Université d'Ottawa · University of Ottawa

Université d'Ottawa · University of Ottawa

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTDOCTORALES

FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES

Naoual EL KARIMI

AUTEUR DE LA THÈSE - AUTHOR OF THESIS

M. A. Sc. (Electrical Engineering)

GRADE - DEGREE

Department of Electrical Engineering

FACULTÉ, ÉCOLE, DÉPARTEMENT - FACULTY, SCHOOL, DEPARTMENT

TITRE DE LA THÈSE - TITLE OF THE THESIS

Design and Implementation of a Personal Assistant Using Agent Technology

A. Karmouch

DIRECTEUR DE LA THÈSE - THESIS SUPERVISOR

CO-DIRECTEUR DE LA THÈSE - THESIS CO-SUPERVISOR

EXAMINATEURS DE LA THÈSE - THESIS EXAMINERS

T. Kunz

L. Oruzcu-Barbosa

J.-M. De Koninck, Ph.D

LE DOYEN DE LA FACULTÉ DES ETUDES SUPÉRIEURES ET POSTDOCTORALES

DEAN OF THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES

Design and Implementation of a Personal Assistant Using Agent Technology

Naoual EL KARIMI, B. Eng.

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
in partial fulfillment of the requirements for the degree of

Master of Applied Sciences

in Electrical Engineering

Ottawa-Carleton Institute of Electrical and Computer Engineering
School of Information Technology
Faculty of Engineering
University of Ottawa

Your file Votre référence
ISBN: 0-494-01467-9
Our file Notre référence
ISBN: 0-494-01467-9

Dedications

Too the belated memory of my dearly loved grandmother

"Gone but never forgotten"

To the loving and the most cherished father and mother

Abdessalam El Karimi and Salama El Mourabet

"Your love, affection and sacrifice, I shall keep forever in my heart"

To my husband, my soul-mate, my inspiration, and my friend

Reda Tkito

"You are my calm voice in the middle of all the storms"

To my dear brothers who got me wherever I am today

El Mondir and Adel

"Without either of you I would be nothing"

To my lovely sister and her husband

Rachida and Uncle Abdellah

"You have taught me that we can be loved and supported unconditionally"

To my dearest sweet sisters

Bouchra, Saloua and Manal

"You are better sisters than words can express"

To My Father and Mother in law

Abderrahmane Tkito & Leila Benali

To My brothers in Law

Taha and Yassine

"You are a special gift of love, trust, and understanding"

To My Uncles and Antes

To the families El Karimi, El Morabet, Tkito and Benali...

Thank you All

ABSTRACT

Intelligent agents are rapidly gaining popularity in our increasingly networked world since their emergence at the beginning of the last decade as a promising design and implementation solution for future applications. They provide information monitoring, searching filtering, and decision aiding that help people focus on the information that is most pertinent to them in a particular context.

One central class of agents is personal assistants. They can be distinguished from ordinary agents by the fact that they work at the interface level rather than the application level. They model their owners' interests, provide services to them, and act on their behalf when required. Their domains of application include Internet information retrieval, E-commerce, telecommunication, and network management.

This thesis aims to present a general model of personal assistants using agent technology. The proposed model aims to efficiently provide all necessary functionality regardless of the application. Our system architecture consists of three distinct agents. They are the User Interface Agent, Information Management Agent, and Personal Agent. They communicate with each other, with the human user, and with other agents using Agent Communication Language in order to provide services to their owners. The model was applied in the Virtual Team Collaboration project as an agent that helps the user managing his Agenda, managing the team, scheduling meetings, and acting on the user's behalf. The thesis work show also how can we use our model of personal assistant in E-commerce.

ACKNOWLEDGEMENTS

I would like to express my profound gratitude to my thesis supervisor, Pr. Ahmed Karmouch, who guides me all along the steps of my project, and who didn't spare any effort to advise me in order to achieve this thesis.

I would like also to thank all members from the Multimedia & Mobile Agent Research Laboratory for their strong motivation, the friendly work environment, and their interesting discussions and brainstorms.

Special thanks are due to Hamid Harroud, Mouhsine Lakhdissi, and Reda Tkito for their valuable assistance throughout the realization of this work and the writing of this thesis, for their continual encouragement, and for their precious friendship.

I would not forget the administrative staff at the University of Ottawa, who deserves my deep recognition for his wonderful help.

Finally, I want to extend my sincere appreciation to every person who had, in one way or the other, contributed to the accomplishment of this modest work.

Thank you all.

TABLE OF CONTENTS

LIST OF FIGURES IV

LIST OF ABREVIATIONS V

LIST OF ABREVIATIONS V

CHAPTER 1: INTRODUCTION 1

1.1 MOTIVATION 1
1.2 OBJECTIVE 2
1.3 MAIN CONTRIBUTION 3
1.4 THESIS OUTLINE 3

CHAPTER 2: BACKGROUND & RELATED WORK 5

2.1 AGENT PARADIGM 5
2.1.1 Common Agent Definition 5
2.1.2 Characteristics And Classification 6
2.1.3 Mobile Agents 8
2.2 MULTI-AGENT SYSTEMS 11
2.2.1 Overview 11
2.2.2 Agent Communication 12
2.2.3 A Survey of Agents Frameworks 12
2.3 PERSONAL ASSISTANT AGENTS 14
2.3.1 FIPA Specification For Personal Assistant 14
2.3.2 Personal Assistant Building Approaches 16
2.3.2.1 Knowledge-based approach 16
2.3.2.2 Machine learning approach 16
2.3.2.3 Collaborative agent approach 17
2.3.3 Domains Of Application 18
2.3.3.1 Network assistant 18
2.3.3.2 Travel assistant 19
2.3.3.3 Internet assistant 21
2.3.3.4 E-commerce assistant 22
2.3.3.5 E-mail assistant 22
2.3.3.6 Meeting-scheduling assistant 23
2.3.3.7 News Filtering assistant 24
2.4 CHALLENGING ISSUES 24
2.5 PROPOSED MODEL 26
2.6 SUMMARY 27

CHAPTER 3: DESIGN AND CONCEPT 28

3.1 INTRODUCTION 28
3.2 OBJECTIVE 29
3.3 MODEL AND ARCHITECTURE 29
3.3.1 Design Choices 29
3.3.2 General Architecture 32
3.3.3 User Interface Agent 33
3.3.3.1 Definition and role 33
3.3.3.2 Different components 35
3.3.4 Information Management Agent 36
3.3.4.1 Definition and role 36
3.3.4.2 Different components 36
3.3.5 Personal Agent 38
3.3.5.1 Definition and role 38
3.3.5.2 Different components 39
3.4 SYSTEM INTERACTIONS 39

3.4.1 Human-Agent Interaction 39
3.4.2 Agent-Agent Interaction 40
3.4.3 Communication Protocol 40
3.5 PERSONAL ASSISTANT IN V-TEAM COLLABORATION PROJECT 43
3.5.1 Overview of V-team Project 44
3.5.2 Services of The Personal Assistant 45
3.5.2.1 Internal services 46
3.5.2.2 External services 47
3.5.3 Application scenario 49
3.6 EXTERNAL SOFTWARE INTEGRATION 54
3.7 ADVANTAGES AND LIMITATIONS 55
3.8 SUMMARY 56

CHAPTER 4: IMPLEMENTATION 58

4.1 INTRODUCTION 58
4.2 IMPLEMENTATION CHOICES 58
4.2.1 FIPA-OS Platform 58
4.2.2 Programming Language 61
4.2.3 Why XML For Data Storage 61
4.3 SYSTEM AGENTS 63
4.3.1 Agent Objcet 63
4.3.2 User Interface Agent 64
4.3.3 Information Management Agent 66
4.3.4 Personal Agent 68
4.3.5 Agents Communication 69
4.4 PROTOTYPE 72
4.4.1 Overview 72
4.4.2 User Interface Agent 72
4.4.3 Information Management Agent 73
4.4.4 Personal Assistant 74
4.5 SUMMARY 75

CHAPTER 5: PERSONAL ASSISTANT IN E-COMMERCE 76

5.1 INTRODUCTION 76
5.2 RELATED WORK 77
5.2.1 Kasbah 77
5.2.2 Magnet 77
5.3 STATE OF THE ART 78
5.4 THE OVERALL SCENARIO 80
5.5 ARCHITECTURE 83
5.5.1 Actors And Roles 84
5.5.1.1 Personal Agent 84
5.5.1.2 User Interface Agent 85
5.5.1.3 Information Management Agent 86
5.5.1.4 Surrogate Mobile Agents 88
5.5.2 Examples Of Services 89
5.5.2.1 External services 89
5.5.2.2 Internal services 90
5.5.3 Negotiation Protocol 90
5.5.3.1 Between mobile agent and other buying/selling agents 90
5.5.3.2 Between personal assistant and other buying/selling agents 91

CHAPTER 6: CONCLUSION 92

6.1 SUMMARY 92
6.2 SUGGESTION FOR FUTURE WORK 93

REFERENCES 95

LIST OF FIGURES

FIGURE 2.1: A CLASSIFICATION OF AGENTS 8
FIGURE 2.2: A VIEW OF AN AGENT TYPOLOGY 8
FIGURE 3.1 THE STRUCTURE OF THE PERSONAL ASSISTANT 31
FIGURE 3.2: GENERAL ARCHITECTURE OF THE PERSONAL ASSISTANT 33
FIGURE 3.3: DIFFERENT SCENARIOS 40
FIGURE 3.4: THE PROTOCOL OF USING EXTERNAL SERVICES 41
FIGURE 3.5: THE PROTOCOL OF REPLYING TO EXTERNAL REQUESTS 42
FIGURE 3.6: THE PROTOCOL FOLLOWED IN THE CASE OF NEW EVENTS 43
FIGURE 3.7 SYSTEM INTERACTION WITH DIFFERENT V-TEAM COLLABORATION PROJECT 44
FIGURE 3.8: VIEW OF PERSONAL AGENT SERVICES 45
FIGURE 3.9 THE PROCESS OF TEAM CREATION, AGENDA MANAGEMENT, AND MEETING SCHEDULING 50
FIGURE 3.10: THE PROTOCOL OF MEETING NEGOTIATION 52
FIGURE 3.11:THE PROCESS OF SESSION INITIALISATION 53
FIGURE 4.1: FIPA-OS CORE COMPONENTS 59
FIGURE 4.2: UML VIEW OF AGENT CLASS DIAGRAM 64
FIGURE 4.3: UML VIEW OF USER INTERFACE AGENT CLASS DIAGRAM 66
FIGURE 4.4: UML VIEW OF INFORMATION MANAGEMENT AGENT CLASS DIAGRAM 68
FIGURE 4.5: UML VIEW OF PERSONAL AGENT CLASS DIAGRAM 69
FIGURE 4.6: OVERVIEW OF ACL MESSAGE STRUCTURE 70
FIGURE 4.7: GUI SCREEN FOR AGENDA MODIFICATION 73
FIGURE 4.8: GUI SCREEN FOR ATTENDANCE RULES SPECIFICATION 73
FIGURE 4.9: EXAMPLE OF XML FILES STORED BY THE IMA 74
FIGURE 4.10: EXAMPLE OF REQUESTS HANDLED BY THE PERSONAL AGENT 74
FIGURE 5.1: SCENARIO OF USING MOBILE AGENTS 81
FIGURE 5.2: SCENARIO OF CONTACTING EXTERNAL BUYER/SELLER AGENTS 82
FIGURE 5.3: VIEW OF THE PA INTERACTIONS 84

LIST OF ABREVIATIONS

ACL	Agent Communication Language
AI	Artificial Intelligence
AMS	Agent Management System
AOA	Agent-Oriented Analysis
AOP	Agent-Oriented Programming
API	Application Program Interface
ARA	Agent for Remote Action
ARPA	Advanced Research Projects Agency
CM	Conversation Manager
CORBA	Common Object Request Broker Architecture
DAI	Distributed Artificial Intelligence
DCOM	Distributed Component Object Model
FIPA	Foundation of Intelligent Physical Agents
FIPA-OS	FIPA Open Source
GUI	Graphical User Interface
IIOP	Internet Inter-ORB Protocol
IMA	Information Management Agent
IMT	Iconic Modelling Tool
JDK	Java Development Toolkit
JVM	Java Virtual Machine
KIF	Knowledge Interchange Format
KQML	Knowledge Query and Manipulation Language
MA	Mobile Agent
MAgNET	Mobile Agent for Electronic Trading
MAS	Multi-Agent Systems
MCL	Macintosh Common Lisp

MTS	Message Transport Protocol
Narval	Network Assistant Reasoning with a Validating Agent Language
OMG	Object Management Group
OOP	Object-Oriented Programming
ORB	Object Request Broker
PA	Personal Agent
PIA	Personal Internet Assistant
RMI	Remote Method Invocation
SET	Secure Electronic Transaction
SIP	Session Initiation Protocol
SSL	Secure Socket Layer
TCL	Tool Command Language
TM	Task Manager
UAML	Unified Agent Modelling Language
UIA	User Interface Agent
UML	Unified Modelling Language
URL	Uniform Resource Locator
WWW	World Wide Web
XML	eXtendible Mark-up Language

CHAPTER 1: INTRODUCTION

1.1 Motivation

Intelligent agents are rapidly gaining popularity in the increasingly networked computing world. They emerged at the beginning of the last decade as a promising design and implementation model for future applications. Their domains of application include E-commerce, telecommunication, network management, and information retrieval.

The definition of agents stands from the real human world where agents receive instructions from clients and perform their duties according to their clients' specifications. In the software world, many definitions have been given to agents depending on their degree of autonomy, intelligence, and reactivity. One simple definition was given by Huhns and Singh [1] in 1998: "*Agents are active, persistent (software) components that perceive, reason, act, and communicate*".

The agent concept raises many problems such as the communication mechanism, agents' management, and agent behavior specification. During the last decade, intense research has been directed at both the design and the implementation level. Communication mechanisms have been defined and many platforms have been developed. Amongst the most known Java-based platforms are D'Agent from Dartmouth College, Aglets from IBM, Voyageur from ObjectSpace, and Concordia from Mitsubishi.

A central class of agents are personal assistants. They are also known in the software world as interface agents, intelligent agents, or personal agents. They can be distinguished from

ordinary agents by the fact that they are attached to the user. Every user has his own personal assistant, as he has his own agenda or address book. Personal Assistants identify their owners, reflect their likes and dislikes, manage their personal information, and act on their behalf when required.

However, existing personal assistants present three major problems:

The first one is that most of the existing personal assistants consist of a single agent conceived to provide a specific service. There is no general model of personal assistants that can be applied for all kind of services. Therefore, the user is obliged to learn how to use a different personal assistant for every service.

The second problem is that few of the existing personal assistants provide an enhanced user interface that helps the user interact with his personal assistant. Indeed, the nature of the personal assistant tasks requires much more interactions with the human user than other types of agents. Therefore, enhanced graphical user interfaces, especially those based on visual languages, are required.

The third problem is that few of the existing personal assistants have the ability to interact with other agents and application. This quality is very important to enhance the quality of the service offered to the human user.

1.2 Objective

The main objective of this work is to design and develop a general model of personal assistants using agent technology. An enhanced user interface constitutes an important component of our model of personal assistants. The personal assistant is able to interact with other agents and applications and take some decisions without any intervention from the human user.

The personal assistant was implemented in Java using FIPA-OS platform and integrated as a part of the V-team collaboration project. The project was conducted in collaboration with Nortel Networks Corporation and gave us the opportunity to test our model in the context of a real application.

Another objective of this thesis is to show how our model of personal assistant can be applied in the area of E-commerce. We will specify the new tasks of the personal assistant and give the overall scenario of the application.

1.3 Main Contribution

The main contribution of this thesis work is the design and implementation of a general model of personal assistants using agent technology. The personal assistant consists of a multi-agents system where each agent accomplishes specific tasks.

The conception work includes also the communication protocol between different system's agents and between external agents and the personal assistant.

Another contribution is the implementation of all the system components and the integration of the Iconic Modelling Tool as an important part of the User Interface Agent.

The last contribution is the application of our model of personal assistant in the domain of E-commerce. We will redefine the task of each agent of the system, and propose a negotiation mechanism between the personal assistant and other E-commerce agents.

1.4 Thesis Outline

The rest of the thesis is organized as follows. Chapter 2 introduces software agents, discuss some agents' related issues, and gives an overview of existing personal assistants. Chapter 2

presents also the drawbacks of existing personal assistants and the model that we propose to address them.

The Chapter 3 is dedicated to present our personal assistant design and concepts. It explains the general architecture of the system, describes in detail its components, and presents the application of our model in the V-team collaboration project. Chapter 4 describes the implementation of our personal assistant, and includes some of the system prototypes.

Chapter 5 describes another application of our personal assistant in electronic commerce. It explains how the model can be applied, gives a scenario of the application, and proposes a negotiation mechanism between the personal assistant and other E-commerce agents. Chapter 6 summarizes the thesis and provides some suggestions for future research work.

CHAPTER 2: BACKGROUND & RELATED WORK

2.1 Agent Paradigm

2.1.1 Common Agent Definition

Since the emergence of the agent concept in the last decade, researchers have offered a variety of definitions according to their own use of the word "agent". However, they were unable to agree on a consensus definition on it.

Some of these definitions are discussed in the following:

Pattie Maes [2], from the MIT's Media Laboratory, exhibits three crucial properties to the definition of an agent. They are situated in a complex environment, autonomous, and goal-oriented.

> "*Autonomous agents are computational systems that inhabit some complex dynamic environment, sense and act autonomously in this environment, and by doing so realize a set of goals or tasks for which they are designed*"[2].

IBM's [3] definition adds other properties to agents. They are intelligent and have some knowledge that represents the user's goal.

> "*Intelligent agents are software entities that carry out some set of operations on behalf of a user or another program with some degree of independence or autonomy, and in doing so, employ some knowledge or representation of the user's goals or desires*" [3].

Wooldridge and Jennings [4] give a definition of agent close to the artificial intelligence (AI). Indeed, they combine agency with artificial intelligence.

> *"It is interesting to note that one way of defining AI is by saying that it is the sub field of computer science which aims to construct agents that exhibit aspects of intelligent behaviour."* [4].

The real problem in producing a universal definition to agency is the diversity of the area in which agents are being applied. Examples of such fields include telecommunication and smart networks, in human computer interaction as intelligent user interfaces and believable computer personas, and in control of large-world situations such as power systems management and air traffic control [5].

Therefore, since agents are involved in such many fields of interest, the question surrounding agents is not what an agent is but how can we classify agents and what are the features that they should have in common. In the following section we give some of the most known agents' characteristics and the most important classifications that we found in the literature.

