns6:compareOpt>
```

Figure 41 Example of Test Scenario for CMS System (adopted from (Morandini, Nguyen et al., 2007))

4.2 Case study two: Airport services

This case study is related to design of services for an airport using GAIA. "Madrid-Barajas airport domain has six different zones that have unique non-overlapping requirements. These are 1) Airport Zone (hosts services of all zones), 2) Customs Zone (for customs services), 3) Commercial Zone (provides services for the stores, cafeterias and restaurants), 4) Offices Zone (services for the airline service provider, here IBERIA airline), 5) Check-In Desk Zone (services for the check-in desks) and 6) Finger Zone (service for biometrics like finger printing)." (Sánchez-Pi, Carbó et al., 2010)

The GIAA based design has identified different agent types which are: Client, Provider and Central, agents. The Central agent denotes the infrastructure which captures location information from various sensors. The Provider agent acts on behalf of various services; Client agent denote users with wireless devices. These are shown in Fig.42

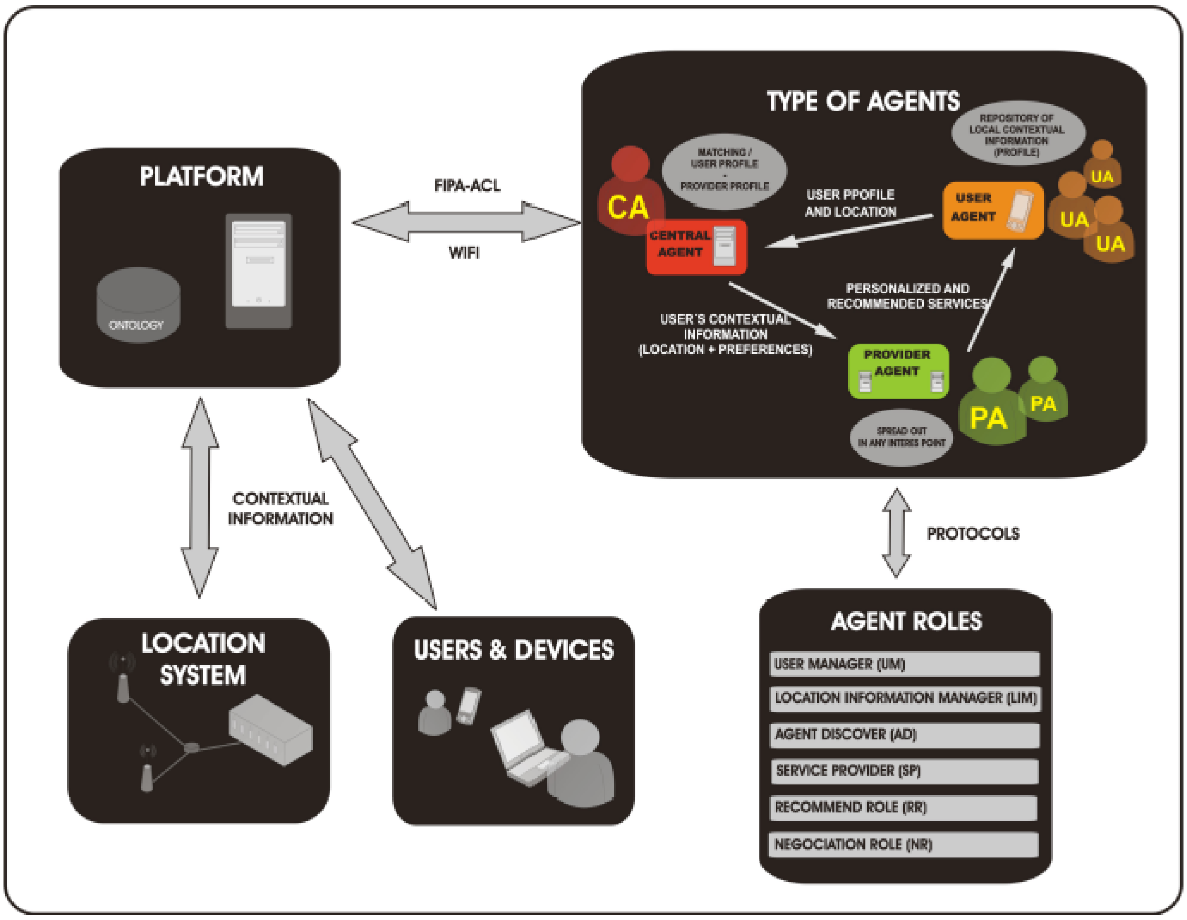

PLATFORM
ONTOLOGY
FIPA-ACL
WIFI
TYPE OF AGENTS
MATCHING / USER PROFILE - PROVIDER PROFILE
REPOSITORY OF LOCAL CONTEXTUAL INFORMATION (PROFILE)
CA
CENTRAL AGENT
USER PPOFILE AND LOCATION
USER AGENT
UA
UA
UA
PERSONALIZED AND RECOMMENDED SERVICES
USER´S CONTEXTUAL INFORMATION (LOCATION + PREFERENCES)
PROVIDER AGENT
PA
PA
SPREAD OUT IN ANY INTERES POINT
CONTEXTUAL INFORMATION
PROTOCOLS
LOCATION SYSTEM
USERS & DEVICES
AGENT ROLES
USER MANAGER (UM)
LOCATION INFORMATION MANAGER (LIM)
AGENT DISCOVER (AD)
SERVICE PROVIDER (SP)
RECOMMEND ROLE (RR)
NEGOCIATION ROLE (NR)

Figure 42 MAS system Architecture for airport (adopted from (Sanchez-Pi, Carbó et al., 2010))

4.2.1 Analysis and Design using GIAA Methodology

Gaia methodology has two key phases which are analysis phase, and design phase. The Gaia process begins with the start of analysis phase, that is used for collection and organization of the system specification that becomes the basis for the design of the organization. Further the extended Gaia version considers organizational structure and its rules. The analysis phase generates several models like: the preliminary role model, the environment model, the preliminary interaction model and organizational rules. The output of the analysis phase is used by design phase. The intention of the design phase is to convert the analysis models into an adequately low level of abstraction so that design techniques can be applied on this in order to realize agents. “The entities of the Gaia process are detailed but technology-neutral that is implemented using a suitable agent-programming framework such as Jadex platform. The architectural phase is about defining the system’s organizational constitution in the language of its topology along with details of the preliminary role, and interaction models. Finally, the detailed phase includes two models which are: (i) the agent model and (ii) the service model” (Sánchez-Pi, Carbó et al., 2010). These identify, the agent types present in the system and the key services needed to realize the agent's role.