2.1.2 Characteristics And Classification

The various definitions of agents allow us to extract the following properties [5]:

- Autonomy: Reflects the ability of the agent to work separately from the user according to its own knowledge and its internal state given the boundaries and the limitation of its tasks.
- Reactivity: Describes the ability of an agent to perceive its environment and to respond to changes in a timely fashion.

- Pro-activity: Agents should also be able to do a task pre-emptively and to take some initiatives that will benefit the end user without direct intervention from him.

- Sociability: The user and the agent should co-operate with each other to reach a specific goal. The communication should be two ways rather than a user sending orders to the agent. This co-operation allows the user to verify what is required when the agent gives feedback on what it believes the user wants.

Agents that satisfy all the four previous characteristics are said to adhere to a weak agency. An agent adheres to a strong agency if it responds to some properties that normally are attributed to human such as mental notions of knowledge, beliefs, intention and obligation [6].

Other agents' characteristics include [5]:

- Veracity: An agent will not knowingly communicate false information.

- Benevolence: An agent cannot have conflicting goals that either force it to transmit false information or to effect actions that cause its goals to be unfulfilled or embedded.

- Learning: It refers to the ability of the agent to learn while it is interacting with its environment.

- Personalization: An agent is personalised to a particular user. A personal agent should be able to observe his owner behaviour and conduct some conclusion that will allow him to provide some services to his owner later.

- Mobility: An agent should be able to migrate from one machine to another to perform a specific task.

Agents may be classified according to the subset of these properties that they fulfil. Examples include Mobile Agents, Personal Agents, and Co-operative Agents. (Figure 2.1).

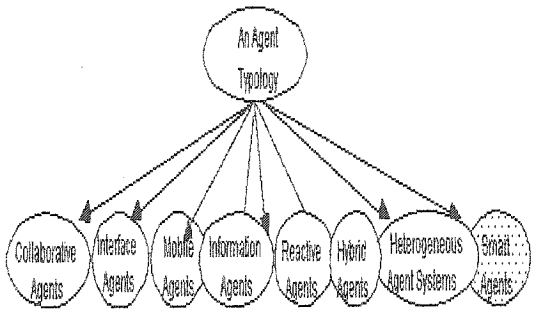

FIGURE 2.1: A CLASSIFICATION OF AGENTS [7]

Another classification can be done according to their tasks. Examples include information-gathering agents, e-mail filtering agents, and buying /selling agents.

To summarise agents' characteristics we present Bradshaw's figure that classifies agents according to their abilities to learn, co-operate and act autonomously. Indeed, it uses these three minimal characteristics to derive four types of agents: collaborative agents, collaborative learning agents, interface agents, and truly smart agents. (Figure 2.2).

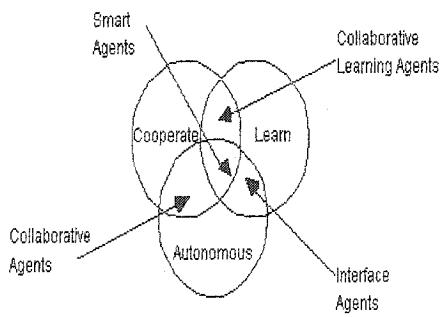

FIGURE 2.2: A VIEW OF AN AGENT TYPOLOGY [7]

2.1.3 Mobile Agents

Mobile Agents (MA) can be defined as entities that can cross the network to fulfil specific tasks. Mobile agent technology emerged when researchers from the Distributed Artificial

Intelligence (DAI) start giving them some importance. The idea of entities crossing the network is not new and emerged since the advent of distributed computing. The main difference is that it was the distributed operating system, which desires entities to move and makes them move across the network whereas a mobile agent has the basic ability to transport himself to new locations even though this property is not sufficient to describe a mobile agent [5].

This approach has many advantages [12]:

- *Efficiency*: MA can help reduce the traffic in the network by moving the computation to the data rather than the data to the computation. The MA crosses the network to the location where resources are available, and then it makes the intermediate computation and returns the result.

- *Persistence*: MA is not affected by the failure that can occur to certain nodes of the system. It has the ability to survive and reach many resources once it is launched. This is very useful to the end user since he can log on, launch the agent and log off, the mobile agent will then continue performing its task. The user can then log on and get the results from his mobile agent.

- *Peer-to-peer communication*: While the Client/Server paradigm offers one-way communication where a client sends requests to the server and the server replies to it, mobile agents are equivalents and offer two-way communication. Mobile agents can play the role of a client that is looking for information and at the same time reply to other agents' requests.

- *Fault tolerance:* In Client/Server paradigm, a connection between the two sides is mandatory to perform a specific task. If the network shuts down it is always difficult to recover the lost information and to re-synchronise the client with the server. With

mobile agent paradigm the problem is not present since it is not necessary to maintain a permanent connection.

Wayner summarises the drawbacks of mobile agents that are still discouraging the use of them widely in the software world in four points [13]:

- *Authentication:* Deals with the issue of how can systems know that a given agent is what it claims to be.
- *Security:* Deals with two issues. The first one is how can we avoid host nodes from being accessed by destructive or unauthorised agents. The second issue is how to protect agents from destructive hosts.
- *Secrecy:* Treats the question of how can we protect the internal state and the private data of the agent from being copied or intercepted by other entities while it is crossing the network.
- *Payment for services:* Treats the issue of how can agents pay for the service that it requests and how can we make sure that the payment was made correctly and that the service was satisfactory.

Mobile Agents have a number of domains of applications. Some often-cited applications include Secure Brokering, Parallel Processing, Information Dissemination, Workflow Applications and GroupWare, Network Management Services, Electronic Commerce, and Information Retrieval. Those different applications have in common the fact that they are greatly concerned by the network bandwidth use. Some of the platform that support agent mobility are: D'agents (Agent Tcl) , ARA, Aglets, Concordia, Odyssey (Telescript) , Voyager, Grasshopper, and IKV++.

2.2 Multi-Agent Systems

2.2.1 Overview

An agent by definition is not an isolated entity. It is able to communicate and co-ordinate with other entities. This means that agents that are not able to work together are destined to become useless. The nature of agents as collaborative entities has motivated many research groups to work on the standardisation of multi-agent system (MAS). Indeed, various definitions from different disciplines have been proposed for the term multi-agent system.

The motivation of the increasing interest in MAS research can be summarised as follows:

Modularity and abstraction: In MAS, complex, a large and unpredictable problem can be composed to modular component (agents) specialized in solving a particular problem aspect. Indeed, a single agent capacity is always limited by its knowledge, its computing resources, and its perspective [10]. Different system agents should then co-ordinate with each other to ensure the resolution of the global problem.

Openness and distribution: The characteristics of an open system is that its components are not known in advance, can change over time, and can consist of highly heterogeneous agents implemented by different people, at different times, and with different software tools and techniques [10]. Such systems require that agents co-operate with each other in peer-to-peer communication to increase the capacity of a single agent to resolve its problem.

Learning: In MAS, agents can consult other agents in the system that are specialised in the same problem and have more experience and knowledge than it. This allows increasing the knowledge of the agent and relieving the programmer from giving all the required knowledge to the agent. The programmer can give to the agent just a minimum knowledge and the agent can then learn step by step by interacting with other agents in the system.

2.2.2 Agent Communication

One of the most important aspects of multi-agents systems is communication. Indeed, agents are always conducted to co-operate with each other to fulfil their tasks. However, communication between agents is not easy. Complex common protocols should be defined to reach those requirements. Indeed, researchers in agents' communication define three standards to achieve multi-agent interaction: A common agent communication language and protocol (i.e: KQML) , a common format for the content of communication (i.e: KIF), and a shared ontology.

The Knowledge Query and Manipulation Language (KQML) [11], developed by the ARPA (Advanced Research Projects Agency) Knowledge Sharing Effort, is a language and set of protocols that allow agents to interact with each other. It was conceived as both a message format and a message-handling protocol to support run-time knowledge sharing among agents. The Knowledge Interchange Format (KIF) [11] is often used in conjunction with KQML to express the content of a knowledge base. There are two intentions behind developing such language. The first one is that it can be used to support translation from one content language to another. The second intention is to create a common content language between two agents, which use different native representation language.

Ontology [11] can be considered as the vocabulary with which queries and assertions are exchanged. Each ontology defines a set of classes, functions, and object constants for some domains. By translating commands written in KIF to various other forms, sentences can be interpreted without any ambiguity and independent from the context.

2.2.3 A Survey of Agents Frameworks

With the growing interest that agents are receiving as a new potential concept, a number of agent platforms have been designed and implemented to provide frameworks within which

agent systems could be developed. In our model we choose FIPA-OS (FIPA Open Source) as agent's platform for agent creation, management, communication and mobility. The following section presents a review of the most prevalent agent platforms:

- **D'Agent [8]:** better known as Agent TCL, is an agent system from Darmouth College. It can be viewed as programs that can be written in different programming languages. The first version of D'Agent was written in TCL (Tool Command Language) but was extended later to support Java and Scheme language

- **ARA [8]:** The Agent for Remote Action (ARA) is an agent platform from the University of Kaiserslautern. It is similar to the D'Agent concept in that it supports many languages trough interpreters. The complete ARA system (core and agents) runs as a single application process on top of an unmodified host operating system.

- **Aglet [8]**: is a mobile agent system from IBM. It follows the same principle as Java applets. All Aglet frameworks are entirely written in Java to ensure maximum portability. The communication is done via a whiteboard mechanism and a message-passing scheme that supports coupled asynchronous as well as synchronous peer-to-peer communication.

- **Concordia [8]:** Another agent platform is Concordia form Mitsubishi Electric. It consists of three components all written in Java: Java Virtual Machine (JVM), server, and at least one agent. In general there are many server components that constitute the agent execution environment. They run on a single or several JVM.

- **Odyssey [8]:** Odyssey is a Java reincarnation of Telescript. Odyssey inherits most of its features from it but uses a standard agent communication language instead of the Telescript agent object-oriented language. Odyssey is based on four concepts: Agent and Worker, Places, Ticket, and Petition.

- **Voyager [8]:** is a Java-based platform from ObjectSpace that presents the particularity to be the most tightly integrated to the world of Java object-oriented programming. It is considered as an enhanced Object Request Broker (ORB). The principal component of Voyager is the Voyager ORB that allows an enhanced communication mechanism.

- **Grasshopper [27]:** It is based on three principles: Agencies, Places, and Agents. Each agency is the actual runtime environment for mobile and stationary agents, consist of two parts, i.e the core agency and one ore more places. Grasshopper is developped by the IKV++ Technologies AG and is supported by Java 2.

2.3 Personal Assistant Agents

P. Maes defines an Interface Agent as "*computer programs that employ Artificial Intelligence techniques to provide active assistance to a user with computer-based tasks*"[15]. This kind of agents is a metaphor to personal assistants. They assist a user in different ways and in multiple areas: they perform tasks on the user behalf, train or teach the user, help different users collaborate, and monitor events and procedures. Some examples of applications where a personal assistant can be useful include News filtering, information retrieval, mail management, meeting scheduling, and Internet assistance.

2.3.1 FIPA Specification For Personal Assistant

The Foundation of Intelligent Physical Agents FIPA [16] provided its first output in 1997 known as FIPA 97. It provides a specification of basic agent technologies that can be integrated by agent systems developers to build complex systems with a high degree of interoperability. It produces two kinds of specification:

- Normative specifications that mandate the external behaviour of an agent and ensure interoperability with other FIPA specified subsystems,

- Informative specifications of applications for industry on the use of FIPA technologies.

FIPA defines a personal assistant (PA) as "*software agent that acts semi-autonomously for and on behalf of a user, modelling the interest of the user and providing services to the user of other people/PAs as and when required*" [16].

The FIPA 97 specification focuses on the generic requirement for the personal assistant applications, other FIPA application scenarios –especially personal travel assistance and audio-visual entertainment and broadcasting include the notion of personal assistance for specific applications. The Personal Assistant includes several internal and external functions and services in order to extend the personal assistant basic functionality. They include managing the user diaries, filtering and sorting e-mails, managing a user desktop environment, managing the user activities, locating and delivering (multimedia) information, recommending entertainment, purchasing desired items, and planning travels.

One of the basic functions of a PA is the management of the user directory, meeting schedules services, and information management services (E-mail and news filtering, sorting and prioritising all sort of received information)

One scenario in the FIPA specification for personal assistants is that of arranging meetings among several participants located across companies and using different calendar management systems. This scenario lends itself well to agent technology, due to the need for user profiling, integration of heterogeneous software, action on a user behalf, local control in particular of the user calendar [16].

The FIPA specification introduces also a User Interface Agent that will constitute a sort of translator between the human user and the personal assistant [16]. The interface agent will be responsible for interacting with the human user and for making the other agents transparent to

him. Moreover it is able to understand the user's requests and translate them for the other agents.

By introducing the user interface agent humans will be viewed as agents in their own right and their interactions with software and agents may also be achieved using ACL. This specification was used as a starting point in our project.

2.3.2 Personal Assistant Building Approaches

2.3.2.1 Knowledge-based approach

The knowledge-based approach consists in endowing the agent with knowledge about the user and the application. Maes [15] defines two problems that we can face when we try to build this kind of agents. The first one is that of competence. The agent must be able to determine when it has acquired enough knowledge to help the user. The second problem is that of trust. The user must feel comfortable when he delegates a task to his agent. Moreover, this approach demands a lot of works from the engineer to provide the agent with all the required knowledge and they are fixed once and for all at the design and development stage. The agent cannot be customised according to its owner's behaviour and habits once its owner launches it.

2.3.2.2 Machine learning approach

Maes [15] defines an alternative approach, which is the machine learning approach. It consists of providing the agent with just a little background. The agent has the ability to learn from the user's behaviour and to extract some rules about the user's habits. This approach rests upon two hypotheses: The first one is that the application includes many repetitive actions. The second is that these repetitive actions are different from one user to another.

The agent has four different ways to learn from its owner. The first one consists of the fact that the agent is always looking what the user is doing. It extracts some rules according to the user

repetitive actions. The second method consists of the direct and indirect feedback that it gets from the user. For example, the user neglects continuously some suggestions given by his agent. Another method is by giving explicit examples to the agent showing him what to do in a particular situation. Finally, the agent can learn by asking other agents that are performing similar task [15].

The advantages of this approach are [15]: it requires less work from the end-user and developer, the agent is better able to adapt to the user over time, and the knowledge the agent gains can be shared with other individuals in the community.

This approach has been applied to four applications: electronic mail handling, meeting scheduling, electronic news filtering (Usenet Netnews), recommending some items (books, music or other forms of entertainment).

2.3.2.3 Collaborative agent approach

Collaborative agent approach aims to remedy the deficiencies that can be found with the machine learning approach. Indeed, even though the machine learning approach has many advantages, it has two drawbacks. The first one is the slow learning curve of the agents, especially at the beginning. The second one is that even after learning the user behaviour the agent still experiencess trouble when encountering a new situation.

The solution is the collaborative agent approach. A collaborative interface agent will allow the agent to take advantages of the ability and the experience of other existing agents and to speed up its learning curve.

Following this approach, Lashkari [14] defines two types of communication in its framework. A desperate communication occurs when the agent does not know how to deal with a given situation. An exploratory communication is used when the agent has a good idea about what

action it has to perform in a specific situation but wants to ask other agents in order to know the degree of their competence. Obviously, agents cannot distinguish between a desperate communication and an exploratory communication.

In order to allow different agents to collaborate, a communication protocol should be designed. The communication protocol that has been defined can be briefly described as the following: The agents first register themselves with a "Bulletin Board Agent" that all the agents know. When an agent wishes to locate a peer it queries the bulletin board agent for the location. The agents then communicate between each other in a request and reply manner. However, an agent is not obliged to reply to a request [14].

They distinguish between two types of requests: one at the situation level collaboration and another one at the agent level collaboration. The first one refers to the case when a new situation occurs and an agent sends a desperate or exploratory request to some agents in the market. The second level occurs when an agent wants to know how much other agents trust a specific agent in the market.

Lashkari [14] describes an experiment that his group members have conducted. They wished to show that multi-agents collaboration is better than a single agent that is performing the same tasks. Their results do indicate that this is true. The new agents in the collaborative environment were brought up to speed much faster and were able to generate more accurate predictions sooner than agents that did not have collaborative assistance.

2.3.3 Domains Of Application

2.3.3.1 Network assistant

An example of a Network assistant is Narval [17]. It stands for Network Assistant Reasoning with a Validating Agent Language. It is an intelligent personal assistant; released to assist the

group of volunteers that translates the Linux Gazette into French, by taking care of most of the co-ordination-related work. The main task of Narval consists of managing the assignment of each article to the group of persons that offer to translate a specific article.

Narval is based on advanced techniques like petri-nets, rule-based systems and programming by contracts, planning, automated learning, component programming and XML. It executes recipes, which are sequences of actions linked by transitions. Conditions can be associated with transitions so that the execution path can be controlled.

Actions have complex prototypes that borrow from programming by contracts. A prototype specifies conditions on the XML nodes the action needs as input and will return as output, whether it can process a list or not. For each input and output a set of Xpath expressions is used to define the name of the tag, the value of an attribute. When executing the actions, Narval will use these expressions to detect errors and failures and abort the recipe if it has no way to recover.

Using a Graphical User Interface (GUI) a user can create a new recipe in a few mouse clicks by assembling building blocks and specifying the conditions associated to the transitions that link blocks together.

2.3.3.2 Travel assistant

Travel related services are increasingly gaining popularity in the electronic world. However, with the huge number of those services the research for access to those services has become very difficult and time-consuming for the user. An urgent need for convenient assistance emerges as a consequence of this wide diversity of services. Personal assistants can be very helpful to facilitate the task to the user during the planning and the execution phase of the travel. They will interact with the user and travel services providers to offer the best deal to their owner. Some of the services that the agent can provide includes multi-modal route

planning, hotel and parking-lot reservations, individualised traffic guidance, cartography services, tourism information, plane reservation, metro guidance, weather conditions, public transportation, and special events [16].

The Personal Travel Assistant should offer the following functionality [16]:

- Comparison of service offerings. The agent should search and evaluate different services available and choose the best deal depending on the user's criteria like the price or other user's preferences,
- The ability of the user to check on what his assistant is doing and to change his behaviour at any time,
- Notification of the user on every significant event,
- Negotiation mechanisms in order to offer some flexibility to the user to specify some features like price.

The task of the personal travel assistant is complex insofar as it includes several features [16]:

- Information retrieval. The travel service should provide Web-based access and research,
- Scheduling. The travel service should interact with personal calendars and schedules, Calendar, e-mail and other application,
- End-user profiling. User likes and dislikes,
- Agent mobility. It is desirable because of the amount of information that the agent should retrieve and the negotiation protocol that may take place.