4.2.2 The Environmental Model

The environment for the airline system is represented by an ontology as shown in Figure 44. This is required as there is a need to define the context knowledge for the communication

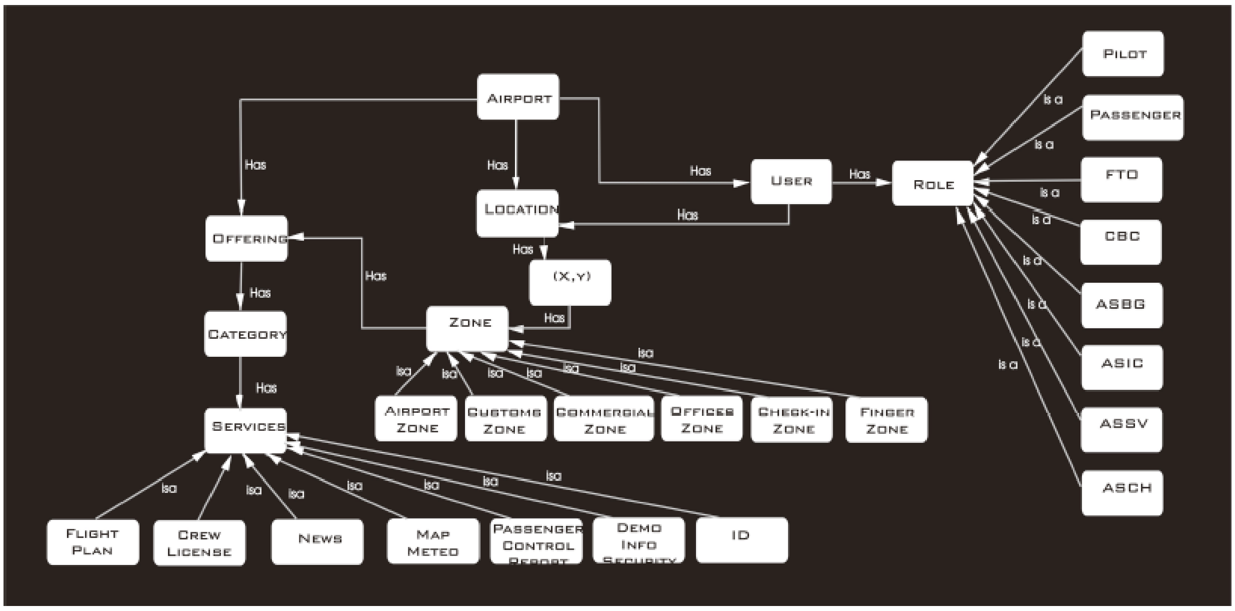


Figure 43 High level concepts of the generic ontology (adopted from (Sanchez-Pi, Carbó et al., 2010))

process between agents. The proposed ontology identifies the next high level concept

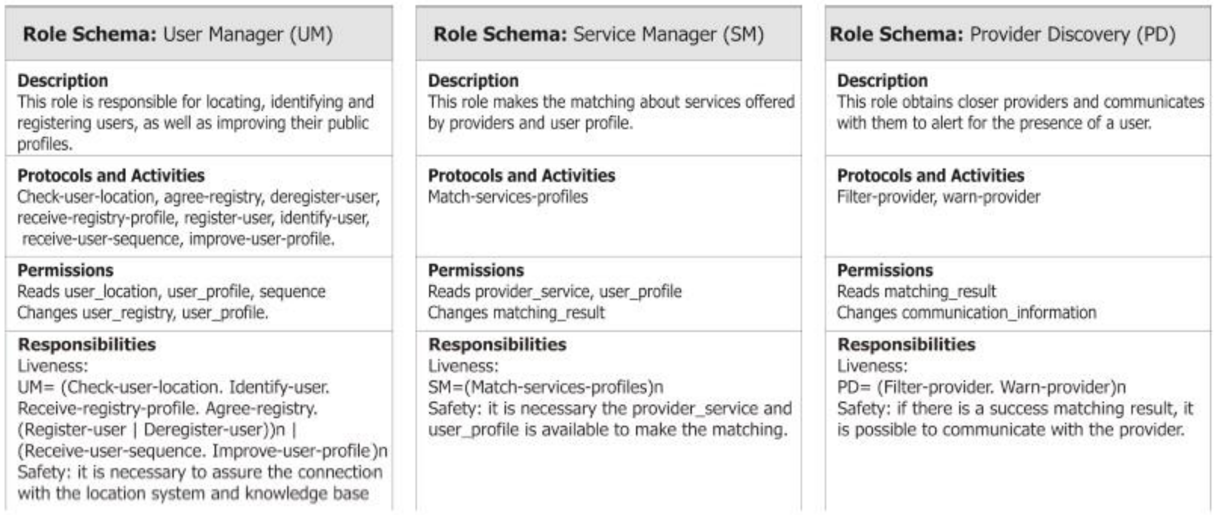
4.2.3 The Role Model

The role model gathers key roles played by the agents. A role is seen as a description of the agent functionality, and has four attributes which are: 1) permissions (these are restrictions on resources),2) responsibilities (normal behaviors of the agent role), 3) activities (actions completed without interaction between roles) and 4) protocols (meant for actions that has interaction between roles).

The following describes few roles identified for the multi-agent system:

- “Information Manager: this agent role is responsible for managing public data related to users’ profile and augmenting them with the environmental data. Also, this role manages information catalogue of services and products offered by airline service providers.
- User Manager: this role is responsible for coordinating all the activities of users and help them to locate, identify and manage (register/deregister) users.” (Sánchez-Pi, Carbó et al., 2010)

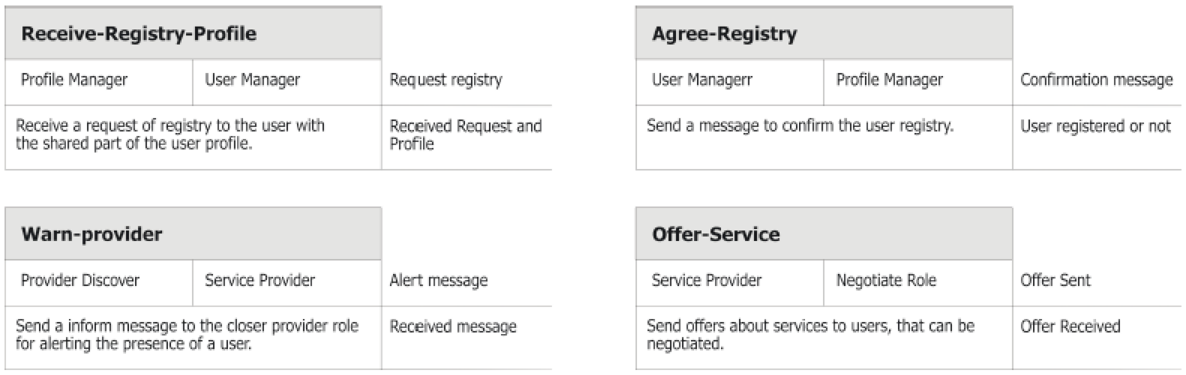
Details of the Role Model is as shown in fig 44



Role Schema: User Manager (UM)	**Role Schema:** Service Manager (SM)	**Role Schema:** Provider Discovery (PD)
Description This role is responsible for locating, identifying and registering users, as well as improving their public profiles.	**Description** This role makes the matching about services offered by providers and user profile.	**Description** This role obtains closer providers and communicates with them to alert for the presence of a user.
Protocols and Activities Check-user-location, agree-registry, deregister-user, receive-registry-profile, register-user, identify-user, receive-user-sequence, improve-user-profile.	**Protocols and Activities** Match-services-profiles	**Protocols and Activities** Filter-provider, warn-provider
Permissions Reads user_location, user_profile, sequence Changes user_registry, user_profile.	**Permissions** Reads provider_service, user_profile Changes matching_result	**Permissions** Reads matching_result Changes communication_information
Responsibilities Liveness: UM= (Check-user-location. Identify-user. Receive-registry-profile. Agree-registry. (Register-user | Deregister-user))n | (Receive-user-sequence. Improve-user-profile)n Safety: it is necessary to assure the connection with the location system and knowledge base	**Responsibilities** Liveness: SM=(Match-services-profiles)n Safety: it is necessary the provider_service and user_profile is available to make the matching.	**Responsibilities** Liveness: PD= (Filter-provider. Warn-provider)n Safety: if there is a success matching result, it is possible to communicate with the provider.

Figure 44 Role model for the airport domain (adopted from (Sanchez-Pi, Carbó et al., 2010))

4.2.4 Interaction Model



Receive-Registry-Profile		
Profile Manager	User Manager	Request registry
Receive a request of registry to the user with the shared part of the user profile.		Received Request and Profile

Agree-Registry		
User Managerr	Profile Manager	Confirmation message
Send a message to confirm the user registry.		User registered or not

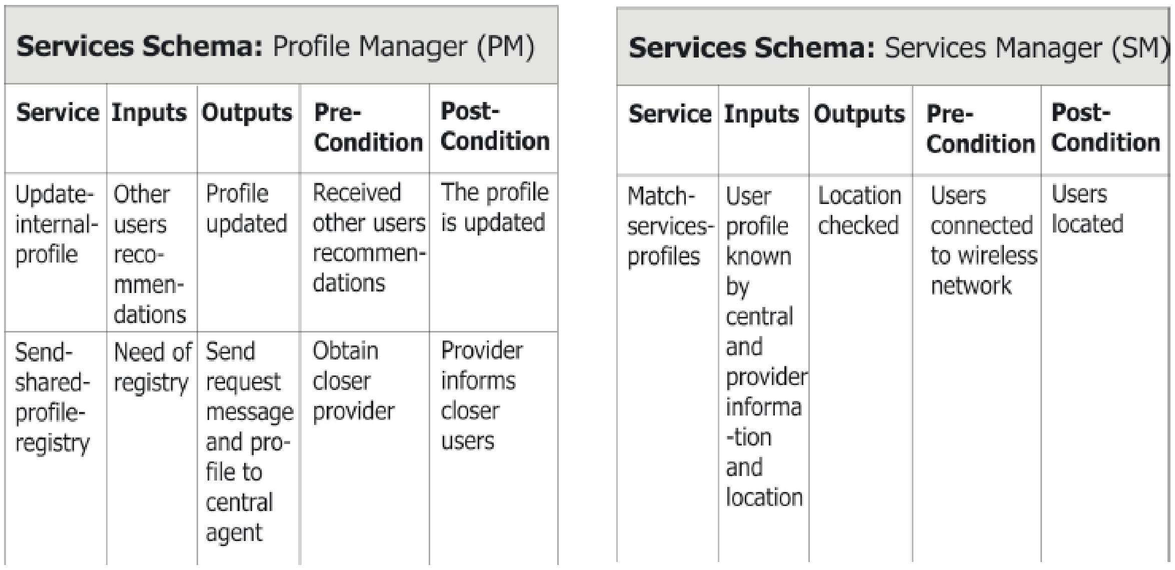
Warn-provider		
Provider Discover	Service Provider	Alert message
Send a inform message to the closer provider role for alerting the presence of a user.		Received message

Offer-Service		
Service Provider	Negotiate Role	Offer Sent
Send offers about services to users, that can be negotiated.		Offer Received

Figure 45 Interaction model for the airport domain (adopted from Sanchez-Pi, Carbó et al., 2010)

Interaction model is used to depict the dependencies and relationships among the agent roles in the MAS. fig 45 depicts few of the interactions.

4.2.5 The service model



Services Schema: Profile Manager (PM)

Service	Inputs	Outputs	Pre-Condition	Post-Condition
Update-internal-profile	Other users reco-mmen-dations	Profile updated	Received other users recommen-dations	The profile is updated
Send-shared-profile-registry	Need of registry	Send request message and pro-file to central agent	Obtain closer provider	Provider informs closer users

Services Schema: Services Manager (SM)

Service	Inputs	Outputs	Pre-Condition	Post-Condition
Match-services-profiles	User profile known by central and provider informa-tion and location	Location checked	Users connected to wireless network	Users located

Figure 46 Service model for the airport domain (adopted from Sanchez-Pi, Carbó et al., 2010)

Its represents the complete set of activities, responsibilities, protocols, and liveliness, associated with the roles / Fig 46 shows.

4.3 Case study three: MaSE / Mobile Search System (MSS)

Dynamic agent systems are good in solving problems like finding services and information from the Internet and aiding mobile clients.

Dynamic agent's area gents those that possess one of the below mentioned properties:

- "Cloning property: This is the ability of an agent to replicate an instance of self in the same or at a different location
- Instantiation property: This is the ability of an agent to replicate instances of a different class other than self
- Mobility property: This is the ability of an agent to move from one machine to another machine in a specified network" (Self and DeLoach, 2003)

In this example, MaSE is used to bring in Mobility for agents being developed. To incorporate mobility into MaSE, agent developer has the flexibility to use mobility to fulfill the required system goals. In the Analysis phase of MaSE, the Role diagram as shown in Figure 47

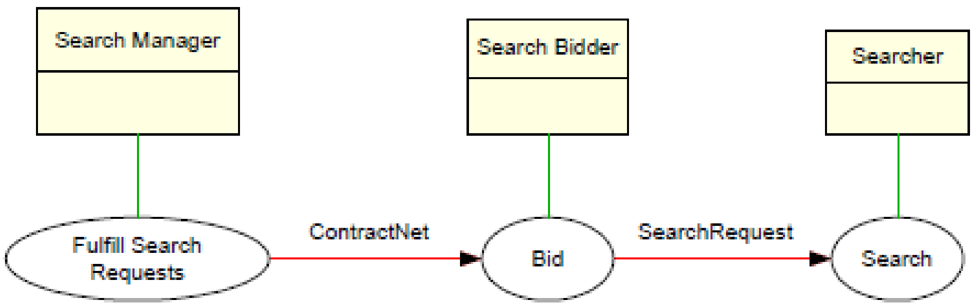


Figure 47 Role diagram for a Mobile Search System (adopted from (Self, DeLoach et al., 2003))

4.3.1 Analysis Phase

"Model for mobility is designed in the analysis phase by using a move activity for a defined state in the task diagram" (Self and DeLoach, 2003). This move activity generates and

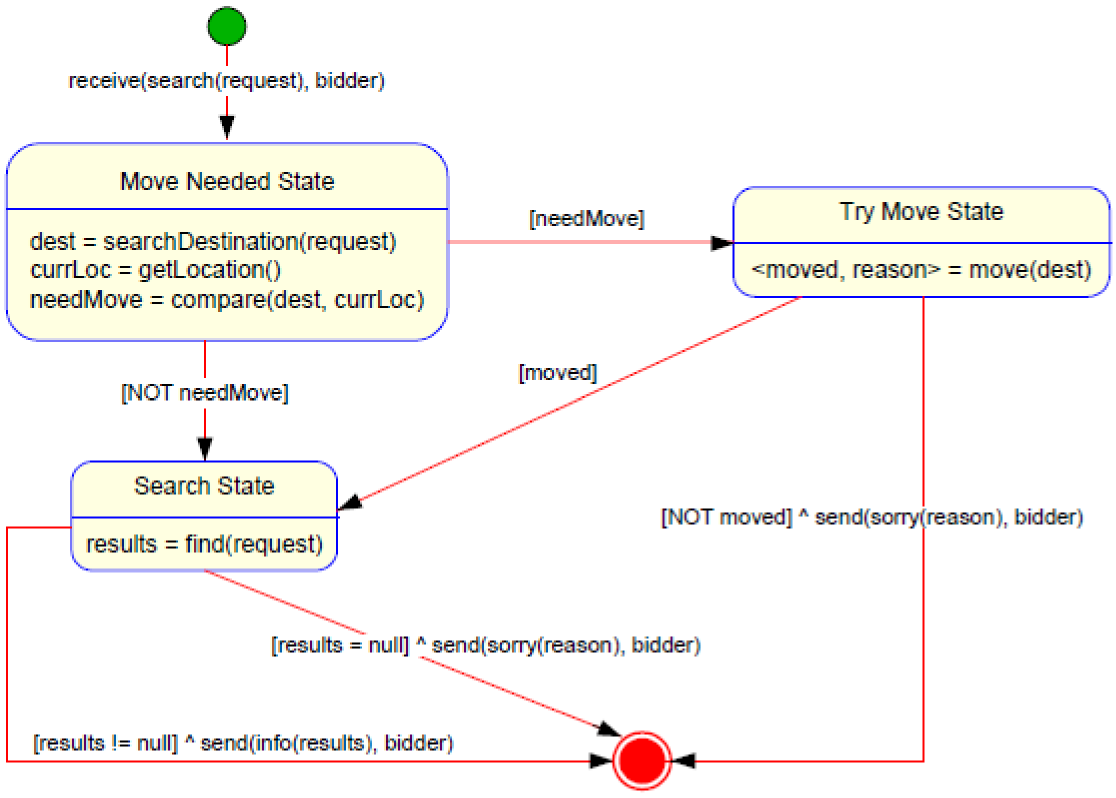

receive(search(request), bidder)
Move Needed State
dest = searchDestination(request)
currLoc = getLocation()
needMove = compare(dest, currLoc)
[needMove]
Try Move State
<moved, reason> = move(dest)
[NOT needMove]
[moved]
Search State
results = find(request)
[NOT moved] ^ send(sorry(reason), bidder)
[results = null] ^ send(sorry(reason), bidder)
[results != null] ^ send(info(results), bidder)

Figure 48 Search Task with Mobility (adopted from (Self, DeLoach et al., 2003))

returns exactly two return values which are: a reason value and a Boolean value. The Boolean value denotes the result of the move (which is either a success or a failure).