2.3.3.3 Internet assistant

The World Wide Web (WWW) has opened the Internet to the general public. By browsing on the net, users have access to a huge volume of information. However, the browsing mechanism has two major weaknesses. The first one is that users get easily lost in the cyberspace. Moreover, some time they spend a lot of time looking for what they have already found a week before. Another problem is that the Internet is still relatively slow and links get always broken because of current problems on servers or some sub-networks [18].

The Personal Internet Assistant PIA [18] is an agent that gives support in accessing World Wide Web and surfing through the Internet. Based on a machine-learning approach, PIA observes the user while downloading Internet pages. It extracts some rules according to his Web accesses and tries to keep a record of certain behaviour patterns.

Using those patterns, it estimates the next access to the regularly used pages, automatically downloads them in the background and saves them in a cache on the local machine. The next time that the browser is being used to access one of these pages, it just fetches it from the local cache.

The basic tasks of the Personal Internet Assistant are as follow [18]:

- Get the information about the Internet accesses of the user. Those information are data about different URLs that the user tried to download with its specific times and dates. This data should be stored in a convenient and flexible data structure.
- Drive some conclusions from this data. The agent should look for the possibility to make any assumptions about the probability to access a given page and find some regularity in order to find a formula to calculate the next access.

- Download a specific page in the background at a specific time without any user interaction and save it in a local cache structure. The browser has to look up the local cache before accessing the web.

The main concept consists of building a proxy server on the local machine. In every browser, the user can specify a proxy server very conveniently, and he can turn it off if he wishes. The advantage of this approach is that the caching strategy is browser independent [18].

The PIA was implemented in Java and is therefore platform independent. Through a built-in option window, it can be easily adjusted to suit the personal machine and user-preferred behaviour. It is downloadable anywhere from the net and can be run from any computer.

2.3.3.4 E-commerce assistant

One important area where personal assistants can be very useful is electronic commerce. Indeed, software agents are playing an increasing role as mediators in E-commerce. They assist users specifying their need, looking for the desired item, negotiating the price and if authorised comple the purchase.

2.3.3.5 E-mail assistant

An E-mail assistant task consists in managing the user electronic mail. It can delete, prioritise, forward, sort, and archive mail on behalf of its owner according to the user specification or his previous repetitive actions.

Maxim [15] is a personal assistant for electronic mail implemented in Macintosh Common Lisp. It memorises all the situation-action pairs generated by the user by observing his repetitive actions. When a new situation occurs it compares the current situation with the previous saved in its memory and decides which action it has to perform. Each situation is described by a set of features such as the sender or the receiver of the message, the Cc: list, the

date, the subject keywords, whether the message was read or not. The metric distance used to compare different situations is a weighted sum of the differences for the features that make up a situation. It is the personal assistant that determines the weight of each situation by comparing the relationship between a feature and the actions performed. The agent also determines how confident it is in a specific action. The confidence is measured by three elements: whether the nearest neighbour recommend the action or not, the distance between the current situation and the neighbour situation, how many examples did the agent memorise.

The agent uses facial expressions to communicate its internal state to the user. There are faces for "thinking" (the agent is comparing the current situation to memorised situations), "working" (the agent is automating an action), "suggestion" (the agent has a suggestion), and "unsure" (the agent does not have enough confidence in its suggestion). The purpose of those facial expressions is to give the user an idea about what the agent is doing easily [15].

2.3.3.6 Meeting-scheduling assistant

Meeting scheduling is a typical application where agent technology can be applied. The meeting-scheduling agent can assist the user in managing the meeting schedule. It can accept, reject, schedule, reschedule, and negotiate meeting time on behalf of the user.

Agent for Meeting-Scheduling [15] is a system fully implemented in MCL (Macintosh Common Lisp). It uses the learning base-approach described before. Indeed, the meeting scheduling application fulfils the requirements necessary to apply this approach. They are repetitive tasks and every action depends on a specific user.

The meeting-scheduling agent had been tested with real users and the results were very encouraging.

2.3.3.7 News Filtering assistant

News filtering is another application of agents. Even though the application doesn't need much intelligence it can be very time-consuming for a human user. With agent technology, the user can create one or many agents for each news and train them on what items should be selected according to some features chosen and described before by the human user. Once an agent is launched, it starts recommending some elements to his owner. The user then can select some important elements and gives feedback to his agent about them in order to increase the probability to recommend similar article in the future.

NewT [15] is a system fully implemented in C++ on a Unix platform that helps users to filter Usenet Netnews. The current implementation of NewT is content based. It means that users can give feedback only according to the content of the article. On the other hand, it does not use social filtering. Indeed, agents cannot use the recommendation of other agents that are performing the same task.

NewT system had been tested on group of twelve persons and the results obtained were very promising [15].

2.4 Challenging issues

There are already a lot of agents in the software world that can be included in the category of personal assistants. They help people accomplish their routine jobs in different areas and domains using different technologies and concepts at both the design and implementation level. A simple overview of those personal assistants allows us to make some remarks.

First of all, all the personal assistants that were described before are conceived to provide a single service to the human user (i.e: Network assistant, Travel assistant, Internet assistant, E-commerce assistant, E-mail assistant, Meeting scheduling assistant, News filtering assistant...).

The design and the implementation of each personal assistant varies from one to another according to the nature of each service. Consequently, there is a waste of time and duplication of effort at the design and the implementation level. Indeed, each personal assistant demands lot of work from the designer and the developer to find out the design, the concept, and the technology adequate for each type of personal assistant. Till now, there is no standard model that can be applied to all kinds of personal assistants regardless of the type of the application. Consequently, the user is obliged to learn how to use a different personal assistant for each service and find himself obliged to work with heterogeneous platforms, technologies and concepts.

The second drawback is that the existing personal assistants do not take advantage of the ability and the experience of other existing agents. Indeed, most of them are not collaborative since no communication mechanism can be defined given the heterogeneity of the platforms and the technologies used. In our concept, we think that a personal assistant does not have to deal with all the user requests. It can ask for help from other specialised agents to achieve tasks that are not part of its initial job. This approach requires less work from the programmer and makes him focus on a single service. The concept of agent that consists of abstracting and embodying each entity in an agent responsible for fulfilling a specific task fit well with our approach. The collaboration between the different personal assistants can be expressed in terms of agent interaction. Moreover, since the emergence of the agent concept, many agents' platforms have been designed and implemented. They facilitate agent management and communication. The implementation of a personal assistant using one of those platforms will relieve us from redefining some aspects such the communication mechanism.

Moreover, there are some services that are used generally together. For example, a meeting-scheduling assistant will have obviously to manage also the user agenda and address book. Therefore, it will be very useful that a single personal assistant include all those applications in

order to enhance the quality of the service offered to the human user. This approach will relieve the user from working with a different interface for each application.

On the other hand, most of the existing personal assistants are not able to react to external requests without a direct intervention from the human user. In our concept, the personal assistant does not only assist the user in different tasks but also acts on his behalf according to the degree of trust that he has in him. A personal assistant can communicate with other agents and applications and takes some decisions without any intervention from the human user.

Finally, the nature of the personal assistant demands much more interactions with the human user than other types of agents. Consequently, enhanced user interfaces are required; especially those that are based on visual languages.

2.5 Proposed model

As mentioned in the introduction, we designed and developed our own model of personal assistants as a multi-agent system where each agent abstracts specific tasks. The model attempts to overcome the shortcomings that are present in the existing personal assistants and to address more appropriately the aforementioned challenging issues. The choice of our design was primarily inspired by the following considerations:

Generic. The model should be generic and not oriented to satisfy the requirements of a specific application.

Enhanced User Interface. The Personal Assistant should provide interaction facilities with the user in order to facilitate the information exchange between the end-user and the personal assistant. Consequently, an enhanced user interface is required.

Extendibility. The personal assistant should offer the possibility to add new services that can arise in the future.

Openness. It represents the capability of the system to interact with the external world and to profit from services provided by others with similar systems.

Security. The openness of the system makes it vulnerable from the security point of view. It is crucial that our general architecture takes into consideration this point.

Flexibility. It represents the ability of the system to change some features of its components without demanding much work at the conception and implementation level.

Co-operation. Different services are not separate entities but they should be used together in order to offer a better service to the end-user.

2.6 Summary

The purpose of this chapter was to introduce what an agent is, and different technologies that contributed to this increasing widespread. It has also described some examples of personal assistants and different approaches applied to building a personal assistant. We ended the chapter by discussing the advantages and drawbacks of the existing personal assistants and describing our proposed approach. The next chapter is dedicated to detailing the design of our model of personal assistants.

CHAPTER 3: DESIGN AND CONCEPT

3.1 Introduction

Maes defines interface agents as "*Computer programs that employ artificial intelligence techniques to provide active personal assistance to a user with computer-based tasks*"[1]. This type of agents are viewed as personal assistants that give support to users in managing different tasks such as e-mail filtering, information retrieval, and meeting scheduling. The main objective of a personal assistant is to relieve the user from routine tasks and help him focus on more important things. In our concept, the role of personal assistants is not limited in providing some services to the user, but they have other qualities that allow them to take some decisions on his behalf.

To achieve its role, our personal assistant replies to some basic characteristics. The first one is autonomy. The personal assistant should be able to act, to some extent, independently from the user. The personal assistant is able to control over its internal state without any intervention from the user. The second quality is knowledge. It includes facts and rules about the end-user and vis-à-vis the external world. The personal assistant will use this knowledge to reply to external requests without the user intervention. The last one is interaction capability that makes him able to communicate with its environment. This interaction can go from a simple information exchange to an enhanced negotiation mechanism between the personal assistant and other agents.

In this Chapter we will explain our design choice, present the system architecture, describe in details its components, and explain the system interactions. The last section will be dedicated to the application of our model of personal assistant in the V-team collaboration project.

3.2 Objective

This part of our work has two objectives. The first one is to design a general model of personal assistants that provide a large variety of different services to the end user. The second objective is to apply this model in the context of the V-team collaboration project.

3.3 Model and Architecture

3.3.1 Design Choices

Our system architecture is based on the metaphor of software agents. We choose to use the agent concept for three reasons. First of all, as described in chapter 2, agent technologies have increasingly gained popularity in the software world since their emergence at the beginning of the last decade. They represent a new vision that promises to replace the object oriented programming language. Many platforms have been developed to support agent technologies. Consequently, it would be a good idea to take advantage of the work that has been done in this area to attain our objectives since the use of agent technologies will relieve us from redefining some aspects like the communication mechanism between the PA and external agents.

Moreover, there are already many personal assistants that provide active assistance to the human user. The design of an agent-based personal assistant will allow it to profit from a large range of services provided by the existing personal assistants.

Finally, the personal assistant constitutes a part of the V-team project that is based on the agents' architecture. Our personal assistant will responsible for interacting with V-team project agents and we thought that the use of agent technologies would be more appropriate than a Client \Server concept or CORBA or any other communication middleware.

The design that we adopted follows the separation of concerns concept that represents one of the most important principles in software engineering. The appropriate application of the concept helps to reduce software complexity and improve the comprehensibility, promote tractability, facilitate reuse, non-invasive adaptation, customisation and evolution, and simply component integration. Indeed, all the functional components of the system were embodied and abstracted through agents. By breaking down the architecture into clearly defined roles associated to separates agents, the management of the system can be expressed in terms of agent interactions. Moreover, the architecture becomes more modular and extensible in its design, and facilitates the communication mechanisms to enhance the system interoperability.

When designing and developing our personal assistant, the aim was to satisfy some requirements that have been listed earlier in chapter 2. Namely they are extendibility, openness, and flexibility. Indeed, the introduction of a User Interface Agent as a separate entity gives the system more extendibility. Other existing software can be added to this application or can replace the existing Graphical User Interface without a need to change the architecture of the system. The user interface agent is able to understand what type of needs the user is expressing, formulate his requests, and delegate the tasks to the other internal agents of the system (Figure 3.1). Following FIPA specifications that have introduced the user interface agent and that constitute a starting point of our application, the user interface agent will constitute a sort of translator between the human user and the system agents.

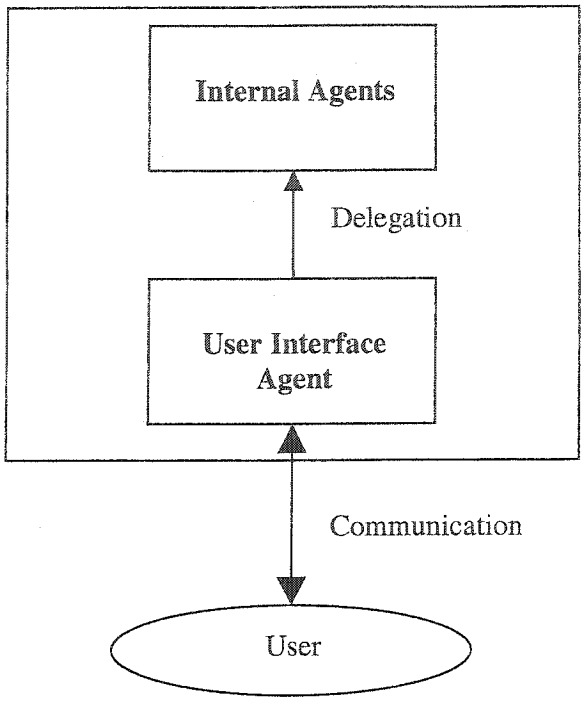

FIGURE 3.1 THE STRUCTURE OF THE PERSONAL ASSISTANT

We chose also to separate the Information Management Agent from the Personal Agent for security reasons. The fact that the Personal Assistant is capable to communicate with other agents makes him vulnerable from the security point of view. External entities have access to the private information of the user without any control. The main scenario that we adopted is that the Personal Assistant transmits the request from the foreign agent to the Information Management Agent. The Information Management Agent will check if it doesn't have any constraints that prohibit the disclosure of the requested information and then it can send a response to the Personal Assistant that will transmit it in his turn to the external agent.

Another case is when the human user wants to perform an action that he is not allowed to do, the Information Management Agent will have the charge of notifying him that he do not have the permission.

The Information Management Agent can be viewed as a central authority that is responsible for enforcing certain policies in the system and protecting the system from malicious access. It

helps control the overall system from both external entities wishing to access the internal data of the system and internal entities to perform wrong actions.

Another reason for having the Information Management Agent as a separate entity is to provide a certain independence of the system from the physical representation of the system information. The structure of this information and the data repository could be changed without redeveloping other entities.

The Personal Agent is the entity that constitutes the link between the system and the external world. The system is open to the external world since it can interact easily with it via the Personal Agent. However, the internal architecture of the system is hidden from other agents or applications that may interact with the system. External entities cannot know what is the exact architecture of the system, what are its different components, and how the user's information are presented. From an external agent point of view there is only one entity with which it can interact and co-operate.

3.3.2 General Architecture

The design requirements led us to decompose our personal assistant into three essential and comprehensive parts according to the concept, the goal, and the role of each part of the design. Following the agent paradigm, every entity was embodied and abstracted into an agent responsible for fulfilling a specific task. Namely, they are the User Interface Agent (UIA), Information Management Agent (IMA), and Personal Agent (PA). They communicate with each other and also with external agents using the Agent Communication Language (ACL) in order to provide a better service to the user.

Figure 3.2 gives an overview of different components and different interactions in the system.

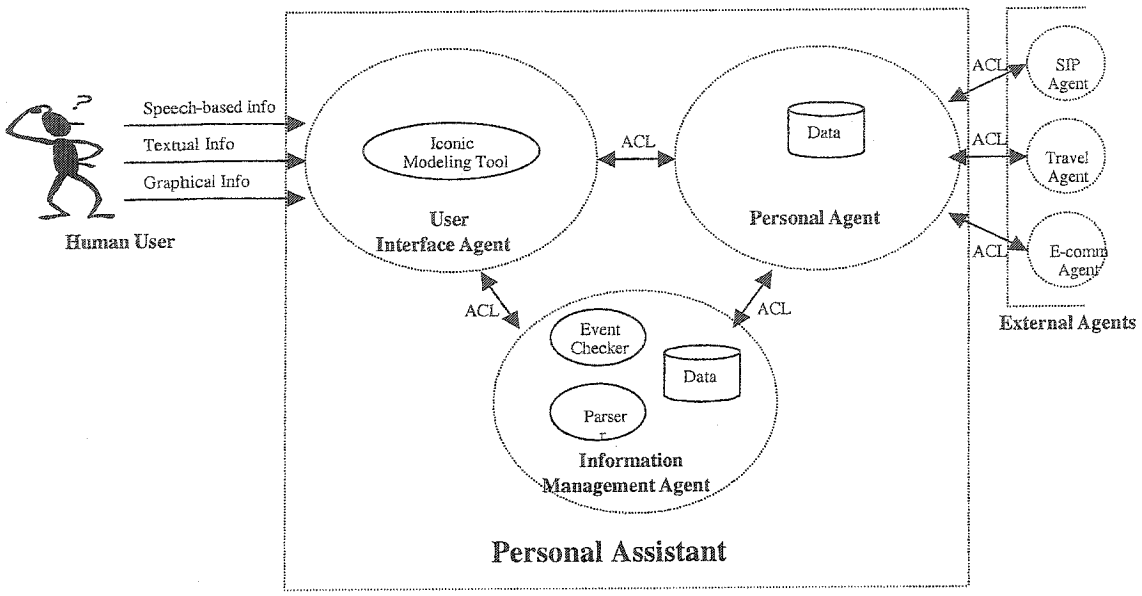

FIGURE 3.2: GENERAL ARCHITECTURE OF THE PERSONAL ASSISTANT

In our concept the User Interface Agent is an agent that is responsible for direct interaction with the human user. Briefly, its main task is to translate the human language that can be speech-based, Graphical, and/or textual information to Agent Communication Language (ACL) and vice versa. The Information Management Agent is responsible for storing information, replying to the system agents' requests, and controlling the system. Finally the Personal Agent is responsible for communicating with external agents and transmitting tasks to them. The following sections provide a description of each component.

3.3.3 User Interface Agent

3.3.3.1 Definition and role

The User Interface Agent is responsible for translating the human language into the agent communication language and vice versa. It facilitates the communication between the human user and the system, assists the user in formulating his likes and dislike, and presents the required information in a comprehensive language to the human user.

In our model, we decided to provide an enhanced Graphical User Interface using a visual language in order to facilitate the interaction between the human user and the personal assistant. The Iconic Modelling Tool (IMT) [19] represents an important component of the User Interface Agent.