"When a move is desired, then the task is forwarded to next state that is the Try Move State. If the move becomes successful, then the task begins the search function. If the move is unsuccessful, then a sorry (reason) message is returned to the Bidder" (Self and DeLoach, 2003)

4.3.2 Design Phase

The design phase model comprises of the agent classes, the communications defined between those defined classes and the components that belong to those classes. These are as shown in Figure 49 and 50. The tasks generated from the analysis phase are then transformed

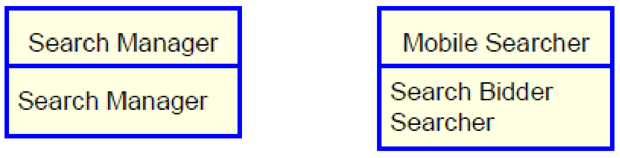


Figure 49 MSS Agent Classes (adopted from (Self, DeLoach et al., 2003))

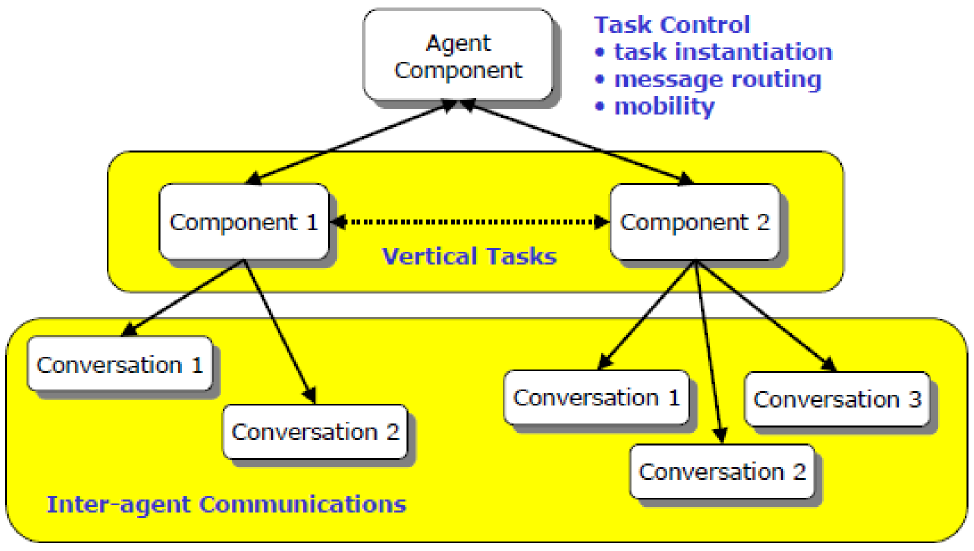
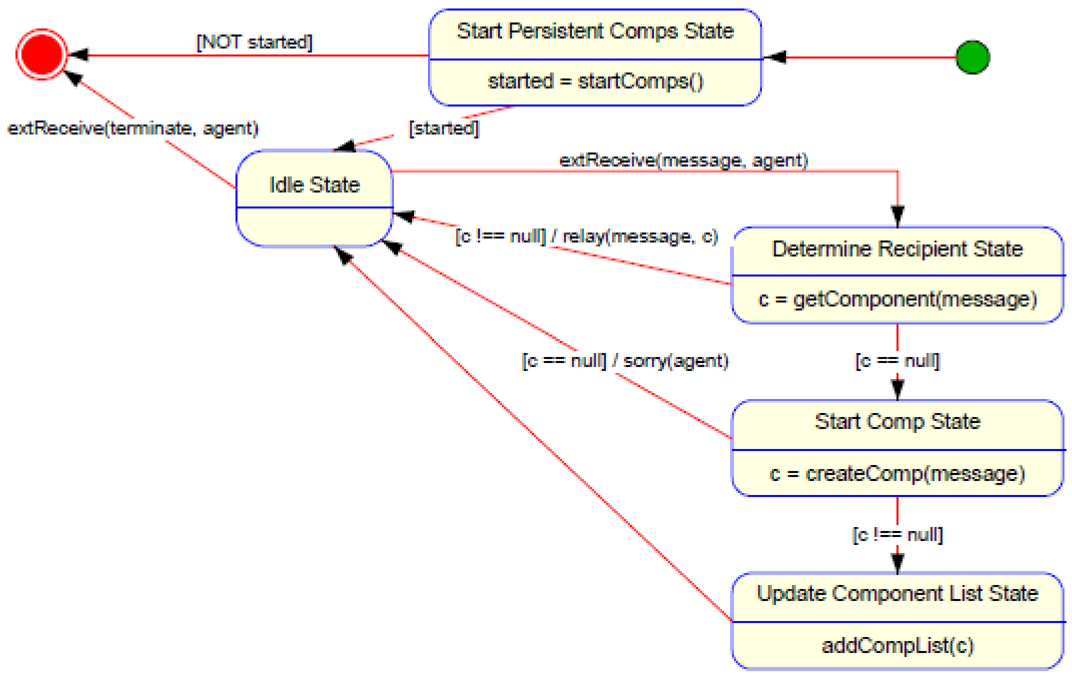


Figure 50 MaSE/agentTool Agent Architecture (adopted from (Self, DeLoach et al., 2003))

Using MaSE and agent Tool, agent components are derived from the tasks defined in the roles that are assigned to each agent class. Figure 50 shows the default agent design and architecture generated using the agent Tool.

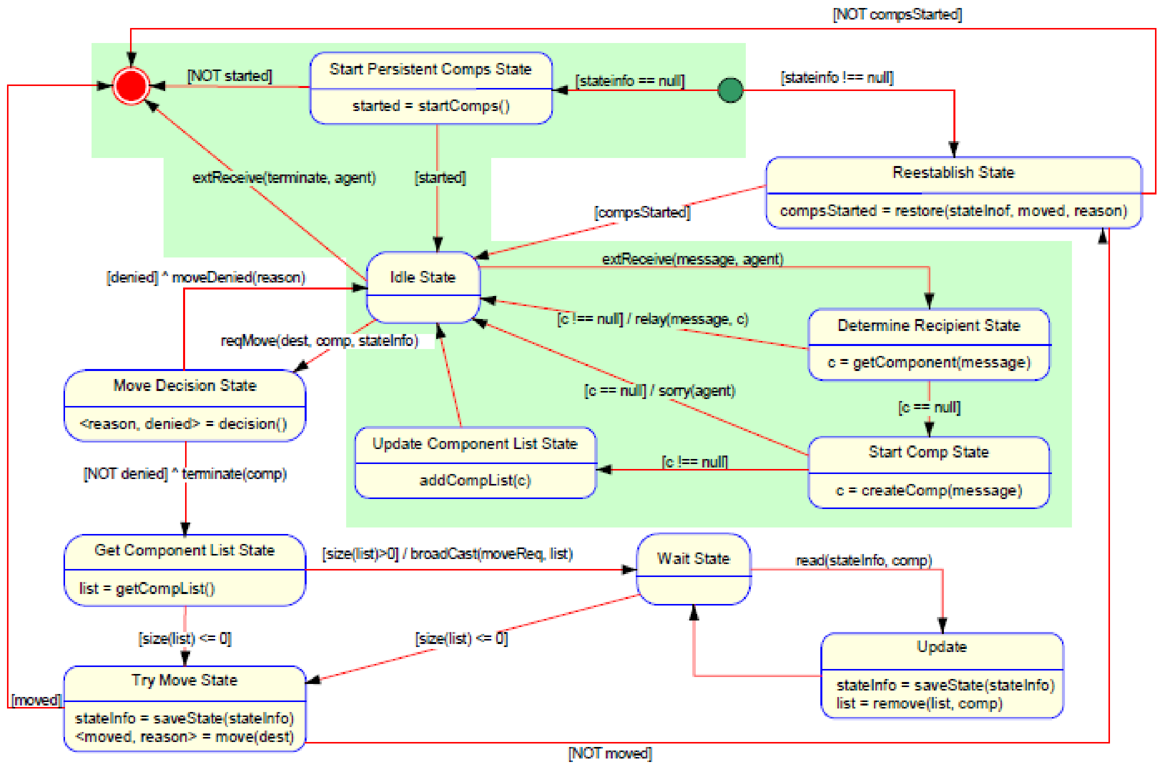
Figure 51 shows the “Mobile Searcher Agent Component. This contains both the persistent and non-persistent (transient) components.” (Self and DeLoach, 2003)



[NOT started]
Start Persistent Comps State
started = startComps()
extReceive(terminate, agent)
[started]
Idle State
extReceive(message, agent)
[c !== null] / relay(message, c)
Determine Recipient State
c = getComponent(message)
[c == null] / sorry(agent)
[c == null]
Start Comp State
c = createComp(message)
[c !== null]
Update Component List State
addCompList(c)

Figure 52 Agent Component for Mobile Searcher Agent Class (adopted from (Self, DeLoach et al., 2003))

Figure 52 shows the Agent Component of Figure 51 being transformed to handle the



[NOT compsStarted]
[NOT started]
Start Persistent Comps State
started = startComps()
[stateinfo == null]
[stateinfo !== null]
Reestablish State
compsStarted = restore(stateInof, moved, reason)
extReceive(terminate, agent)
[started]
[compsStarted]
extReceive(message, agent)
[denied] ^ moveDenied(reason)
Idle State
[c !== null] / relay(message, c)
Determine Recipient State
c = getComponent(message)
reqMove(dest, comp, stateInfo)
Move Decision State
<reason, denied> = decision()
[c == null] / sorry(agent)
[c == null]
Update Component List State
addCompList(c)
[c !== null]
Start Comp State
c = createComp(message)
[NOT denied] ^ terminate(comp)
Get Component List State
list = getCompList()
[size(list)>0] / broadCast(moveReq, list)
Wait State
read(stateInfo, comp)
[size(list) <= 0]
[size(list) <= 0]
Update
stateInfo = saveState(stateInfo)
list = remove(list, comp)
[moved]
Try Move State
stateInfo = saveState(stateInfo)
<moved, reason> = move(dest)
[NOT moved]

Figure 51 Agent Component for Mobile Searcher Agent Class with Mobility Functionality (adopted from (Self, DeLoach et al., 2003))

mobility requirements (existing states and transitions from Figure 51 are shown in the shaded area).

Getting mobility incorporated into the analysis phase of the MaSE methodology has been accomplished with above steps. Adding mobility to the design phase required “defining requirements and mapping those requirements to the different types of components” (Self, DeLoach et al., 2003), that included the agent component, which comprise a mobile agent class. A semi-automated transformation was generated in the agentTool environment as a proof... These transformations from agentTool are capable of producing the necessary conversations between agents and the internal design of the agent components.

5 RESEARCH REPORT - COMPARATIVE ANALYSES OF THE METHODOLOGIES

This chapter will delve into comparing and contrasting the key strengths and limitations of the popular methodologies. An impartial approach is used while considering the various aspects used during agent development. Several parameters like the applicability of methodology in the agent development lifecycle (SDLC), models designed, processes used, maturity in using modern software engineering techniques, suitability as specific to industrial applications, etc. are used

The feature analysis framework (Tran, Low et al., 2005) and a custom subset of features is used to compare and contrast and critically analyses the aspects.

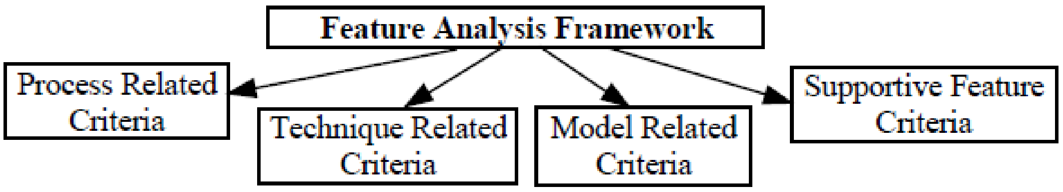

Feature Analysis Framework
Process Related Criteria
Technique Related Criteria
Model Related Criteria
Supportive Feature Criteria

Figure 53 Structure of the adopted "feature analysis framework" (adopted from (Tran, Low et al., 2005))

The structure of the "Analysis framework" (Tran, Low et al., 2005) is as shown in figure 53. Putting these adopted four components in my own words:

"Process based criteria's: This has criteria's that examines the maturity of a methodology's depth in the system-development processes.

Technique based criteria's: This uses criteria that examine the techniques used by the methodology to develop MAS.

Model based criteria's: This uses criteria that evaluates the capabilities of the models used by the methodology models.

Supportive Feature based criteria's: This uses criteria that evaluate various high-level capabilities available with the methodology." (Tran, Low et al., 2005)

Details of each criteria are depicted in the below table 5

5.1 Analysis of methodologies

Table 5 "Feature analysis framework" for examining agent-system-development methodologies (adopted from Tran, Low et al., 2005)

Sr. NO	PROCESS BASED CRITERIAS

1	**Development lifecycle:** This criteria looks into the development lifecycle used by the methodology (e.g. is it waterfall)?
2	**Coverage of the lifecycle:** This criteria looks into the phases of the lifecycle that are covered by the methodology (e.g. is it analysis, design, implementation…)?
3	**Development perspective**: This criteria looks into the development perspective supported by the methodology (i.e. is it topside-downside, bottom side-upside, or hybrid-mix)?
4	**Applications specific domain:** This criteria looks into how the methodology is applicable to a specific or multiple different application domains?
5	**Size of MAS:** This verifies the size of MAS the methodology and how it is suited for diff. sizes?
6	**Agent nature:** This looks into the methodology support for agent type (i.e. mixed-agents), or of a specific type (i.e. identical- agents)?
7	**Support for verification:** This criterion verifies if the methodology has rules that allow for the verification and validation of the developed models and specifications for correctness?
8	**Steps in the development process:** This criterion inspects the development steps that are supported by the methodology?
9	**Notational components:** This criterion documents the models and diagrams that are generated from the steps related to the process
10	**Comments about the strength and weakness of the steps:** This criteria provides the evaluator to document any comments for a process or step that is not able to record
11	**Easier to understand the process related steps:** This looks into the ease of understanding/ comprehend the process steps
12	**Usability of the methodology:** This verifies if the process steps are simple to pursue
13	**Input and output definition:** This verifies if the inputs and outputs to each process step are defined, and have examples?
14	**Refinability:** This verifies if the process related steps give a clear way to refine the models via. a gated SDLC phases in achieving the final deliverable, Alternatively, this

	looks into the minimum steps required for connecting the implementation level artefacts to the design specification
15	**MAS development approach:** This looks into the methodology's a. Common MAS development approach (e.g. is it OO-based or knowledge-engineering based)? b. Approach in using "role" in development process? c. Approach in role identification, in development?
	TECHNIQUE BASED CRITERIAS
16	**Availability of techniques and heuristics:** This verifies the techniques available a. What are the techniques used to perform individual process step? b. What are the techniques used to produce individual representable components (i.e. the modeling techniques used)
17	**Comments about the strengths and weaknesses of the techniques:** This criterion gives the examiner to note comments on the techniques used in each step to produce the models.
18	**Ease of understanding of techniques:** This verifies if the techniques are easy to understand?
19	**Usability of techniques:** This verifies if the techniques are easy to understand and use?
20	**Facility for demo and examples:** This verifies if examples and heuristics for the techniques are provided or not and if sufficient for the purpose?
	MODEL BASED CRITERIAS
21	**Concepts:** This verifies if the concepts of the models are capable of expressing?
22	**Expressiveness:** This looks into how well each model expresses the ideas? (e.g. is the model capable of shaping the idea to a great level of detail, or look from various angles?)
23	**Wholeness:** This verifies if all necessary agent-oriented concepts of the target MAS are captured by the models?
24	**Formalization/Preciseness of models:** This verifies if the notation (syntax) and semantics of models are clearly defined?

25	**Model derivation:** This verifies if there exists a clear process or logic and procedure for transforming models, or creating a model from the information available in another
26	**Consistency:** This looks into the points: a. If there are any rules and guidelines that ensures consistency between the abstraction levels inside each model (i.e. any internal consistency), and among different models? b. If the representations expressed are in a mode which provides consistency check among them?
27	**Complexity:** This verifies if there are a manageable number of concepts described for each and every model or diagram
28	**Simple to understand models:** Related to easiness in understanding of models
29	**Modularity:** This verifies if the methodology and its models provide enough support for modularity of agents?
30	**Abstraction:** This verifies if the methodology provides a way to generate models at different levels of abstractions?
31	**Autonomy:** This verifies if the models support and able to represent the autonomous attributes of agents (i.e. the ability to enact without aid of humans, and if the agents are capable of controlling their own states and actions)?
32	**Able to Adapt:** This verifies if the models support and represents the adaptability attribute of agents (i.e. the ability to discover and progress upon with the experience)?
33	**Cooperative behavior:** This verifies if the models bear and symbolize the cooperative conduct of agents (i.e. the capability to work together with other agents in achieving the goal)?
34	**Inferential capability:** This verifies if the models bear and symbolize the inferential potential of agents (i.e. the facility to take action on acts or tasks that are from abstract task specifications)?
35	**Communication ability:** This inspects if the models bears and symbolize mature communication ability (i.e. the ability of the agent to communicate with other agents using language similar to the human-like acts)?

36	**Personality:** This verifies if the models supports and represents the agent's personality (i.e. the ability to mimic attributes similar to human character)?
37	**Reactivity:** This verifies if the models supports and represents reactivity of the agent (i.e. the ability to sense and respond)?
38	**Temporal continuity:** This verifies if the models bears and symbolize agent info continuity feature (i.e. saving of model data like state and identity for long span of time)?
39	**Planned behavior:** This verifies if the models support and represents behavior like deliberation in agents (i.e. the ability to decide in a deliberative situation, or act proactively)?
40	**Concurrency:** This inspects if the methodology has models to capture concurrency (e.g. provision for representation of concurrent processes and synchronization of such concurrent processes)?
41	**Human Computer Interaction:** This looks into the models if they have provision to represent human users and how the user interface is designed?
42	**Models Reuse:** Looks into if the methodology provides, or uses, a library of reusable models?
	SUPPORTIVE FEATURE BASED CRITERIAS
43	**Software and methodological support:** Checks if the methodology has support for tools and informative libraries (e.g. information library of agents, organizations, agent components, architectures and support)?
44	**Open systems and scalability**: Verifies if the methodology has feature for open systems interfacing and its scalability (e.g. if the methodology has provision for dynamic integration/removal of newer agents and resources)?
45	**Dynamic structure**: Verifies if the methodology provides support for a strong dynamic structure? (i.e. if the methodology has provision for dynamic reconfiguration of the agent system)?
46	**Agility and robustness**: Verifies if the methodology provides support for robustness and agility (e.g. if the methodology has provision to capture normal successful processing and exception fault processing; if it provides techniques to analyze performance of all the configurations, or/and provides techniques to detect failure and recover from such failure)?

47	**Support for conventional objects**: Verifies if the methodology caters for the use and integration of objects designed in system (e.g. if the methodology can model the agents' interfaces with object instances from a class)?
48	**Features for mobile agents**: Verifies if the methodology caters for the use and integration of mobile agents in system (e.g. if the methodology can model the attributes: which, when and how of a mobile agent
49	**Base for self-interested agents**: Verifies if the methodology has provision to support self-interested agents (that is agents whose goals may be independent or in clash with goals of other agents)
50	**Ontology support**: Verifies if the methodology caters for the use and integration of ontology concept in system (i.e. Ontology-based agent systems)?

5.2comparative analysis methodologies

As per my study of the methodologies and the references (Tran, Low et al.,2005) and (Henderson-Sellers and Giorgini, 2005) I have compiled the below results.