Different communicative acts between the user and the system can be summarised as the following:

- U → S: User gives an order to execute a task (e.g. schedule a meeting, purchase an item).
- S → U: System sends the progress status of the task (e.g. success or failure with the reasons).
- U → S: User specifies his likes, dislikes, his constraints, and his rules (e.g. preferences, availability)
- S → U: System sends a notification (e.g. the meeting time).
- S → U: System asks for permission to perform a task (e.g. permission to buy an item).
- U → S: User sends the permission to commit the task.

The User Interface Agent should provide the Application-Program Interface API for different applications and area. In our model, we provide a common interface that is independent from the application itself. This may be accomplished by providing a sort of generalization based on the common points between all the applications. Indeed, each application can be viewed as a set of actions to perform, a set of information to provide, and a set of rules and constraints to specify.

We choose to divide each manipulation of the GUI into four steps. In the first step the user choose the area of the application then he chooses the action he wants to perform. After that, he specifies his constraints or/and rules.

3.3.3.2 Different components

Iconic Modelling Tool

Visual languages are ubiquitous in human culture and range from informal ambiguous sketches to rigorously defined technical diagrams such as electronic circuit design or musical notation [19]. They have become a key component of human-computer interaction with the advent of GUIs and remain an active research area.

Iconic Modelling Tools [19] as a visual language has been used several time; especially in mobile agent applications. It makes non-programmers able to customize the agent behaviour without knowing any programming or Script language.

We use the Iconic Modelling Tool to help users construct their requests in an efficient way. The UIA interface based on an Iconic Modelling Tool (IMT) uses a simple, easy to understand and small set of meaningful icons. It is based on the classical easy-to-understand timing-like interaction diagrams where each step is abstracted to a vertical line with appropriate descriptive icons at the top of the line.

By associating icons spatially and semantically, and providing textual information when required, the IMT creates an efficient and effective environment for the end user. Textual information is especially used to represent features that would be difficult to specify visually. The IMT is only responsible for schematizing the user requests. Other classes translate both the visual and textual information into a standard format based on parameter-value pair (e.g. Meetting_Hour: February 1st 3:PM) and encapsulate the result into ACL messages. The ACL

messages will be then sent to the Personal Agent or the User Interface Agent according to the nature of the task.

3.3.4 Information Management Agent

3.3.4.1 Definition and role

The Information Management Agent IMA is responsible for controlling and managing the system information. The IMA can also be viewed as a police agent that controls the system. Basically, every external request passes automatically by the IMA. The IMA checks whether the external agent can accede to the information or not. Other tasks of the IMA include storing the system information, parsing its database, and checking the new events.

Following is a description of different components that constitute the Information Management Agent.

3.3.4.2 Different components

From the tasks described previously, we can distinguish between three different components. A data repository to store the system information, a parser to extract this information, and a daemon that will continuously check for new events.

Data repository

We choose to stock the system information into XML files which provide a way to store structured data in text files. The advantage of using XML files rather than relational database system is that it is more extendible and accepts huge and heterogeneous kind of information. Indeed, adding other applications to the system will not require the redefinition of the database structure. On the other hand, some of the system information cannot be presented with relational database structure. Especially, when it comes to store the user constraints.

The drawbacks of this choice are that we will have to construct our own model to represent different kinds of system data and to implement a parser to extract those data.

The principle of tags and values that characterises XML files is suitable for representing the system information in a hierarchical format similar to directory hierarchy where each directory comprises several subdirectories.

The stored information can be a simple parameter-value pair or a complex constraint. To represent the constraint using XML tags and value format we choose to translate each condition that respects the following format (condition1 op condition2) to (op condition1 condition2). The operation can be "or" or "and" This format is much more suitable for XML files and makes the parsing much less complex.

One example of those constraints is the following:

```
<?xml version="1.0"?>
<UIA>
<Agenda>
<Or>
<And>
<Type>Internal</Type>
<Subject>Virtual Team Meeting</Subject>
</And>
<Period>
<From>03/11/00 2: 30 PM</From>
<To>03/11/00 5: 30 PM</To>
</Period>
</Or>
<Status>Y</Status>
</Agenda>
</UIA>
```

Parser

The second component of IMA is the parser. It is responsible for replying to the Personal Agent requests and triggering new actions. To accomplish its tasks, it has to parse the XML file and to check whether the constraint is satisfied or not according to the constraints submitted by the Personal Agent. It can reply by affirmative or negative or unknown-response

in case it do not have enough information to answer. In this case, it sends the query to the human user to answer directly to the request.

Another task handled by the parser is when some conditions become true and trigger some new actions. The parser then checks the conditions, extracts the actions that should be performed, and sends the task to the personal agent.

Event Checker

The Event Checker is a process that is continuously running to check for new events. When the IMA receives a request to perform an action upon some conditions, it sends to the Event Checker the condition as a parameter-value pair. Once a new event happens and the condition becomes true, the Event Checker sends the parameter and the value to the parser. The parser then looks for the actions that should be performed and sends an order to the personal assistant The personal assistant then directs the task to the adequate agent in the market or sends the information to the human user via the User Interface Agent.

3.3.5 Personal Agent

3.3.5.1 Definition and role

The Personal Agent is responsible for communicating with external agents. It represents a sort of system gateway to the external world. It allows the user to have access to a daily growing number of services provided by different agents. The Personal Agent has information about existing agents and the services they provide. Consequently, it can ask for help from other agents specialized in specific tasks. The Personal Agent also replies to external agent requests on behalf of the user without the user's intervention.

The communication between the Personal Agent and external agents can go from a simple information exchange to an enhanced negotiation mechanism.

Different communicative acts performed by the personal agent are summarised as follows:

- PA→External Agent: transfers a task from the human user to the adequate agent.
- External Agent→PA: Asks for information and replies to the PA requests.
- IMA→PA: Gets Information from the IMA.
- PA→ IMA: Forwards the request to the IMA.

3.3.5.2 Different components

Data Repository

To accomplish his task the Personal Agent should have a list describing all the services provided by external agents. This list is stored into XML file where each service is represented by an identifier, a description of the service and the name of the agent providing the service. The Personal Agent parses this file to find the adequate agent.

3.4 System Interactions

The system has two types of interactions: human to agent interaction and agent-to-agent interaction.

3.4.1 Human-Agent Interaction

Since humans do not speak ACL directly, some sort of a translation service from the human language to ACL is required. Indeed, this issue was resolved by creating the User Interface Agent that is responsible for translating human dialogue into agent dialogue; Therefore, the communication between the human user and the system will be accomplished via this special agent. From the system point of view it will view the user as an agent that communicates using agent language.

3.4.2 Agent-Agent Interaction

The communication between different agents of the system and with external agents is done with FIPA Agent Communication Language (ACL). It is based on speech theory: messages are actions, or communicative acts, as they are intended to perform some action by virtue of being sent [16].

A FIPA ACL message contains a set of one or more message elements. Precisely the elements needed for effective agent communication will vary according to the situation; the only element that is mandatory in all ACL message is the per-formative, although most ACL messages contain sender, receiver, and content elements [16].

3.4.3 Communication Protocol

An overview of different interactions that the system should handle allows us to distinguish between three different categories of scenarios. The following is a summary of those scenarios with some examples. (Figure 3.3)

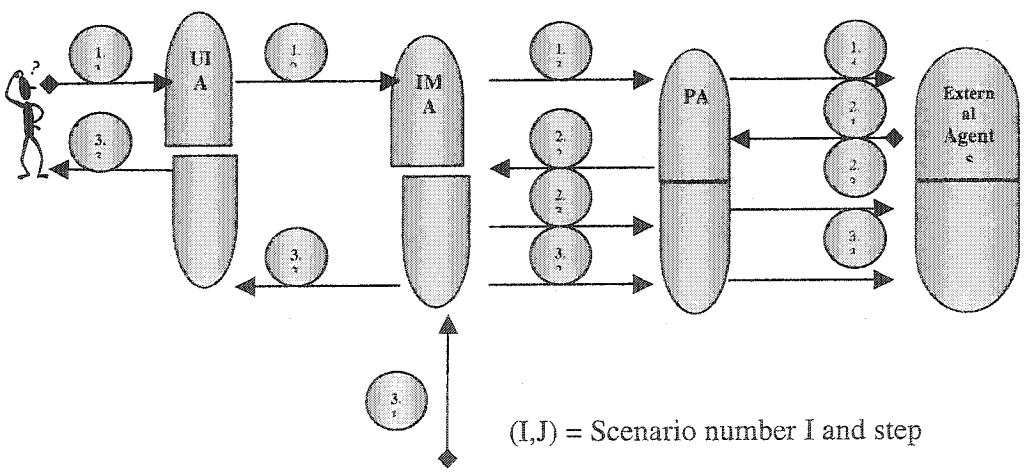

FIGURE 3.3: DIFFERENT SCENARIOS

The First category's main scenario is the following: The user sends an order, via the User Interface Agent to his personal assistant to accomplish a specific task. The Information Management Agent must validate the order before it transmits it to the Personal Agent. The Personal Agent then, looks for the agent in his list that can provide the service and contacts it to accomplish the task. Finally, The Personal Agent reports to the human user via the User Interface Agent.

Examples of this scenario are: organising meetings, selling/buying items, and planning travels.

Figure 3.4 illustrates the important steps involved in this scenario.

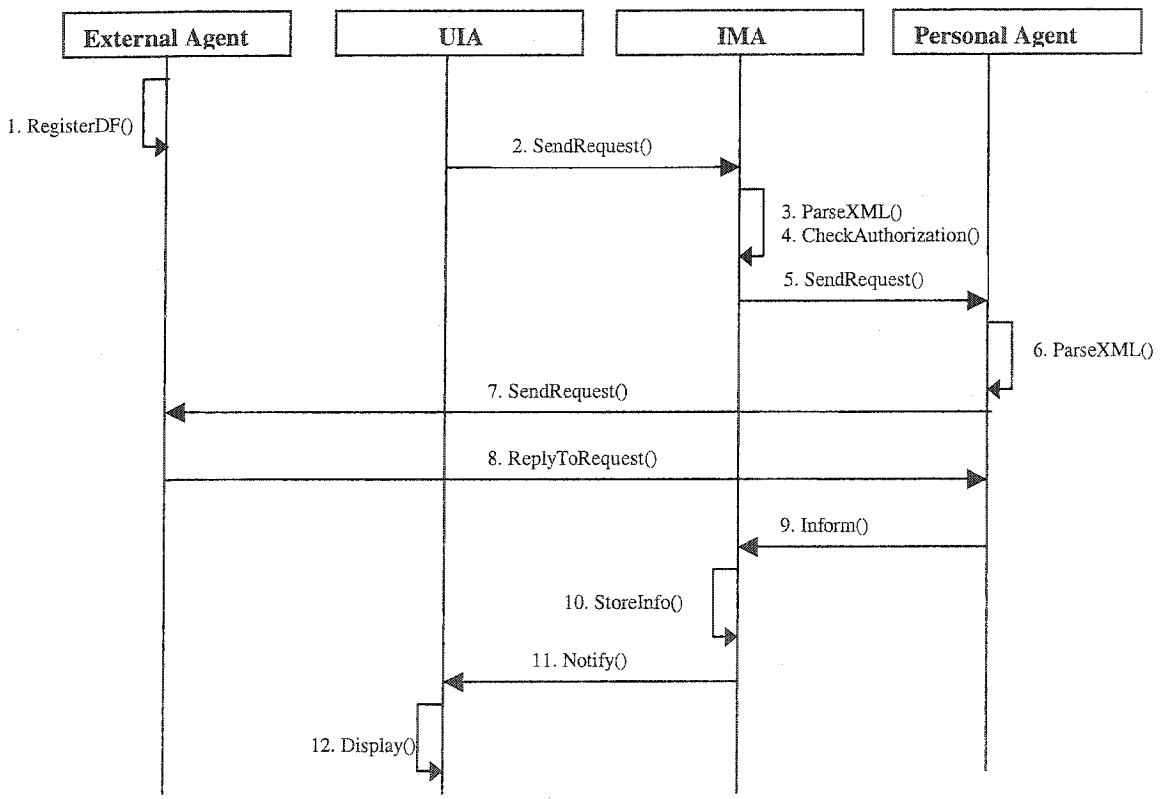

FIGURE 3.4: THE PROTOCOL OF USING EXTERNAL SERVICES

The steps are as follows:

1. Agents wishing to provide a specific service register in the Directory Facilitator.

2. The User asks his Personal Assistant for a specific service. He should give it a description of the task and the necessary parameters to accomplish its duty.

3. The Personal Assistant parses the list and designates the first agent it finds and that can provide the service.

4. The Personal Assistant contacts the agent in order to transmit the task to it. If the agent is available, it achieves its task.

The Second category is when an external agent asks for information from the Personal Agent. The Personal Agent transmits the request to the Information Management Agent that will consult its files in order to get the required information and reply the Personal Agent.

Examples of this scenario are: asking for the availability of a user, and requesting the price of an item to buy or sell.

Figure 3.5 describes the protocol used in this category of scenario:

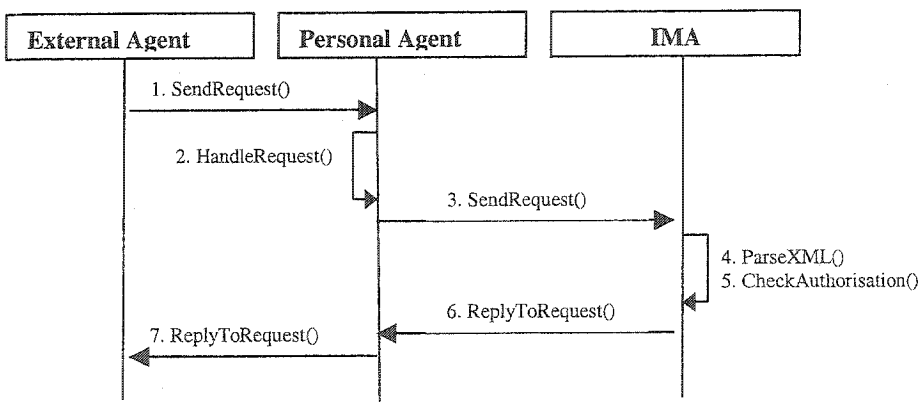

FIGURE 3.5: THE PROTOCOL OF REPLYING TO EXTERNAL REQUESTS

The Third category is when a new event happens. The Information Management Agent checks the new event, notifies the Personal Agent that will transmit the task to the adequate agent to perform the action. The agent can be external or the User Interface Agent.

Examples of this scenario include: calling for a meeting at the end of each month, forwarding every mail received from P1 to all the members of the team, and issuing a reminder to all the participants of a meeting.

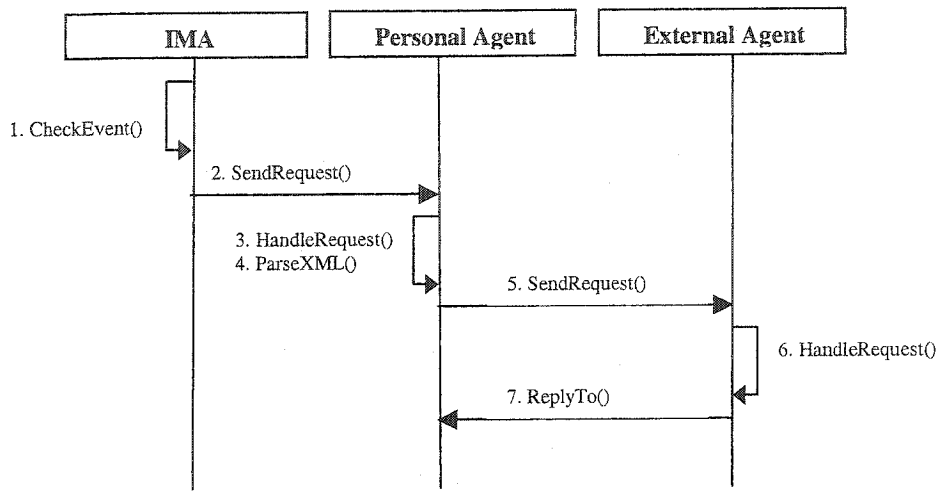

FIGURE 3.6: THE PROTOCOL FOLLOWED IN THE CASE OF NEW EVENTS

3.5 Personal Assistant In V-team Collaboration Project

The V-team project aims to create a collaborative environment for virtual teams. The Personal Assistant in this case is responsible for reflecting the user visibility and availability, scheduling meetings, and assisting the user via a graphical user interface. In this section we will describe briefly the project and some relevant scenarios showing the application of our model of Personal Assistant.

3.5.1 Overview of V-team Project

V-team is a collaboration project that aims to provide a collaborative environment for virtual teams. Based on the agent architecture, it uses some existing services and protocols (ex SIP, RSVP, RTP, RTCP...) to offer to the participants the basic services needed for the management of a virtual team. Some of those services are virtual team creation, Scheduling & notification, Conferencing, and application handling [20].

The V-team project introduces a layered architecture where each layer uses the services of a lower layer and provides services to the upper layer. Basically we can distinguish between three principal layers. The first one is the network control and services. It is concerned with the network and its basic services such as the quality of service and agent platform. The second is the virtual team collaborative environment that represents basic services to manage the virtual team layer. Finally, the participant interface layer represents the agents system oriented to the human user [20].

The following picture depicts the relationship between the personal assistant and the V-team project agents.

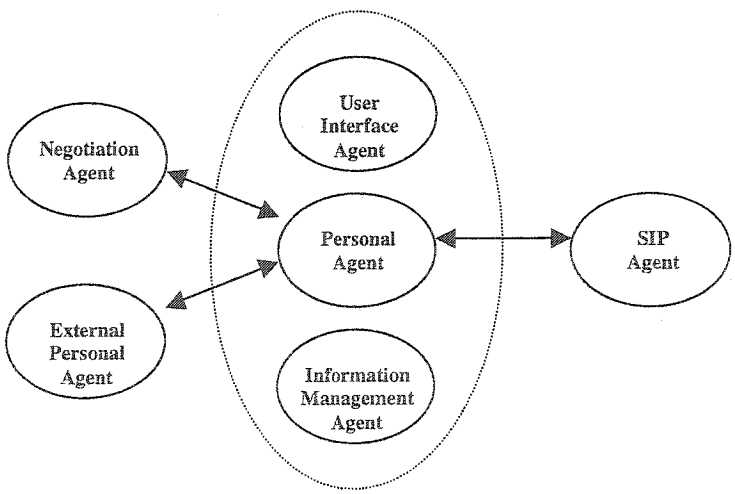

FIGURE 3.7 SYSTEM INTERACTION WITH DIFFERENT V-TEAM COLLABORATION PROJECT

3.5.2 Services of The Personal Assistant

The Personal Assistant provides a common interface from the end-user terminal to a wide range of services. Those services can be part of the personal assistant or provided by other agents.