Table 6 Comparative analysis results (customized adoption from (Tran, Low et al., 2005) and (Henderson-Sellers, and Giorgini, 2005))

Evaluation criteria	**Prometheus**	**Tropos**	**O-MaSE**	GAIA	INGENIAS	MAS-CommonKADS	BDI-ASP
			Process based Criteria's				
Development Lifecycle	Iterative nature across all **phases**	Iterative nature across all **phases**	Iterative nature across all **phases**	Iterative nature across all **phases**	Iterative nature across all **phases**	Is Risk-driven & **component-based**	**Not Specific**
Coverage of the lifecycle	**Analysis & Design**	**Conceptual, Analysis Design & Code**	**Analysis & Design**	**Analysis & Design**	**Analysis & Design**	**Conceptual, Analysis& Design**	**Analysis and Design**
Development perspective	**Bottom Up**	**Top Down**	**Top Down**	**Top Down**	**Hybrid**	**Hybrid**	**Hybrid**
Application domain	**Any**	**Any**	**Any**	**Any**	**Any**	**Any**	**Any**
Size of MAS	**Not specified**	**Any size**	**< 10 agents**	**< 100 agents**	**Not Specific, Any Size**	**Not Specific**	**Not Specific**
Agent nature	**BDI Agents**	**BDI–like Agents**	**Heterogeneous**	**Heterogeneous**	**Agents with goals and states**	**Heterogeneous**	**BDI agents**
Support for verification	**Yes**	**Yes**	**Yes**	**No**	**Yes**	**Briefly mentioned**	**No**
Ease of understanding of process steps	**High**	**High**	**High**	**High**	**High**	**High**	**High**
Usability of the methodology	**High**	**Medium**	High, except for internal **agent modeling.**	Medium. Missing many important **steps**	**High**	**High**	Medium, Lack of detailed instructions **for each step**
Refinability	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**
Approach towards MAS Development	Object Oriented and Non-Role Oriented	**"I*modelling framework", and Non-Role Oriented**	Object Oriented, Non-Role Oriented, and **Goal Oriented**	Object Oriented, Role Oriented, and **Object Oriented**	Object Oriented, and **Role Oriented**	Knowledge Engineering, and Non-Role Oriented	Object Oriented and Role Oriented

Model based Criteria's							
Completeness	**High**	**High**	**High**	**High**	**High**	**High**	**Medium**
Formalization/ Preciseness	**High**	**High**	**High**	**High**	**High**	**Low**	**High**
Model derivation	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**No**	**Yes**
Ease of understanding	**High**	**High**	**High**	**High**	**Medium**	**Medium**	**High**
Consistency	**Yes**	**Yes**	Yes	Yes	**No**	No	No
Modularity	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**
Abstraction	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**
Autonomy	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**
Adaptability	**No**	**No**	**No**	**No**	**Possible**	**No**	**No**
Cooperative behavior	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**
Inferential capability	**Yes**	**Yes**	**Yes**	**No**	**Yes**	**Yes**	**Yes**
Communication ability	**Yes**	**Yes**	**Yes**	**No**	**Yes**	**Yes**	**Yes**
Personality	**No**	**No**	**No**	**No**	**No**	**No**	**No**
Reactivity	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**
Deliberative behavior	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**
Temporal continuity	**No**	**No**	**No**	**No**	**Yes**	**No**	**No**
Concurrency	**No**	**No**	**Yes**	**No**	**No**	**No**	**No**
Human Computer Interaction	**Yes**	**Yes**	**No**	**No**	**Yes**	**Yes**	**No**
Models Reuse	**Yes**	**Possible**	**Yes**	**Yes**	**Possible**	**Yes**	**Yes**
Supportive Feature based Criteria's							
Software and methodological Support	**Yes**	**No**	**Yes**	**No**	**Yes**	**No**	**No**

Open systems and scalability	**No**	**No**	**No**	**Yes**	**No**	**No**	**No**
Dynamic structure	**No**	**No**	**No**	**Yes**	**No**	**No**	**No**
Agility and robustness	**Yes**	**No**		**No**	**No**	**No**	**No**
Support for conventional Objects	**Yes**	**No**	**No**	**No**	**Yes**	**No**	**No**
Provisions for mobile Agents	**No**	**No**	**No**	**No**	**No**	**No**	**No**
Features for ontology	**No**	**No**	**No**	**No**	**No**	**Yes**	**No**

Table 7 Comparative analysis on support for steps in the development process (adoption from (Tran, Low et al., 2005) and (Henderson-Sellers, and Giorgini,2005))

Steps	**Prometheus**	**Tropos**	**O-MaSE**	GAIA	**INGENIAS**	**MAS-CommonKADS**	BDI-ASP
Problem Domain Analysis							
Identify system goals	0	3	3	0	3	0	0
Identify system roles	0	1	3	3	3	0	2A
Identify system functionality/tasks	3	3	3	3	3	2A	1
Develop use cases/scenarios	3	1	3	0	2A	2B	0
Produce sequence diagrams	0	2B	3	0	2B	2B	0
Identify design requirements	0	2A	0	0	2A	0	0
Identify agent classes	3	3	3	3	3	3	3
Agent's Inter-action design							
Agent inter-action paths	3	3	3	3	3	3	2A
Define exchanged messages	1	3	3	0	3	2B	0
Specify interaction protocols	3	3	3	0	3	3	0
Specify contracts/commitment	0	2A	0	0	2A	0	0

Any mechanism to resolve conflicts	0	0	0	0	0	0	0
Type of coordination or control rule (e.g. central or hierarchy)	0	1	1	0	1	0	0
Specify agent communication language	0	0	0	0	1	0	0
Agent Internal Design							
Define agent architecture	0	3	3	0	2B	1	0
Describes agent's mental attributes (e.g. beliefs, goals, , plans…)	3	3	0	0	2B	3	3
Describes agent's behavioral interface (e.g. services, capabilities,)	3	0	0	3	0	0	3
System/Environment Design							
Define system architecture/organizational structure	0	1	0	0	1	0	0
Describe dynamic group formulation of agents or dissolution of agents	0	0	0	0	0	0	0
Describe agent's relationships (e.g. aggregation and association, inheritance,)	0	0	0	3	0	2B	3
Specify co-existing non-agent entities	2A	0	0	3	0	0	0
Specify infrastructure/environment facilities	0	1	0	0	0	1	0
Specify agent-environment interaction mechanism	3	1	0	0	1	1	0
Instantiate agent classes	0	0	3	1	0	0	3
Specify agent instances location	0	0	3	0	0	0	1

Legend:

0: there is no support

1: denotes that the step is incorporated, however there is no techniques and / or examples

2A: denotes the methodology has techniques that it can perform the step

2B: denotes the methodology has examples to show how the step is performed

3: denotes the step is detailed with techniques and examples

5.3 My evaluation criteria's: Project based

Table 8 evaluation criteria's: Project based

Evaluation criteria	**Prometheus**	**Tropos**	**O-MaSE**	**GAIA**	**INGENIAS**	**MAS-CommonKADS**	**BDI-ASP**
Project based Criteria's							
Most suitable Project size	Small to Medium	Large	Medium	Small to Medium	Medium	Medium to Large	Medium to Large
Most suitable Project type	Academics and IT	Business and IT	Business and IT	Social and IT	Business and IT	Business and IT	Business and IT
Technology stack	Java / JACK based/ PDT tool	Java / JACK based	JADE	JADE	JADE	JADE	JADE
Project risk	Medium to Low	Medium to high	Low or medium	Low or medium	Low or medium	Medium to high	Medium to high
Open source tools	Yes, using Eclipse	Yes, using Eclipse	Only Java based tools and few .Net	Limited support with JADE	Yes, using Eclipse and Maven	Limited with C++ / Java libraries	Limited with Java libraries

5.3.1 Methodology selection based on project needs

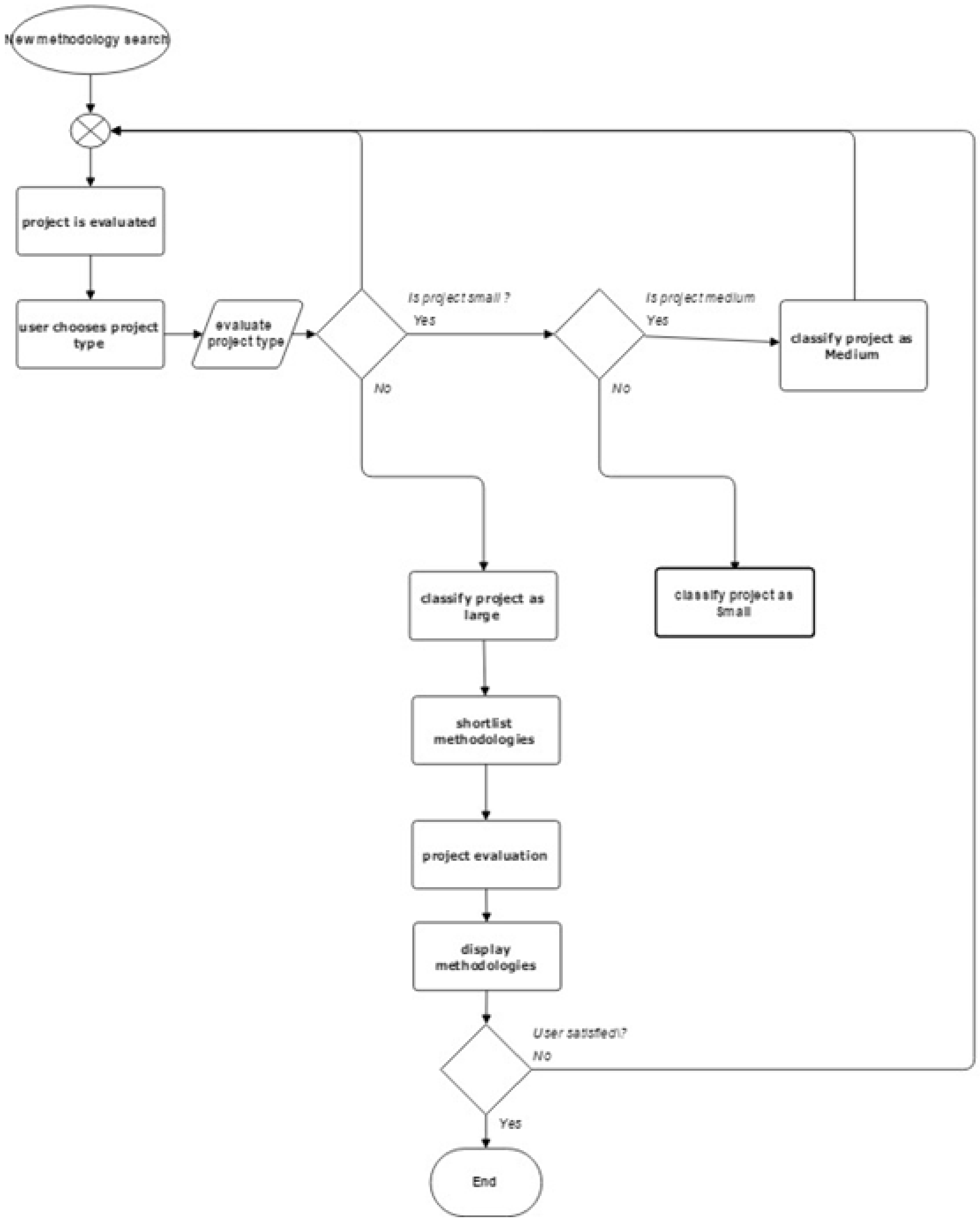


Figure 54 methodologies selection flow chart

6 RESEARCH RESULTS AND DISCUSSION

This chapter has presented an evaluation and comparison of the 7 AOSE methodologies described in Chapters 2-4 of this thesis. Overall, it is impossible (and not recommended) to determine which methodology is the best among these 7. The decision should depend on the target application. Each application entails a different set of requirements that indicate which evaluation criteria are the

most important and should be supported by the chosen AOSE methodology. When considering which methodology to adopt for a particular application, were commend ranking the required features that the methodology should support. Despite our efforts to provide a comprehensive and objective comparison, I
acknowledge the following limitations:

- My evaluation has been solely based upon the available documentation of the methodologies and their documented case studies, examples, and projects. We have not personally applied any of these methodologies on a real project. An optimal comparison would be to use the 7 methodologies on the same application(s).
- Some evaluation criteria are subjective in nature, particularly the usability and understandability of the methodologies' process steps, techniques, and models.

My research has been tabulated in Table: Comparative analysis results. I feel this is the best option to try and refer to when trying to start a new project using MAS. Each domain and each project is unique and I assume the decision to choose the best fit for the project is with the technical team and developers. I would recommend that the technical and decision makers meet at least once and discuss on the prospectus of each methodology and mapping the project needs to those provided by each methodology and related tool. Also, my research into these methodologies has brought forth few interesting facts that are shown in table below

Table 9 My views on the methodologies

Methodology	**Prometheus**
My views	During my research, I found that this methodology is good at Analysis and Design phases and needs external code generation tools like JACK and PDT (Java based)
Methodology	**Tropos**
My views	I find that Tropos is an All-in-One Methodology and has tools and good real examples. It covers all phases of SDLC. It uses several tools using Java, Eclipse plugins and UML

Methodology	**O-MaSE**
My views	This is object- oriented and focus is on Analysis and design. This does not capture Business related use cases and is more specific to agent Goals. It has AgentTool which is one tool that helps in limited code generation.
Methodology	**GAIA**
My views	GAIA serves the middle portion of the SDLC that is Analysis and Design. It does not capture Requirements. The models and entities are technology agnostic and can be easily adopted. Jadex is a suitable framework
Methodology	**INGENIAS**
My views	**This serves the Analysis, Design and Implement phases. It has models that are technology agnostic however if specific tools are used in Analysis then that locks to that tool. Java based tools like IDK, Agent Tool is popular.**
Methodology	**MAS-CommonKADS**
My views	Is good at Agent oriented software engineering. It supports the later on phases of SDLC like Design and Code generation. It is more focused on UML/OMT based models and code generation. However, it has limited support for coding and testing and relies on external tools. Popular tools are Plug-In for rational rose and newer rational tools.
Methodology	**BDI-ASP**
My views	This is an extended version of the traditional waterfall model customized to BDI agents. It has many steps and covers most of the SDLC life cycle except code, with key deliverables at each milestone

Finally, I wish to conclude that based on the project needs and the availability of the required tools and manpower the right methodology needs to be chosen. I have provided the best available methodologies and showcased how these are used in the SDLC of projects.

7 CONCLUSIONS AND FUTURE WORK

The methodologies proposed here have been successful with many multi-agent systems and related distributed systems, in areas of robotics and information systems. They have been used in many research projects and also in the Software Engineering course at various Universities. MAS is an ever-growing technology and I am sure there is ample scope for future work in this area.

In conclusion, I wish to pass on this thesis to the concerned person who will most likely benefit in his/her course and project. I am sure research scholar will lead the future of MAS and AI and we look forward for a great technology ahead.

8 REFERENCE

Wood, Mark. Multiagent systems engineering: a methodology for analysis and design of multiagent systems. Ohio: n.p., 2000. Print.

M. Wooldridge, "Agent-based software engineering," in IEE Proceedings - Software Engineering, vol. 144, no. 1, pp. 26-37, Feb 1997. doi: 10.1049/ip-sen:19971026

Alonso, Cristina Gómez, David Isern Alarcón, and Antonio Moreno. Software engineering methodologies to develop multi-agent systems: state -of-the-art. Tarragona: Universitat Rovira i Virgili, 2007. Print.

Sudeikat, Jan, Lars Braubach, Alexander Pokahr, and Winfried Lamersdorf. "Evaluation of Agent–Oriented Software Methodologies – Examination of the Gap Between Modeling and Platform." Agent-Oriented Software Engineering V Lecture Notes in Computer Science (2005): 1-16. Web.

Werneck, Vera Maria B., Rosa Maria E. Moreira Costa, and Luiz Marcio Cysneiros. "Modelling Multi-Agent System using Different Methodologies." Multi-Agent Systems - Modeling, Interactions, Simulations and Case Studies (2011): 78-79. Web.

Padgham, Lin, John Thangarajah, and Michael Winikoff. "AUML protocols and code generation in the Prometheus design tool." Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems - AAMAS '07 (2007): 1-2. Web.

Giunchiglia, Fausto, John Mylopoulos, and Anna Perini. "The Tropos Software Development Methodology: Processes, Models and Diagrams." Agent-Oriented Software Engineering III Lecture Notes in Computer Science (2003): 35-36. Web.

Garcia-Ojeda, Juan C., Scott A. Deloach, Robby, Walamitien H. Oyenan, and Jorge Valenzuela. "O-MaSE: A Customizable Approach to Developing Multiagent Development Processes." Lecture Notes in Computer Science Agent-Oriented Software Engineering VIII (2007): 5-7. Web.

Wooldridge, Michael, Nicholas Jennings, and David Kinny. "The Gaia Methodology for Agent-OrientedAnalysis and Design." Kluwer Academic Publishers, n.d. Web. 18 Apr. 2017.

Cernuzzi, Luca , Thomas Juan, Leon Sterling, and Franco Zambonelli. "THE GAIA METHODOLOGY: BASIC CONCEPTS AND EXTENSIONS." N.p., n.d. Web. 18 Apr. 2017.

Bauer, *bernhard, *jörg P. Müller, and James Odell. "Agent UML: A Formalism for Specifying Multiagent Software Systems." Agent-Oriented Software Engineering Lecture Notes in Computer Science (2001): 9-10. Web.