Basically, the internal services are those that the human user uses daily such as Agenda, Address Book, E-mail, and/or other services. Those services will also allow the personal assistant to construct its base of knowledge. Using this base of knowledge, the personal assistant will be able to take some decision on behalf of its owner.

External services are, basically, those that the user may use just occasionally such as E-commerce agents; travel planning agents, and the Meeting Negotiation agents. The human user can simply ask his Personal Assistant to find out an agent that can provide certain services and gives him the required information and instructions to accomplish the task. This can be easily provided since agents are social and can interact and co-operate with each other to fulfil their tasks.

The following figure gives a view of different types of services that can be provided by a personal assistant.

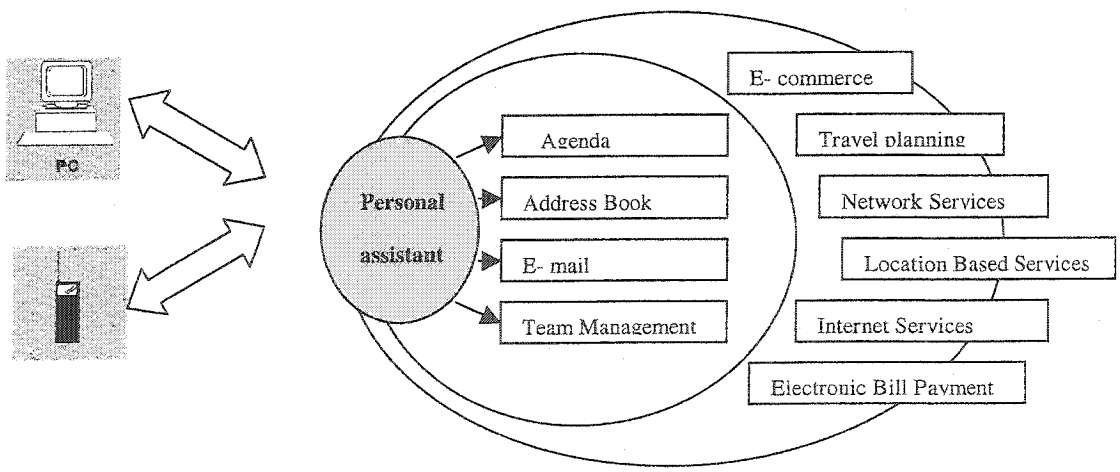

FIGURE 3.8: VIEW OF PERSONAL AGENT SERVICES

In the following sections, we will describe the internal services in the context of the V-team Collaboration project and give some examples of services the user can access via his Personal Assistant.

3.5.2.1 Internal services

Team Management

We include team management facility as an internal service of our personal assistant. It includes basic functionality needed for team management such as team creation, the modification of the team members, and some other inseparable services such as meeting scheduling services.

- **Team creation:** Team creation includes the following information: Identification of the team, Team project, Members, and Meeting schedule by default.

- **Meeting Scheduling:** One central service in team management is meeting scheduling. The personal assistant should help the user organise the meeting without much intervention from both the organiser and the other participants. To achieve this task, functionality should include identifying the appropriate time for all the participants, issuing a reminder to all the participants, and cancelling meetings.

  The identification of the appropriate time needs many interactions between different PAs. The PA of each participant must co-operate to ensure that the time of the meeting is adequate for all the team members.

  The simplest scenario is that the chairman specifies the time and the duration of the meeting and all the participants are required to attend the meeting. Another scenario is that the chairman specifies a set of times to start the meeting and the duration. The negotiation agent plays the role of a broker that interacts with different PAs to

determine the appropriate schedule. The broker should, obviously, have an enhanced negotiation protocol.

The agenda of the user is essential to achieve the negotiation. Obviously, the PA uses it to determine whether his owner will be available in a specific time or not.

Information Management Services

- **Agenda:** It reflects the user availability and visibility. The user has the possibility to specify his conditions in form of constraints or rules. In general, constraints parameters are not limited to the time. The user can have some preferences depending on the type of the meeting, the subject, the location, and/or other constraints. The user can also represent his availability in form of rules. For example: Not available each Monday from 10 to 12 PM.

- **Complementary services:** Other complementary services can be included like the address book and e-mail. Those services will be used together in order to provide enhanced functionality to the users insofar as they will facilitate the contact between different members and relieve the user from some annoying tasks like managing the address book, filtering e-mail, and sending back answers.

3.5.2.2 External services

One of the most important characteristics of agents is communication. This property allows them to profit from services offered by external agents to enhance the quality of the service offered to their owner. Indeed, from the point of view of the user it is his personal assistant that provides that service. The user will not have to look for other agents, but the task is transmitted to his personal assistant. Basically, the personal assistant has a list of the available agents and a

description of the services that they offer. Agents have to register in a common blackboard in order to be transparent to the personal assistant and to communicate with it.

SIP Agent

The external agent that we used is the SIP agent that is responsible for the session initiation. SIP, the Session Initiation Protocol, emerges as the protocol of choice for Internet conferencing, telephony, presence, events notification and instant messaging [21].

It offers the advantages of being:

- Open, simple, extensible, and lightweight protocol.
- Internet heritage- easier to integrate with telephony and Internet functions.
- Already implemented or planned on most softswitches and gateways.
- Allows control of bear path.
- Supports multiple call legs.
- Same protocol used between services and call control entities.

Some of the services enabled by SIP are [21]:

- User Location: Determination of the end system to be used for communication
- Call Setup: Ringing and establishing call parameters at both called and calling party
- User Availability: determination of the willingness of the called party to engage in communications
- User Capability: determination of the media and media parameters to be used

- Call handling: The transfer and termination of calls.

Negotiation Agent

The negotiation agent is responsible for negotiating the meeting schedule. The negotiation protocol can go from a simple to an enhanced negotiation mechanism. The most important parameter in the negotiation mechanism is the time. One simple scenario is that the chairman chooses a number of possible times. The negotiation agent chooses the time when most of the group member can attend the meeting. A complex scenario consists of leaving it to the negotiation agent to decide the time that fit all the members of the team.

3.5.3 Application scenario

The following scenario gives an example of the use of our Personal Assistant in a collaborative environment as an agent that assists the user in performing his routine tasks and that acts and takes some decisions on his behalf. In collaboration with the SIP agent the Personal Assistant helps also to reflect the user visibility. The main scenario can be divided into three different parts according to different intervening entities that it includes.

The first part of the scenario is basically conducted by three intervening entities: the Human User, the User Interface Agent, and the Information Management Agent. It includes all the communication acts that should happen between the human user and his UIA. In this part, the human user provides the Personal Assistant with all the knowledge that it requires to perform its task. Different steps of the scenario are as follows:

- **Team Creation and Management:** The personal assistant creates the team via the UIA. The creation of the team consists in specifying its members, the chairman, and the

team's project. The UIA uses the Iconic Modelling Tool to help the user formulate his request. The team information is sent to the IMA to be stored.

- **Agenda Management:** Each member of the team specifies his constraints and his availability via the UIA. The UIA offers a set of criteria with which the user can specify his availability. This information should be sent as well to the IMA.

- **Meeting Scheduling.** Once the chairman decides to schedule a meeting he/she specifies the meeting features such as the objective of the meeting and the list of the participants. Then, he/she suggests a set of possible times and sends it to the IMA.

Figure 3.9 illustrates the process of creating the team, managing the agenda or initiating the meeting scheduling.

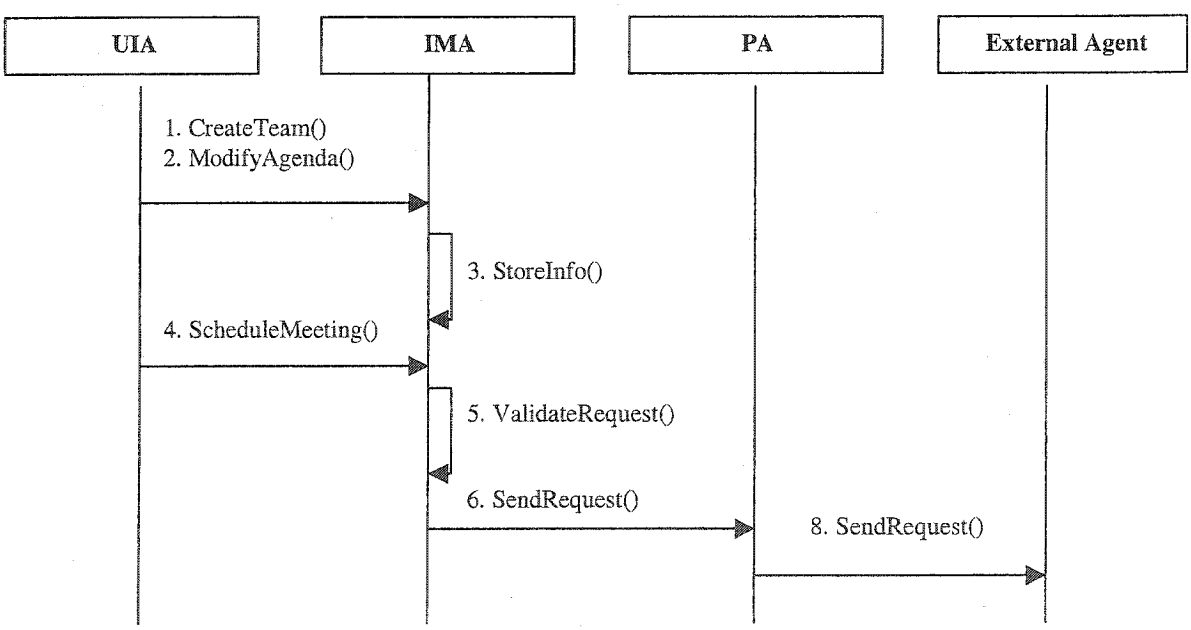

FIGURE 3.9 THE PROCESS OF TEAM CREATION, AGENDA MANAGEMENT, AND MEETING SCHEDULING

The second part is conducted without any intervention from the human user. The intervening entities are the Personal Agent, the Information Management Agent, and external agents. Namely, they are the SIP Agent and the agent responsible for negotiating the meeting schedule:

- **Meeting Negotiation.**

1. The Personal Assistant transfers the task of negotiating the meeting parameters to the agent responsible for negotiating the meeting schedule.

2. The Negotiating Agent checks the availability of each participant by communicating with its personal assistant

3. The Personal Assistant of each participant checks with its corresponding IMA whether his owner is available or not.

4. The negotiating agent collects all the responses, decides what is the adequate time to meet for all the participants and reports to each Personal Assistant that has been included in the negotiations. The Personal Agent then notifies the human user via the User Interface Agent.

Figure 3.10 illustrates the protocol of negotiating the meeting schedule.

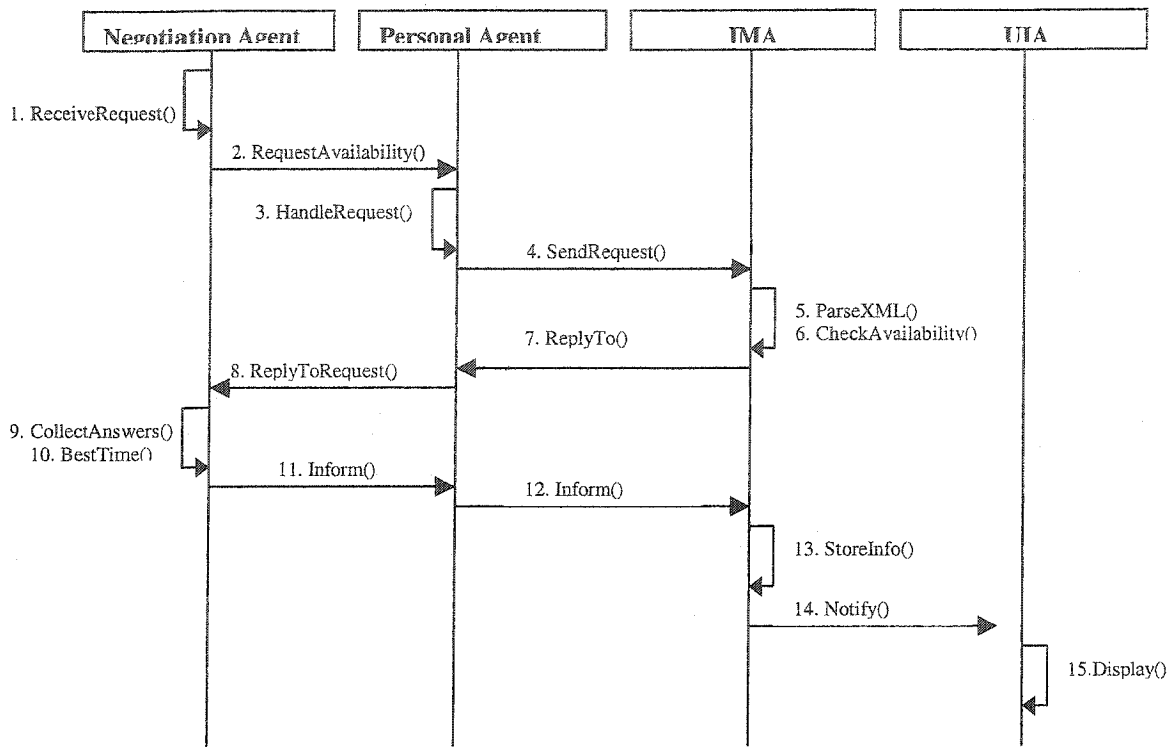

FIGURE 3.10: THE PROTOCOL OF MEETING NEGOTIATION

➢ Session Initialisation

The third part deals basically between different agents of the Personal Assistant and the human user. The steps are:

1. Once the meeting time arrives the SIP Agent sends an invitation to each participant using SIP services.

2. The Personal Agent of each participant receiving the invitation sends a request to his IMA to check whether the human user is available or not.

3. The IMA replies to the Personal Agent, which forwards it to the chairman. If the response is unknown the human user has to intervene to reply.

Once all the participants reply, the meeting session can begin by launching the team participant agent for each participant.

Figure 3.11 illustrates the process of session initialisation.

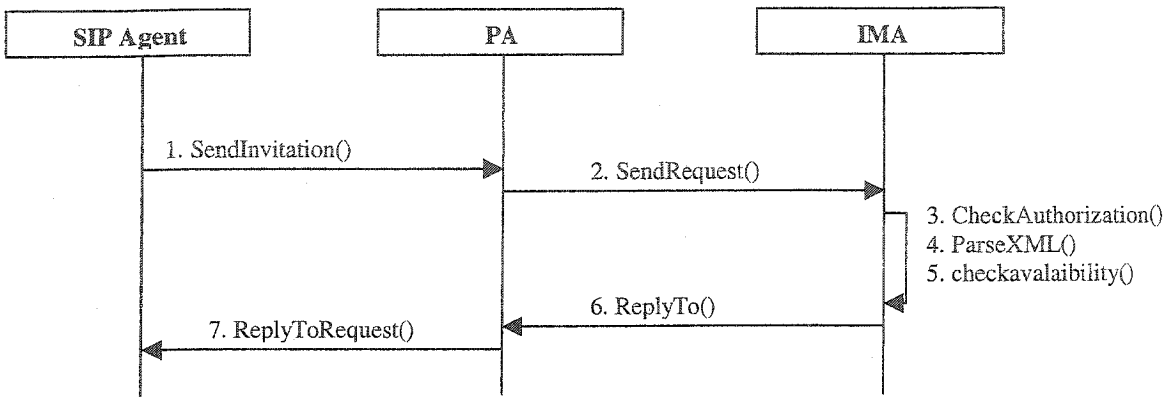

FIGURE 3.11: THE PROCESS OF SESSION INITIALISATION

The Personal Assistant also assists the user during the meeting session. One example is that the user can specify to his personal assistant via the UIA the time he should leave the meeting. The IMA will send a command to the personal assistant to notify the user that the time is up and that he has to leave.

- **User Mobility**

Another important aspect where the use of agent technologies can be very important is the user mobility. The Personal Assistant, by definition, belongs to the human user and is attached to him and not to a specific machine. The Personal Assistant should be able to follow his owner and to assist him when he chooses to move from one machine to another. The main scenario is as follows:

1. The user indicates using SIP that he will move to another address before leaving his current machine.

2. The SIP Agent sends an invitation to the permanents address of each participant

3. The SIP proxy server forwards the invitation to the current address of the participant.

4. The SIP Agent sends the new address of the user to the Personal Assistant that will move to the current address of his owner.

3.6 External Software Integration

The capability of the system to integrate existing software is a very important aspect since it will give the system more openness and extendibility and allow the personal assistant to acquire new capabilities; i.e, existing software can be used as interfaces instead of the interface offered by the User Interface Agent. Examples of those systems include among others: Address book, E-mail, Calendar, Telephone, Fax, and News.

In general, all software systems have a software description that defines the nature of the software system and how to connect to it. An agent called the Wrapper Agent could be introduced to connect the system to external software using wrapper technologies. This agent will allow the personal assistant to invoke commands on the software system by translating the messages contained in ACL messages into operations on the software system.

The wrapper services allow the agent to [16]:

- ✓ Connect to the software system,
- ✓ Invoke commands on the software system,
- ✓ Return the result of each operation,
- ✓ Query the properties of the software system,
- ✓ Set the parameters of a software system,

✓ Close the connection to the service.

The agent providing the wrapper services can be specific to one software system or a type of software systems. Since the system may include many wrapper agents, the Personal Agent should have a description of all the wrapper agents in the system and the description of the software that they are able to connect the system to.

The main scenario is that the wrapper agents register themselves in a sort of blackboard. The Personal Assistant that has a description of the services provided by existent software sends a request to the blackboard for the agent providing the wrapper service to the specific software. Once the blackboard returns the name of the agent, the Personal Assistant asks that agent to initialise the connection to the specific software and to invoke operations on it, after finishing the personal assistant closes the connection to the software used.

3.7 Advantages and Limitations

We presented in this chapter the application of our model of personal assistant in the context of the V-team collaboration project. Our personal assistant offers the human user a number of services. Namely team management, Agenda, Address book, and e-mail management . The most important service offered by our personal assistant is the meeting scheduling. By collaborating with the negotiation agent the personal assistant can help to organise a meeting without intervention from the user using two negotiation protocols. In the first protocol, the negotiation agent proposes a set of possible times then chooses the adequate time for all the group members according to the answers from their personal assistants. In the second scenario, the personal assistant of each member specifies his availability then the negotiation agent decides the adequate time for all the team members.

Our personal agent is also able to collaborate with other agents of the V-team project; especially the SIP agent that is responsible for managing a group conference.

However, our personal assistant can be enhanced to overcome the following shortcomings: The interaction between the personal assistant and external agents was handled case by case. We have not studied a general interaction protocol between the personal assistant and the external agents. Also we have not specified how external agents present themselves to the personal assistant.

The user interface agent could be also enhanced by including the possibility of transforming voice information to ACL messages. Also we focus in our project on how to design a general model of personal assistant rather than providing enhanced functionality for agenda, address book and e-mail address management. Other existing famous software may offer better features. Consequently, the integration of those software using wrapper technologies described in the previous paragraph will be very useful

3.8 Summary

The purpose of this chapter was to describe a general model of Personal Assistants. It stated that the main objective of our work is to provide a generic model of personal assistant using agent technologies. Some of the essential requirements of the system presented are extendibility, openness, and flexibility to changes.

One example that allowed us to test our personal assistant was the V-team project that manages virtual team collaboration. The role of our personal assistant is to assist the user on managing the team creation, meeting scheduling, and the user's agenda. Moreover, the Personal Assistant was also responsible for acting on behalf of the user and taking some decision instead of him and collaborating with external agents.

The last section has described some issues related to our personal assistant. Essentially, we have proposed a general solution that will allow us to integrate existing software using wrapper technologies.

The next chapter will show how different components of the system were implemented. It will also give examples of some scenarios of the application.

CHAPTER 4: IMPLEMENTATION

4.1 Introduction

This section describes the implementation of various components that constitute the Personal Assistant. The project was developed mainly on Windows NT, PC-based machine. Agent management was performed using FIPA-OS platform. We used also JAVA for the development of different system components in addition to other tools such as XML.

The development was principally oriented to satisfy the requirement of the V-Team collaboration project. Most of the scenarios presented were in collaboration with agents of the V-Team collaboration project. However, in our design and implementation we provided a generic and extendible model.

We will start by explaining our choice of the programming tools. Then we will describe the implementation of different agents systems and the communication mechanism. Finally we will present some prototype screens.

4.2 Implementation Choices

4.2.1 FIPA-OS Platform

FIPA-OS (FIPA Open Source) is an open agent platform originating from Nortel Networks. It supports agent communication language, which conforms to the standard. FIPA-OS supports the majority of FIPA experimental specifications and is being continuously improved as a managed Open Source project, making it an ideal choice for any FIPA compliant agent

development activity[23].

FIPA-OS components include optional and mandatory components. Examples of optional components are the Parser Factory, Jess Agent Shell, and Choice Constraint Language CCL. FIPA-OS components include switch-able implementation components that include the Message Transport Protocol (RMI, IIOP...), Database (Memory Database, Serialization Database), and Parser (SL, ACL, XML, RDF) [23].

The core component of FIPA-OS are illustrated by the following figure:

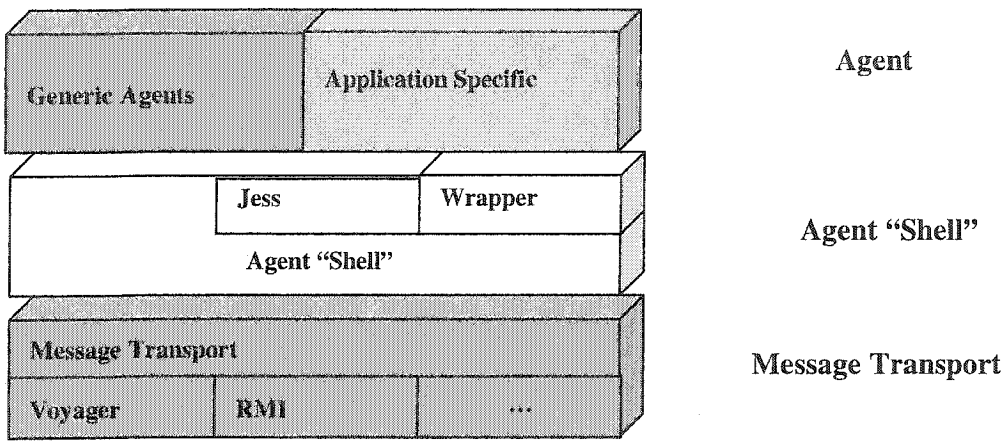

FIGURE 4.1: FIPA-OS CORE COMPONENTS

The FIPA-OS implementation includes support for:

- An agent Shell for producing agent, which then can communicate with each other using FIPA-OS facilities.
- Multi-layered support for agent communication.
- Message and conversation management.

- Dynamic platform configuration to support multiple Internal Platform Message Transport IPMTs, multiple type of persistence.
- Abstract interfaces and software design pattern.
- Diagnostics and visualisation tools.

To support agent interactions, FIPA has defined an Agent Communication Language (ACL) in order to provide a universal message-oriented communication language. Based on speech-act theory, FIPA ACL defines twenty per-formatives [16].

A key focus of the platform is that it is supports openness. Indeed, FIPA-OS is open in that many parts of it can be replaced or omitted and multiple types of parts can be used. FIPA-OS is also an open source. It offers the user the source code in addition to the executable code. The licensing allows the user to modify and distribute it. Users can build their own executable code using standard build tools such as compilers.

Moreover, FIPA-OS defines a standard base protocol based on speech acts with several ontologies and no specific service encoding is necessary while other agent platforms require a specific service ontology, encoding, and protocol combination.

However, FIPA-OS presents two major drawbacks: security and mobility. Although specifications pertaining to security within the context of FIPA specifications were started in 1998, there is no coherent completed picture for agent security within FIPA-OS.

On the other hand, even if it has the potential to support mobile agents, mobility has been implemented as a prototype only in FIPA-OS

FIPA-OS is implemented in pure Java. Two alternative distributions are provided; one supporting JDK 1.1 which is recommended for agents intended for deployment on small

footprint devices and one supporting JDK 1.2 for all other uses. In our work we use the second version that uses JDK1.2.

4.2.2 Programming Language

Our choice of Java as the programming language comes directly from our choice of the agent platform FIPPA-OS since it is the only programming language supported by FIPA-OS.

Java is an object-oriented language that was one of the factors that have influenced the quick spread of agent technologies. It represents the language of choice for the implementation of different agent systems. Examples include FIPA-OS, Aglet, Concordia, Odyssey, Voyager, and Grasshopper.

Java offers a number of advantages that can be summarised as follows[22]:

Portability. Thanks to its "byte code"; Java code can be executed in any device that supports the Java Virtual Machine. The slogan "Write once, run anywhere" is often used to describe the portability of Java.

Widespread. Java is widely accepted through the world and has great support from the computer and telecommunication industries.

Distribution. Java classe libraries come with support for network communication. Hence, Java provides great easiness in network programming.

In our project we use the standard JDK 1.2 (Java Development Kit).

4.2.3 Why XML For Data Storage

There is a need for a storage system in our Personal Assistant in order to store the user information. This system should be simple and easy to use for the manipulation and the update

of the data. Also, because of the heterogeneity of the data and the multiplicity of the applications that the Personal Assistant supports the system should have some ability to support structured data such as in relational database management system model.

The eXtendible Markup Language XML [24] is defined as a universal format for structured documents and data in the web. It was primarily conceived to meet the requirements of large-scale web content providers and processing of web documents. It is also expected to find use in certain MetaData applications. Examples of those data include Address book, configuration parameters, financial transaction, and technical design.

XML is a set of rules, lines and, conventions that help to device a text format for such data. Its syntax uses matching start and end tags, as in HTML files to mark up information. The data tag is called an element; elements may be further enriched by attaching name-value pairs called attribute (for example, Price = "2$"). This simple format makes the documents easy to process by a machine and remains understandable by humans.

A software model called an XML processor is used to read XML documents and provide access to their content and structure. XML provides also a mechanism to impose constraints on the storage layout and logical structure.

We can find in the market many XML parsers that help to extract information from XML files. Examples include parsers from Microsoft, IBM, Oracle and Sun. They differ significantly in performance, reliability and conformity to standards.

In this project, we used the Apache XML parser from IBM (commonly known as Xerces parser) that was integrated with the FIPA-OS platform.

4.3 System Agents

4.3.1 Agent Objcet

The system agents are implemented using *FIPAOS.Agent* package. All the system agents extend from *FIPAOSAgent* that is responsible for loading an agent's profile, and initializing the other components of which the agent is composed. When the agent object is instantiated it creates three mandatory components in the following order: the MTS, (Message Transport Service), TM (Task Manager), and CM (Conversation Manager)[23].

The agent shell provides the following functionalities:

- Sending messages which is accomplished by the *forward()* method in either the *FIPAOSAgent* class or Task class.
- Retrieving the Agent's properties such as the profile and the state and locating the platform agent.
- Registration with the platform agent using *registerWithAMS()* and *registerWithDF()* as well as the call-back methods *registartionSucceded()*, *registrationFailed()*, and *registrationRefused()* which should be overridden.

The constructor takes three parameters:

- Platform_profile_location: it represents the location of the platform profile file (e.g. fipa-os\profiles\platform.profile).
- Agent_Name: It represents the location of the agent in the network. This parameter combined with the Agent_Name allows identifying an agent. E.g. if the IP address of the machine is 137.122.109.166 and the name of the agent is UIAgent then the parameter UIAgent@iiop:137.122.109.166 identifies the agent in the network.

The class diagram is as follows:

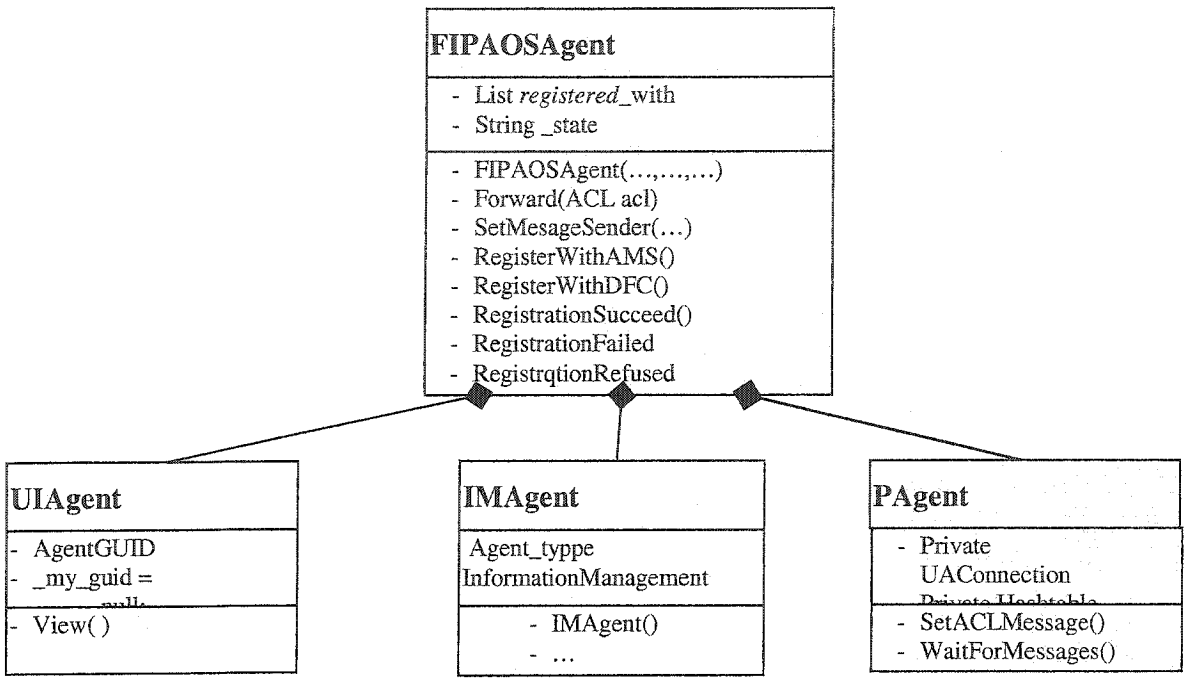

FIGURE 4.2: UML VIEW OF AGENT CLASS DIAGRAM

4.3.2 User Interface Agent

The user Interface Agent is the agent that interfaces the human user with the system. It receives information and direction from the human user, translates it into ACL messages and forwards it to the adequate system agent.

The *UIAgent* constructor uses the constructor of the super class FIPAOSAgent to create the User Interface Agent. It also initializes the *UserInterface* class, displays it and, waits for some human intervention to perform the required action.

The class *UITaskListener* that extends from FIPAOS 's Task class allows the agent to handle different messages coming from the Information Management Agent or the Personal Agent. Its principal methods are (1) *handleInform()* that gets information from different system agents

and displays it to the user, (2) *handleRequest()* that gets requests from the Personal Assistant to specifies some information, and (3) *handleOther()* that handles other messages.

The classes *Agenda*, *Attandency* and *MeetingScheduling* are used to handle requests related respectively to the applications Agenda, User Attandency, and Meeting Scheduling.

The principal interface is *UserInterface* that extends Java's Frame class. It allows the user to choose the area, the application, the conditions, and the rules step by step as described in Chapter 3.

The Iconic Modelling Tool component represents the Graphical User Interface that allows the user to construct the request using a set of icons. We used a small set of icons that represent each step and action performed. To support this, the *UserInterface* uses a number of supporting Java classes. *ImageLabel* is the class that allows image components to be dragged and dropped onto other components. Icon, extending *ImageLabe*l, lets images be used as icons and implements mouse clicks upon them. *MydrawingPanel* is a specialization of Java's Panel class that allows Icon instance to be added.

The most important methods are: (1) *waitForButtonPressed()* that allows to forward the message to the adequate agent , (2) *getACLMessage()* that encapsulates the information into ACL messages containing the sender, the receiver, and message type, *and (3) action()* that specifies the interface to be loaded or the action to be performed upon each event triggered by the human user. Other methods allow drawing the icons that model each step and action performed by the user. The Graphical User Interface was implemented using the Java Swing Package.

Some information specified by the human user should be translated into a format that can be stored in XML file. The class *ToXML* does this.

Other utilities allow the user to show textual information by pressing the *Bubble Button*, to cancel an action and return to the previous stage by pressing the *Undo Button* and, to clear the drawing panel by pressing *Clear Button*.

Once the user finishes specifying the task he can send the information to the adequate agent by pressing *Send Button*.

The following schema illustrates the class diagram of the User Interface Agent:

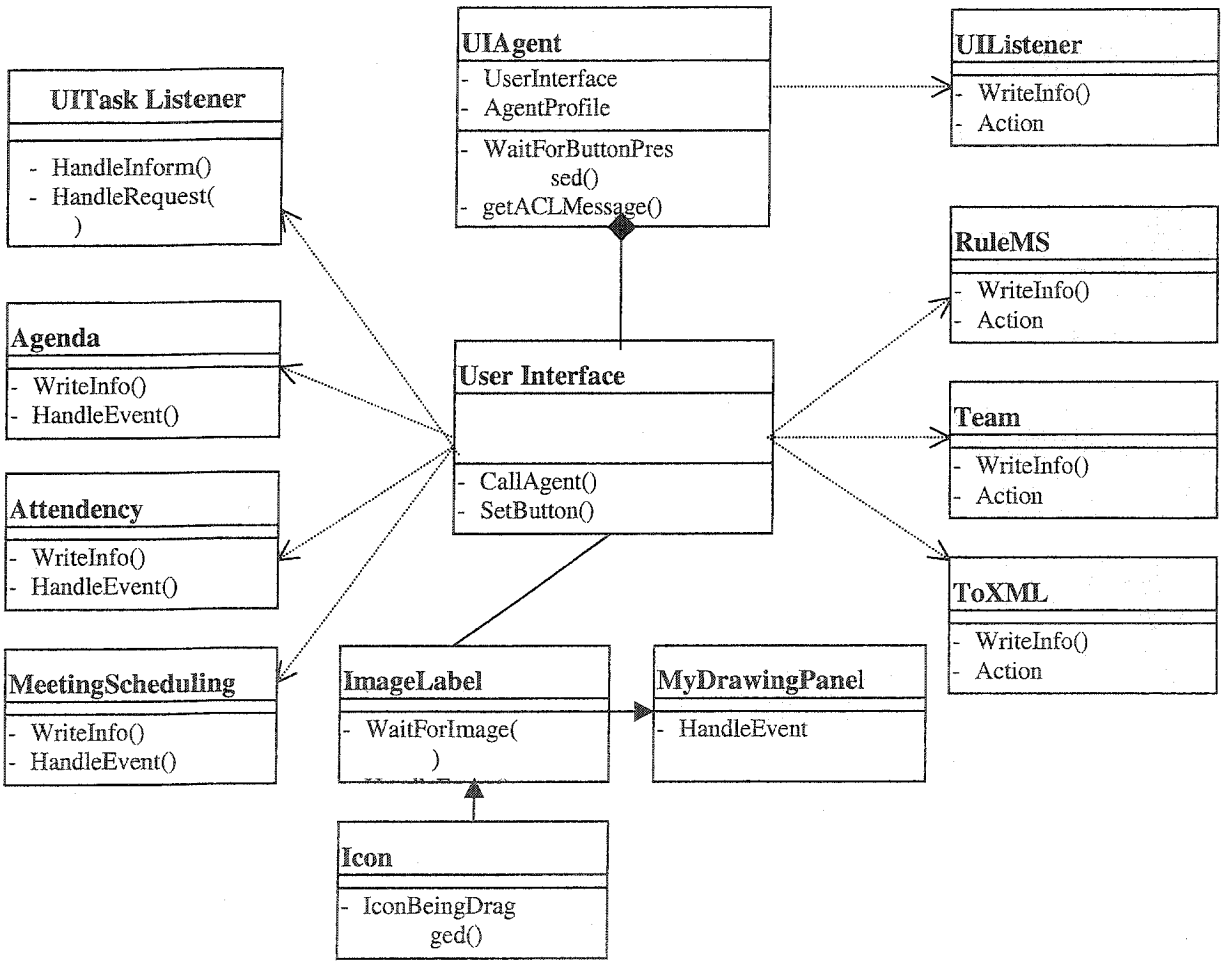

FIGURE 4.3: UML VIEW OF USER INTERFACE AGENT CLASS DIAGRAM

4.3.3 Information Management Agent

The Information Management Agent receives information from the User Interface Agent and stores it into XML files. It also receives requests from the Personal Agent and replies to him. It

acts also as the policeman that controls the system and surveys its interactions with external entities. To do this, it should be able to parse the XML files and extract information from it and check whether the users constraints are satisfied or not and determine actions that should be taken upon some new events.

The principal class is *IMAgent* that extends from FIPAOS's FIPOSAgent class. The instantiation of this class sets the *TaskListener* to IMTaskListener class. This class is responsible for handling different messages that it gets from the system's agent. Its principal methods are (1) *handleInform()* that gets information from the User Interface Agent and stores it into XML files, (2) *handleRequest()* that gets requests from the Personal Assistant to get some information, and (3) *handleOther()* that handles other messages.