Gatti,, Maíra Athanázio de Cerqueira, Arndt Von Staa, and Carlos José Pereira De Lucena. "AUML-BP: A Basic Agent Oriented Software Development Process Model Using AUML." PUC-Rio Departamento de Informática, (2007). Web.

Pavón, Juan, Jorge Gómez-Sanz, and Rubén Fuentes. "Model Driven Development of Multi-Agent Systems." Model Driven Architecture – Foundations and Applications Lecture Notes in Computer Science (2006): 284-98. Web.

Iglesias, Carlos A., and Mercedes Garijo. "The Agent-Oriented Methodology MAS-CommonKADS." Agent-Oriented Methodologies (2005): 47-50. Web.

Iglesias, Carlos A., Mercedes Garijo, José C. González, and Juan R. Velasco. "Analysis and design of multiagent systems using MAS-CommonKADS." Intelligent Agents IV Agent Theories, Architectures, and Languages Lecture Notes in Computer Science (1998): 313-27. Web.

HYUN JO, CHANG, GUOBIN CHEN, and JAMES CHOI. "A FRAMEWORK FOR BDI AGENT-BASED SOFTWARE ENGINEERING." Studia Informatica Universalis, 2004. Web.

Hyun Jo, Chang, and Jeffery Einhorn. "A BDI Agent-Based Software Process." The JOT Blog. ETH Zurich, Nov. 2005. Web.

Morandini, Mirko, Duy Cu Nguyen, Anna Perini, Alberto Siena, and Angelo Susi. "Tool-Supported Development with Tropos: The Conference Management System Case Study." Lecture Notes in Computer Science Agent-Oriented Software Engineering VIII 4951 (2007): 182-96. Web.

Pi, Nayat Sanchez, Javier Carbó, and José Manuel Molina. "Analysis and Design of a Multi-Agent System Using Gaia Methodology in an Airport Case of Use." INTELIGENCIA ARTIFICIAL. Carlos III University of Madrid, (2010):9-17. Web.

Self, Athie, and Scott DeLoach. "Designing and specifying mobility within the multiagent systems engineering methodology." ACM Digital Library. ACM, 2003. Web.

Tran, Quynh-Nhu Numi, Graham Low, and Mary-Anne Williams. "A Preliminary Comparative Feature Analysis of Multi-Agent Systems Development Methodologies." Agent-Oriented Information Systems II Lecture Notes in Computer Science (2005): 157-68. Web.

Henderson-Sellers, Brian, and Paolo Giorgini. Agent-oriented methodologies. Hershey: Idea Group Publishing, 2005. Print.

Garcia-Ojeda, Juan, and Scott DeLoach. "O-MaSE: a customisable approach to designing and building ..." Kansas State University, n.d. Web. 22 Apr. 2017.

Kephart , Jeffrey, and David Chess. "The Vision of Autonomic Computing." IEEE Computer Society Press, Jan. 2003. Web. (pp. 41-50)

Lee, Raymond S. Fuzzy-neuro approach to agent applications: from the AI perspective to modern ontology. Berlin: Springer, 2006. Print.

Padgham, Lin, and Michael Winikoff. Developing intelligent agent systems: a practical guide. Chichester, West Sussex, England: John Wiley & Sons, 2005. Print.

Russell, Stuart J. (Stuart Jonathan), and Peter Norvig. Artificial intelligence a modern approach/ Stuart Russell ; Peter Norvig. New Jersey: Prentice Hall, 2003. Print.

Salah, Khalil, and Ardavan Ashabi. "A Review of Agent-Oriented Development Methodologies and Programming Languages/Frameworks." International Journal of Software Engineering (IJSE),5.3 (2014): 25-28. Web.

Kadera, Petr . "Methods for Development of Industrial Multi-Agent Systems." Czech Technical University in Prague, 2015. Web.

Kubera, Yoann, Philippe Mathieu, and Sébastien Picault. "Proceedings of the ninth International Joint Conference on Autonomous Agents and Multi-Agent Systems ." AAMAS 2010, 2010. Toronto, Canada: 1547-1548

Rao, A., and M. Georgeff. "BDI Agents: from theory to practice." V. Lesser. Proceedings of the First International Conference on Multiagent Systems (ICMAS'95). . The MIT Press. Cambridge, MA, USA, 1995..

Georgeff, M., and A. Lansky. "Reactive Reasoning and Planning." Proceedings of the Sixth National Conference on Artificial Intelligence (AAAI-87),. AAAI ORG, pg. 677-682, Seattle, WA, 1987.

Ferguson I. A., “Towards an architecture for adaptive, rational, mobile agents”. In Werner E. and Demazeau Y., editors, Decentralized AI 3 – Proceedings of the Third European Workshop on Modeling Autonomous Agents and Multi Agent Worlds (MAAMAW-91).

Wooldridge, Michael J. An introduction to multiagent systems. Chichester: John Wiley, 2009. Print.

Michael E. Bratman. Intention, Plans, and Practical Reason. CSLI Publications, http://csli-publications.stanford.edu/, 1st, edition, 1999.Web

Yang, Jung-Jin, Makoto Yokoo, and Takayuki Ito. "Principles of Practice in Multi-Agent Systems." 12th | Jung-Jin Yang | Springer. Springer-Verlag Berlin Heidelberg, 2009. Print.

Flores-Mendez, Roberto. "Towards the Standardization of Multi-Agent Systems Architectures." ACM Crossroads' special issue on Intelligent Agents (2009): 1-10. Web.

Glavic, Dr. Mevludin. "Agents and Multi-Agent Systems: A Short Introduction for Power Engineers." University of Liege (2006): 1-21. Web.

Maalal, Sara, and Malika Addou. "A new approach of designing Multi-Agent Systems." International Journal of Advanced Computer Science and Applications 2.11 (2011): 149-50. Web

Odell, James, Paolo Giorgini, and Jörg P. Müller. Agent-Oriented Software Engineering V 5th International Workshop, AOSE 2004, New York, NY, USA, July 19, 2004. Revised Selected Papers. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005. Print.

BRESCIANI, PAOLO, ANNA PERINI, PAOLO GIORGINI, FAUSTO GIUNCHIGLIA, and JOHN MYLOPOULOS. "Tropos: An Agent-Oriented Software Development Methodology." Autonomous Agents and Multi-Agent Systems, (2004): 203-236. Web

Mohire, Vijayananda. "DERIVING PROJECT-VALUES OF MULTI-AGENT SYSTEM." SlideShare. Karnataka State Open University, 30 Apr. 2012. Web.

J. Odell, H. Van Dyke Parunak, and B. Bauer, “Extending UML for Agents,” Proc. Agent-Oriented Information Systems Workshop, 17th Nat’l Conf. Artificial Intelligence, G.Wagner, Y. Lesperance, and E. Yu, eds., ICue Publishing,2000

G. Booch, J. Rumbaugh, and I. Jacobson, The Unified Modeling Language User Guide, Addison-Wesley, 1999, Print

Huget, M.-P. "Agent UML notation for multiagent system design." IEEE Internet Computing 8.4 (2004): 63-71. Web.

Pavón, Juan; Gómez-Sanz, Jorge (2003). "Agent oriented software engineering with INGENIAS". Multi-Agent Systems and Applications III. Lecture Notes in Computer Science. Springer Berlin Heidelberg. 2691: 394–403

Pavón, Gómez-Sanz & Fuentes, (2005), "The INGENIAS Methodology and Tools", In: Agent-Oriented Methodologies. Idea Group Publishing, USA (2005)

Juan and Gómez-Sanz. "Agent Oriented Software Engineering with INGENIAS." Lecture Notes in Computer Science Multi-Agent Systems and Applications III (n.d.): 394-403. Web. 2017c

Morandini,Nguyen,Perini,Siena, "Tool-supported Development with Tropos:The Conference Management System Case Study", Fondazione Bruno Kessler - IRST,38050 Trento, Italy, Web,2017

Nayat Sánchez-Pi, Javier Carbó and José Manuel Molina, "Analysis and Design of a Multi-Agent System Using Gaia Methodology in an Airport Case of Use", Inteligencia Artificial 45(2010), 9-17, Madrid, Spain, 2010

Self, Athie, and Scott DeLoach. "Designing and specifying mobility within the multiagent systems engineering methodology." ACM Digital Library. ACM, 2003. Web.

Tran, Quynh-Nhu Numi, Graham Low, and Mary-Anne Williams. "A Preliminary Comparative Feature Analysis of Multi-agent Systems Development Methodologies." Agent-Oriented Information Systems II Lecture Notes in Computer Science (2005): 157-68. Web.

Brian Henderson-Sellers, Paolo Giorgini, "Agent-oriented Methodologies" , Chapter XII, Idea Group Publishing, 2005