The class *InformationManagment* is responsible for writing and reading from XML files. It also returns the results of the request to the *IMAgent.*

The class *ParseXML* is responsible for parsing the XML files and returning the results to the class *InformationManagment.* To do this we use the XML parser that constitutes one of the features offered by FIPA-OS. The package used is *fipaos.parser*.

The class *ParseMessage* is responsible for parsing messages coming from the Personal Assistant and storing it in a container as parameter and value couple. The class *ParseXML* uses this table to determine whether the parameters given by the Personal Assistant satisfy the requirements of the human user or not.

One example of those condition is as follows: the personal assistant gets the parameter of a meeting (E.g. Subject = 'V-team project' // Start = 2 PM // Duration = 2h //). The Personal Assistant sends the parameter to Information Management Agent that passes the tree to determine whether the user is available or not. The response can be yes, no, or unknown.

The class daemon is responsible for checking new events in the system and notifying the user. It consists of a process that is continuously running in the machine to check for new events.

Figure 4.4 shows the class diagram of the Information Management Agent:

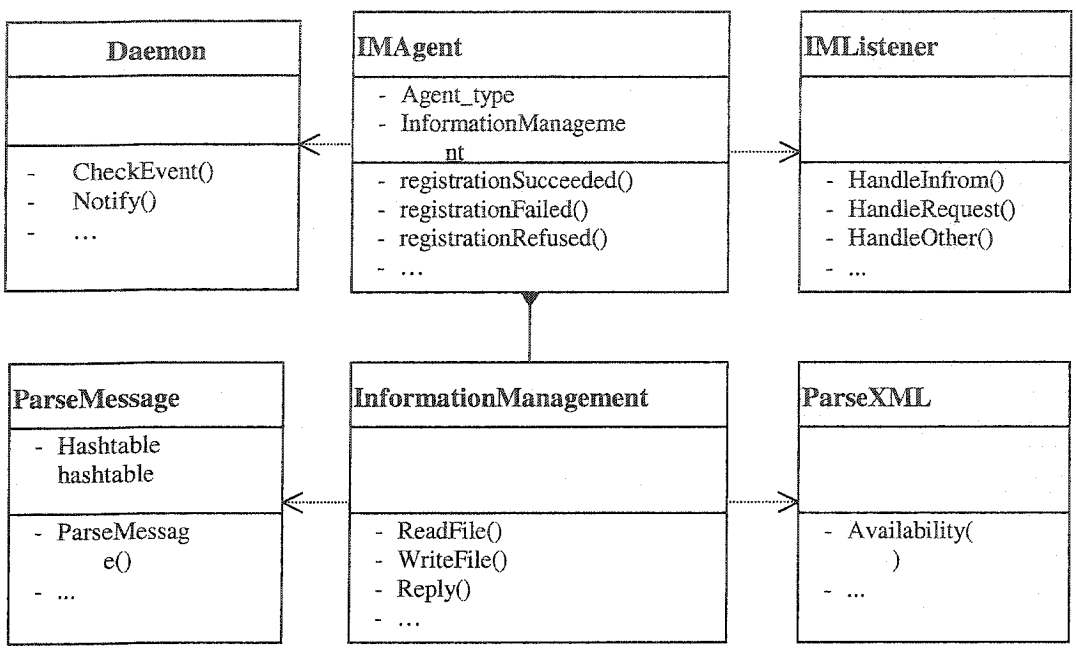

FIGURE 4.4: UML VIEW OF INFORMATION MANAGEMENT AGENT CLASS DIAGRAM

4.3.4 Personal Agent

The Personal Assistant is mainly responsible for the communication between the system's agents and other external agents. It gets requests from external agent and transfers them to the Information Management Agent that determines whether the request can be satisfied or not. It also gets tasks from the Information Management Agent or the User Interface Agent and transfers them to the adequate agent.

The principal class is *PAgent* that extends from FIPAOS's FIPOSAgent class. The instantiation of this class set the *TaskListener* to *PATaskListener* class. This class is responsible for handling different messages that it gets from the system agents and external agents. Its principal

methods are (1) *handleInform()* that gets tasks from the User Interface Agent and translates it to the adequate agent, (2) *handleRequest()* that gets requests from external agents and transmit it to the Information Management Agent.

The personal agent has also an XML file where it stores information about external agents and services provided by each one. It has to parse this XML file to determine which agent can provide the service asked by the human user.

Figure 4.5 shows the class diagram of the Personal Agent.

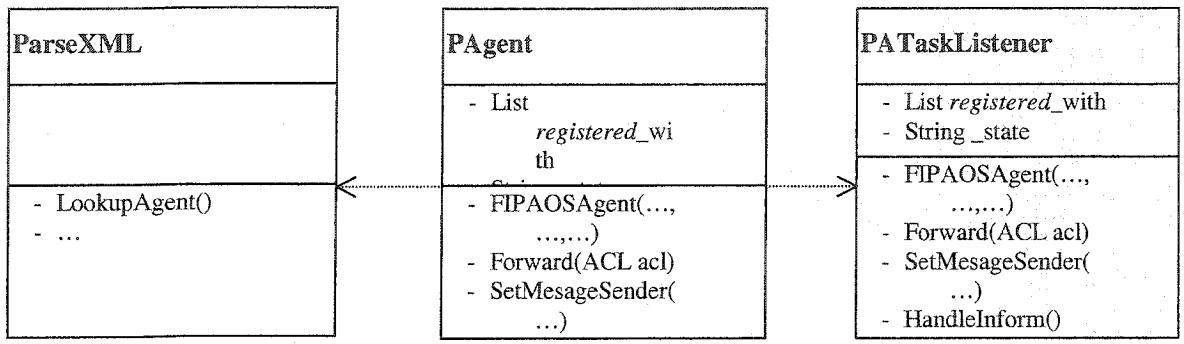

FIGURE 4.5: UML VIEW OF PERSONAL AGENT CLASS DIAGRAM

4.3.5 Agents Communication

Agent Communication Language ACL offered by FIPA-OS enables the agents to communicate. The package used is *FIPAOS.parser.acl*. The principals methods offered to handle *ACLMessages* are: (1) *getContent()* that gets the content of the message; (2) *getReceiver()* that gets the receiver of the message; (3) *getSender()* that gets the sender of the message; (4) setContent() that sets the Content of the message; (5) *setReceiver()* that sets the receiver of the message; (6) *setSender()* that sets the sender of the message. The agent sends

messages using the method *forward()* and gets new messages using *conversation.getMessage()*.

The following is the main structural elements of an ACL message [23].

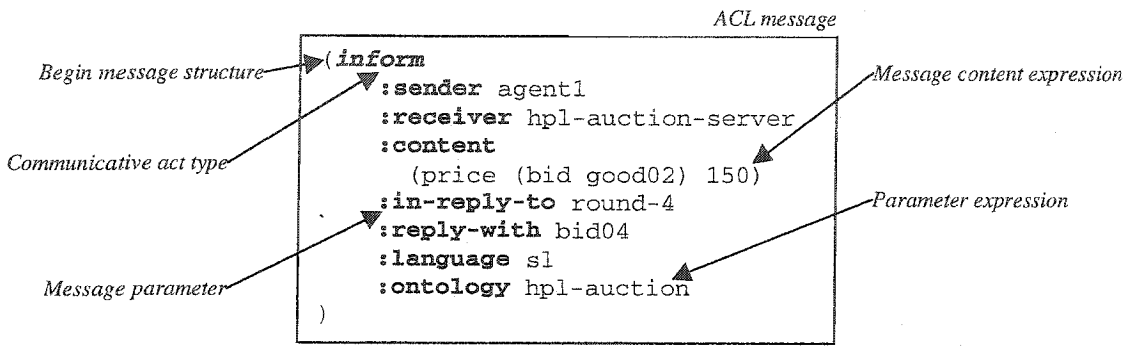

FIGURE 4.6: OVERVIEW OF ACL MESSAGE STRUCTURE

The following is a description of different elements in an ACL message [16]:

Element	**Description**
Per-formative	Denotes the type of the communicative act of the ACL message
Sender	Denotes the identity of the sender of the message, i.e. the name of the agent of the communicative act.
Receiver	Denotes the identity of the intended recipients of the message
Reply-to	Indicates that subsequent messages in this conversation thread are to be directed to the agent named in the reply-to element, instead of being directed to the agent named in the sender element
Content	Denotes the content of the message; equivalently denotes the object of the action

Language	Denotes the language in which the content element is expressed
Encoding	Denotes the specific encoding of the content language expression
Ontology	Denotes the ontology (s) used to give a meaning to the symbols in the content expression
Protocol	Denotes the interaction protocol that the sending agent is employing with this ACL message
Conversation-id	Introduces an expression (a conversation identifier) which is used to identify the ongoing sequence of communicative acts that together form a conversation
Reply-with	Introduces an expression that will be used by the responding agent to identify this message
In-reply-to	Denotes an expression that refers to an earlier action to which this message is a reply
Reply-by	Denotes a time and/or date expression which indicates the latest time by which the sending agent would like to have received a reply

The most important elements that we used in our messages are the per-formative, the sender, the receiver and the content.

We use the language "simple" because it is the language by default that is understood by all the agents. Indeed, the list of external agents that may contact the personal assistant are not known in advance. Consequently, the use of another specific language (e.g Prolog) will limit the category of agents that will be able to communicate with our personal assistant.

We use the ontology "ping" to allow the PA to initiate the contact with another agent. An agent sends a "ping" request to another agent to check if it is alive or not.

4.4 Prototype

4.4.1 Overview

The main objective of this section is to present some prototypes of the application. The screens that we choose are those related to the scenario that we presented in Chapter 3 in the context of the Virtual-Team Collaboration Project. So, let us assume that a project manager wants to organise a virtual meeting. He transfers the task to his Personal Assistant, which contacts the Personal Assistant of every team member in order to agree on the adequate time. Every user should have already specified his agenda to his Personal Assistant, which will be stored by the Information Management Agent. When the Personal Assistant gets the invitation to attend a meeting it replies without any intervention from the human user. In the following we will present some prototypes of each component of our Personal Assistant according to this scenario.

4.4.2 User Interface Agent

The user can interact with his Personal Assistant via the User Interface Agent that offers him the adequate GUI. The following GUI screen illustrates one example of the model that we described previously using the Iconic Modelling Tool feature. This example shows how simply a request can be transmitted to the Personal Agent. It illustrates the two first steps that the user should perform in order to modify the agenda. First, the user chooses the Area (in this case Agenda), and then chooses the action he wants to perform (in this case modify the agenda).

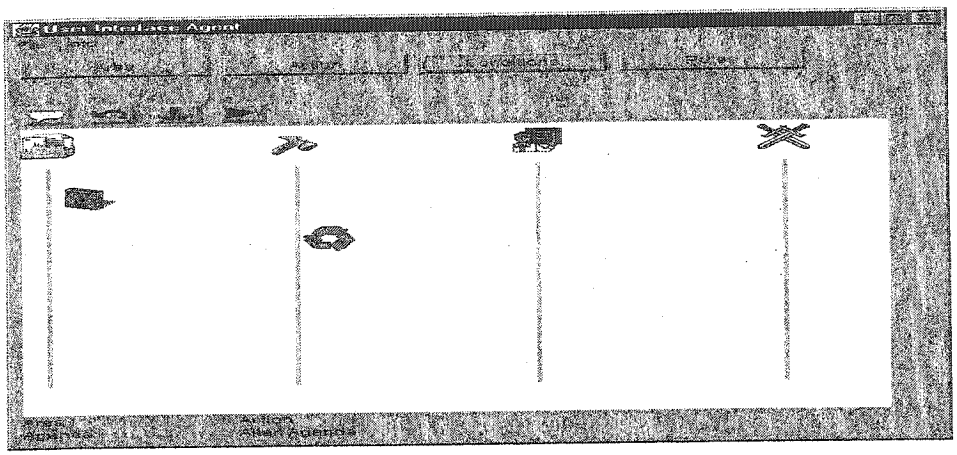

FIGURE 4.7: GUI SCREEN FOR AGENDA MODIFICATION

The second screen illustrates the attendance rules of the user. It offers the user the possibility to specify his conditions according to the type of the meeting, the subject, the location, and the period using brackets and 'Or & And' operators. The personal assistant checks also the syntax error of the condition before it validates it. When the user finishes specifying all the constraints he sends the information to the Information Management Agent.

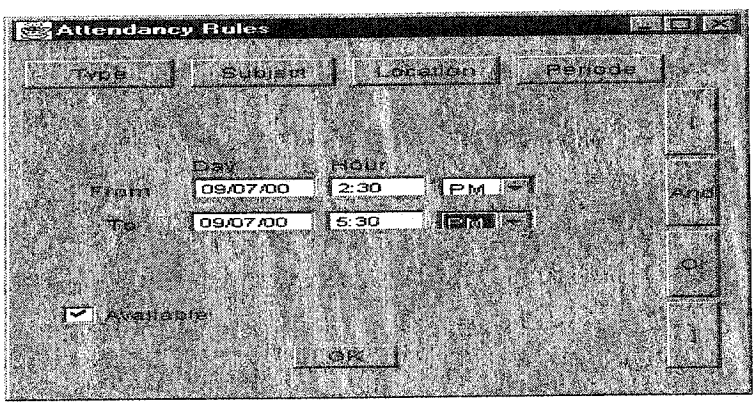

Figure 4.8: GUI Screen for Attendance Rules Specification

Other similar interfaces are implemented to help the user specify the constraints for different applications.

4.4.3 Information Management Agent

The Information Management Agent receives the message from the User Interface Agent, extracts this information and stores it in the XML file. One example of the stored information is the user agenda and the user availability. The constraints are represented as a tree where

each branch represents a condition and can generate into other branches. The following figure shows one example of the XML file.

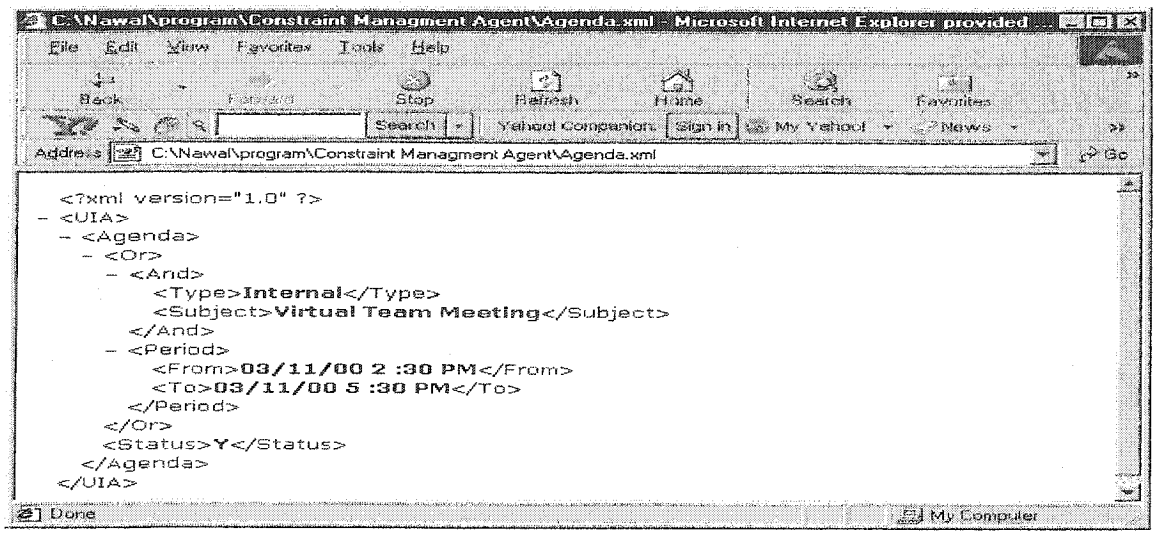

FIGURE 4.9: EXAMPLE OF XML FILES STORED BY THE IMA

4.4.4 Personal Assistant

The Personal Assistant keeps information about external agents, receives requests from them and replies to them when required. One example is when the user gets an invitation from an external agent to attend the meeting. The Personal Assistant sends the request to the Information Management Agent, gets the response from him and forwards it to the agent without any intervention from the human user.

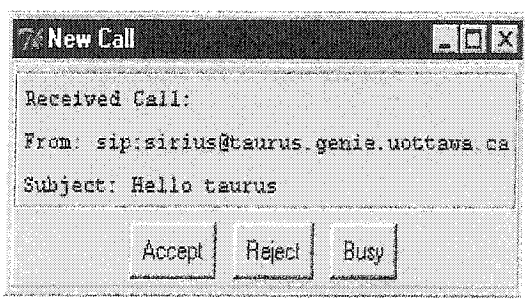

FIGURE 4.10: EXAMPLE OF REQUESTS HANDLED BY THE PERSONAL AGENT

4.5 Summary

In this chapter we discussed the implementation issues and solution for building a Personal Assistant model as applied and integrated with the Virtual-Team Collaboration project. The system was developed using Java platform and FIPA-OS for agent management and communication. XML files were used to store the system information. The chapter then explainined the implementation of different components of the system and the agent communication and ended by giving some prototypes of the application.

CHAPTER 5: PERSONAL ASSISTANT IN E-COMMERCE

5.1 Introduction

E-commerce is one of the most important areas where agent technologies can be applied. Agents help the user identify items he wants to buy or sell, search for the adequate buyer/seller, compare prices, negotiate with different entities, complete the payment, and survey the delivery if authorized by the human user.

There are already a lot of agents in E-commerce that assist the user in performing different tasks. They range from simple search engine to more sophisticated multi-agent system for buying good and services. Examples of those systems include Kasbah from MIT Media Laboratory and Magnet from the University of California. Our objective is to apply our model of personal assistant in the E-commerce domain. Indeed, we were determined since the first step of our work to provide a generic model of personal assistant rather than a specialized one for a single application.

This chapter presents our work in the use of our personal assistant model in the E-commerce domain. A brief overview of existing agents in the E-commerce will be given. We will explain the overall scenario that we have adopted and, we will end by presenting the general architecture and showing how our model of personal assistant presented in Chapter 3 can be applied to provide personal assistance in E-commerce. Finally, we will describe the negotiation protocol adopted by personal assistant.

5.2 Related Work

In the following we will describe two examples of the use of agents in E-commerce are close to our model of personal assistant.

5.2.1 Kasbah

MIT Media Lab's Kasbah is an online World Wide Web marketplace for buying and selling goods. A user creates an agent, provides it with a set of criteria, and dispatches it into the marketplace. The criteria include price, time constraints and quantity of merchandise desired. Users can also choose between three negotiation protocols. In the cool-headed protocol the agent rises the price slowly. The anxious protocol is adopted when the user is willing to buy an item quickly, so he rises its price rapidly. Finally, the greedy protocol is adopted when the user is not anxious to buy the item so he does nott rise the price until the end of the period. When a new Seller/Buyer agent arrives, the market notifies all the buyer/seller agents that a new agent is in the market. The agent then starts contacting other agents and looking for interesting deals. It also replies to different requests coming from different agents. The seller/buyer agent filters the available deals according to the user specifications, and then negotiates a deal on its owner's behalf. Once the agent makes a good deal it quits the market and reports to his owner. The market then notifies all the existing agents that the agent left [22].

5.2.2 Magnet

Magnet stands for Mobile Agent for Networked Electronic Trading. It represents one example of the use of Mobile agents in E-commerce. The main scenario is as follows: the buyer assembles a list of preferred suppliers for each component with the desired delivery schedule and a list of preferred suppliers for each component, then dispatch the mobile agent. The mobile agent then stops at the first site in his list and contacts the supplier to obtain quotations

and delivery schedules for the required component parts. If a quotation is attractive, it reserves the component before moving to the next site. If the supplier is not able to give him a quick answer the mobile agent creates a surrogate agent, endows him with the necessary information and then moves to the next step. In due course, the mobile agent returns to the buyer with the reservations that it made. The buyer then compares the proposition given by the agent with those of all the other surrogate agents in order to determine the best combination of offers. The user then sends messages or agents to the suppliers to commit or cancel the reservation. Magnet is fully implemented in Java. The agent mobility is enabled by the Java aglet[25].

5.3 State of the Art

The E-commerce application that we consider consists of the procurement of different goods and reselling them. Typically, in a retail market, the merchant looks for items that his clients asked for, and sells them with a higher price. His main tasks consist of:

- Looking for potential sellers/buyers in the market that offer competitive prices,
- Negotiating with different entities to get the best deal,
- Comparing different offers,
- Managing the database,
- Completing the transactions.

Even though those tasks seem simple and do not need much knowledge and reflection from the human user most of them are repetitive and time-consuming, which makes personal assistance a critical need. Moreover, the application includes many features such as user profiling, information filtering, competitive services, and decision aiding. E-commerce is an excellent

example to demonstrate the importance of the use of personal assistants in our increasingly networked word. The personal assistant will be responsible for assisting the human user in most of the steps of the buying/selling process. Namely, they are the product brokering, merchant brokering, negotiation, and if authorised, purchase/sell and deliver.

The nature of the buying/selling process needs a lot of information gathering and exchange to find the desired item with a good price and other criteria. Consequently, the whole process can consume a large bandwidth. Mobile agent technology can be very useful to deal with this problem. A mobile agent can be created to move between different sites to look for the best seller/buyer and go back to its original site in order to report to the personal assistant.

This approach presents the following advantages:

- Parallel execution: The system can execute multiple tasks simultaneously since many mobile agents can be dispatched to look for different items when the personal agent is still assisting the human agent in performing other related tasks.
- Disconnected operation: the personal assistant can continue performing its tasks even if the network connection is closed.
- Saving the bandwidth: This is the principal advantage of using mobile agent since the principle of a mobile agent consists in moving the computation to the data rather than the computation to the data

The proposed model is based on the architecture of the personal assistant described earlier as an agent that offers a complete service to his owner. Some of them constitute an integrated part of it, others are provided by external agents. Indeed, each site is composed of a personal assistant, which is able to communicate with different buyer/seller agents and which creates a

set of mobile agents and dispatches them throughout the network to look for goods to sell or to buy.

5.4 The Overall Scenario

Different scenarios that we adopt are described in the following. Indeed, we distinguish basically between two different scenarios according to how much the user is anxious to buy or to sell goods. Other related services will be explained in the next section.

1. The user indicates his general rules using the User Interface Agent that will be sent to the IMA. (E.g. Never deal with a specified company because of its bad record, Never process a request to buy specific items because they are not demanded anymore in the market),

2. The user identifies items that are usually requested from different clients. One way that allows him to do this, is from the requests of items that the Personal Agent gets from other buying agents and that are not available. Another way can be direct requests from clients by e-mail, phone, and fax,

3. The user also specifies the items that he wants to buy/sell in term of constraints, especially, the desired price and the highest/ lowest price he can accept,

4. In this step, there are two different scenarios:

- The user is anxious to buy or to sell a merchandise. So the task is transmitted to the IMA and then immediately to the Personal Agent. The PA creates a surrogate mobile agent, which will be responsible for moving between different sites in order to find the best deal. The PA gives him the description of the item and a list of potential

buyers/sellers. One way that allows the Personal Agent to collect the list of merchants is advertisement. Agents that are interested in buying or selling goods contact the PA and describe different items they have and their address. The Personal Agent keeps this information in an XML file in a form of a couple of items and its potential buyers or sellers. The Personal Agent also keeps the number of times that this item has been requested. This number can help the human user later identify the most requested items in the market. The PA then parses the file in order to extract potential buyers or sellers for the desired item. The mobile agent moves to the first site in his list and starts looking for the item that matches the description. If it finds some interesting deals it adds it to its list. Before leaving the site it reports to the Personal Agent and moves to the next site. When the surrogate agent finishes visiting all the sites in its list, it returns to the original site. Then, it reports to the PA all the deals it has found interesting and ends itself. The PA compares the different offers that the surrogate agent had found and contacts the IMA to validate the offer. Finally, it reports to the human user in order to complete the deal or to authorize it to complete it.

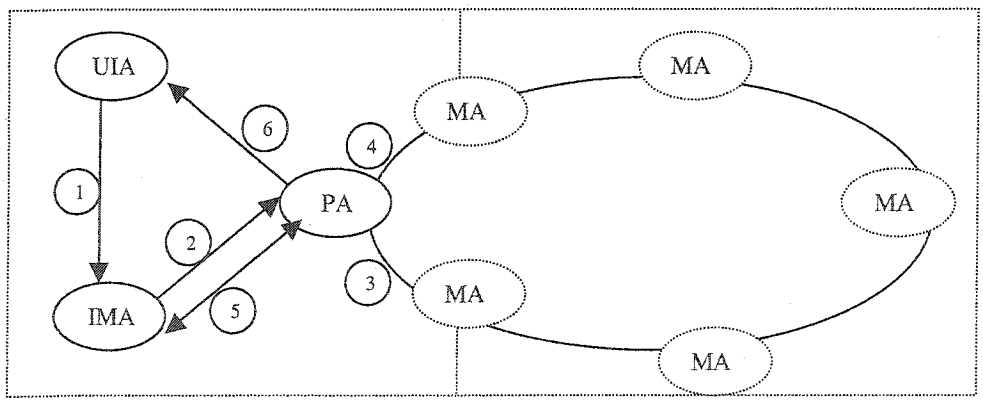

FIGURE 5.1: SCENARIO OF USING MOBILE AGENTS

- The user is not anxious. In general, he just wants to know the market value and the demand curve of different products in order to evaluate the competition in the market.

So, he sends the description of the item to the IMA and waits until externals buying or selling agents contact his Personal Assistant. When an agent contacts the PA, the PA sends a request to the IMA with the description of the item. If the description matches an item for sell or purchase the IMA replies to the PA telling it that the item is available with the desired price and the higher/lower acceptable price. The PA negotiates with the buying/selling agent. At the end, the information is sent to the human user via the UIA telling him that a potential buyer or seller is available with the description of the offer. If the item is requested and it is not available, the PA counts the number of time the item has been demanded and saves it in a local variable. If the number is big enough it sends a notification to the human user.

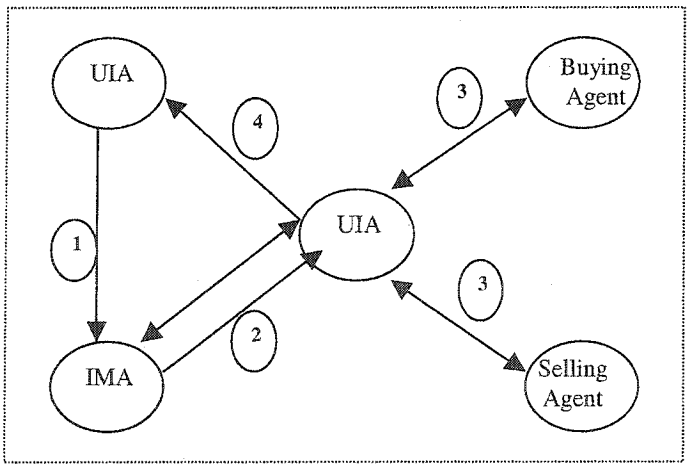

FIGURE 5.2: SCENARIO OF CONTACTING EXTERNAL BUYER/SELLER AGENTS

5. When the deal is completed between the human user and other merchants the human user sends a notification to the IMA informing him that a specific task is completed and that the item should be removed from the database.

The PA can use the services of other specialised agents to complete the deal if authorised by the human user. Examples of those services are payment, delivery, and service after sale.

Our model of personal assistant offers many advantages. First it helps the user to know the market demand and competition. Indeed, in the two examples of eCommerce agents described before the user may ask his personal assistant to buy/sell an item with a constraint on the price that can't be satisfied because the user doesn't have an idea about the prices offered in the market.

The existing eCommerce agents also do not protect the user from hostile buying/selling agents. In our model the user can specify some constraints and rules about different buyers/sellers that may contact him. The user who has a bad experience with a previous agent can inform his personal assistant that will keep this information. When the personal assistant is contacted by the hostile agent to buy/sell a good, the offer will be automatically refused without an intervention from the user.

Finally, our model of personal assistant offers the possibility to different buyers/sellers to advertise their offers. Existing eCommerce assistants contacts other agents only when the user has a specific need.

5.5 Architecture

The architecture of the system stems from the general model of the personal assistant described earlier in Chapter 3. The principal components of the personal assistant are the Personal Agent, the Information Management Agent and the User Interface Agent. Another component is the surrogate mobile agent, which is just a temporary agent. The Personal Agent creates it and it terminate itself when it finishes its task. The relationship between the PA and the mobile agents follows the classic master/slave relationship. The mobile agent reports to the Personal Agent step by step.

The following figure explains the interaction between the Personal Agent and different system agents with which it co-operates to achieve its tasks. Some examples of external services are also indicated.

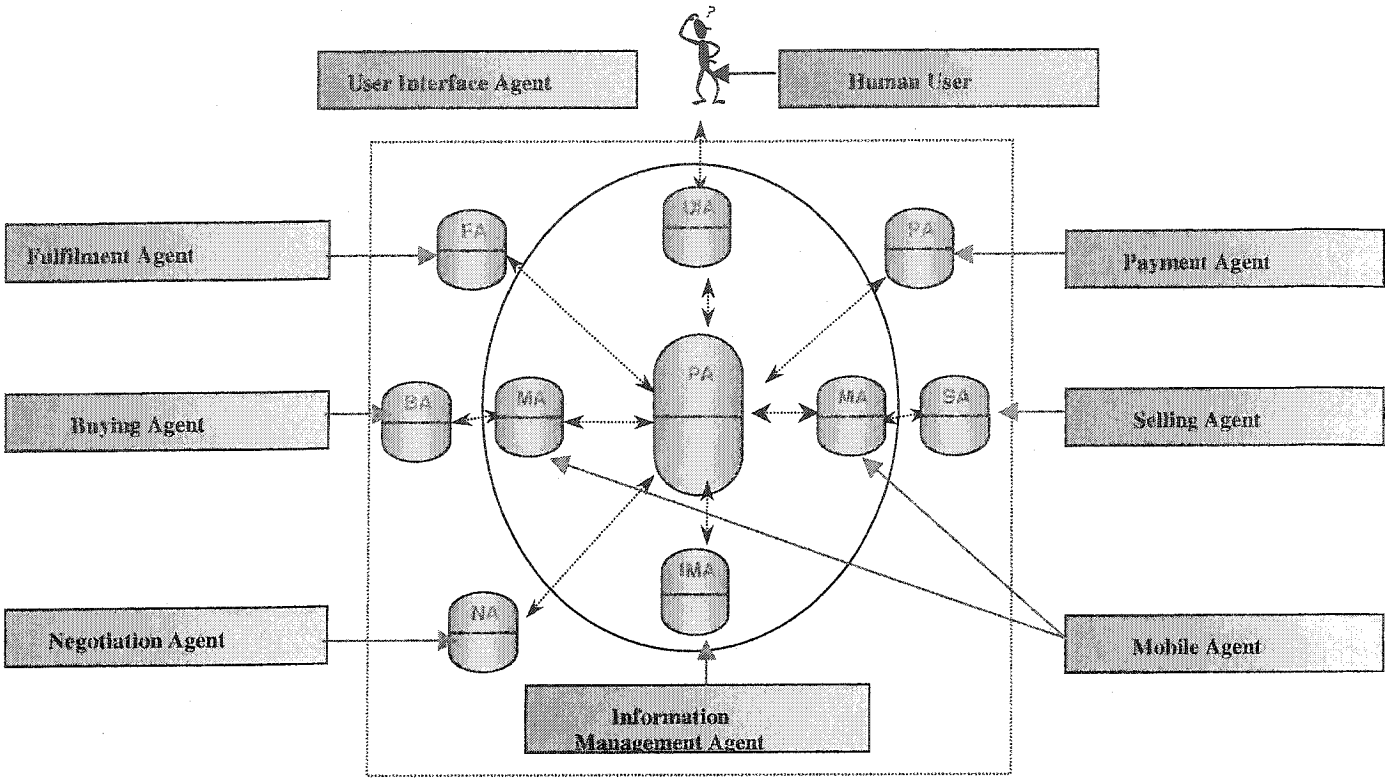

FIGURE 5.3: VIEW OF THE PA INTERACTIONS

5.5.1 Actors And Roles

5.5.1.1 Personal Agent

The Personal Agent is the central component. It is responsible for binding the system and the human user with the external world. It 's main tasks are as follows:

- Assisting the user to identify items that he has to buy or to sell,
- Submitting the order,
- Creating surrogate mobile agents, managing, and communicating with them,
- Comparing different deals and proposing the best one to the human user,

- Completing the deal with different entities when authorised by the human user,
- Communicating with the IMA to validate the deal and to extract the description and the requirement of each item,
- Reporting to the human user on what each surrogate mobile agent is doing in the market,
- Communicating and negotiating with external agents.

One simple negotiation protocol that the Personal Agent can use is the following. The human user specifies the desired price and the lowest/highest price he wants. When a buyer/seller agent contacts the Personal Agent, the PA gives its desired price. If the agent asks for a lower price it decreases the price by the margin that the human user have specified.

One example of messages exchanged between the Personal Agent and the UIA is:

```
( inform
        :sender PersonalAgent
        :receiver UserInterfaceAgent
        :Content (:State
                        (:AgentName MobileAgent1
                         :LastVisitedSite Adresse1
                         :Time Time1
                         :OfferedPrice Price1
                         : BestPrice Price0)
                        )
                                        )
```

5.5.1.2 User Interface Agent

The User Interface Agent is responsible for getting and transmitting the information from and to the human user. The UIA allows the human user to perform the following actions:

- Describe the item he wants to purchase or sell,

- View different agents he launches,
- View what every agent is doing. E.g. what was the last site it has visited and what is the deal it got,
- View a report on different agents that have contacted the Personal Agent to purchase or to sell an item,
- Update the database.

One example of messages exchanged between the UIA and the PA is:

```
( request
        :sender UserInterfaceAgent
        :receiver PersonalAgent
        ( content (:state
                        AgentName: MobileAgent1)
                        )
                                                )
```

5.5.1.3 Information Management Agent

As described before, the Information Management Agent is basically responsible for storing the system information, controlling the system and validating different actions performed by the system agents. Examples of its tasks are:

- Getting items description from the UIA and transferring the task to the PA if the human user is anxious to perform the task,
- Replying to the Personal Agent requests when it is contacted by a potential buyer or seller,
- Storing the rules and the product description that the human user specifies before launching a buying or selling operation,

- Validating the Personal Agent deals with other entities before reporting to the human user,

- Getting instructions from the human user to move, add and modify some products. This occurs, especially, when a new deal is completed or the user notices that the estimation of the price of a specific product was not correct and that it should be modified.

The IMA stores this information into XML files. The format that can be used to describe different operation is illustrated by the following example:

The human user wishes to buy a book of "Hermann Hesse" titled "Siddhartha" immediately. The price desired is 25 USD. Other XML files will be used to describe other features such as general rules.

```
<Task>
<op> Buying </op>
<type> immediate </type>
<Unit> 1 USD </Unit>
<c1> Type: Book </c1>
<c2> Author: Hermann Hesse </c2>
<c3> Title: Siddhartha </c3>
<c4> DesiredPrice: 25 USD </c4>
<c5> AcceptablePrice: 30 USD </c5>
</Task>
```

The message sent to the Personal Agent for creating an agent to buy the item is as follow:

```
( request
        :sender InformationManagementAgent
        :receiver PersonalAgent
        ( content ( (NewOrder(
                                :Action buy
                                :Type Book
                                :Author        Herman
                                Hesse
                                :Title Siddhartha
                                :DesiredPrice 25 USD
                                :AcceptablePrice    30
                                USD
                                :Unit 1 USD))
                                )
                                                )
```

5.5.1.4 Surrogate Mobile Agents

Other components of the system are surrogate mobile agents created by the Persona Agent to look for competitive deals in the market. They are temporary agents that terminate themselves once they finish their tasks. Reasons for creating surrogate mobile agents instead of having the Personal Agent moving itself in the network are:

- To have only a lightweight version of the personal assistant moving in the network in order to reduce the use of the bandwidth,
- To keep the personal assistant perform other tasks such as communicating with other external buyers/sellers and assisting the human user.

As in Master/Slave configuration the Personal Agent should have a complete control over the surrogate mobile agent. Indeed, the mobile agent reports to its master once it leaves a site or gets a deal.

Different actions they perform are summarised as follows:

- Get products description from the personal assistant,
- Move between different sites and negotiate with different buying/selling agents,
- Report to the Personal Agent before leaving each site,
- Terminate itself once it returns to its home and reports to the PA.

The message exchanged between the Mobile Agent and the Personal Agent is as follows:

```
( request
        :sender MobileAgent1
        :receiver PersonalAgent
        ( content (:Summary (
                                :Origin address1
                                :Destination address2
```

```
:Time time1
:OfferedPrice 24 USD
:BestPrice 23 USD))
)
                                    )
```

Agent mobility is enabled by FIPA-OS platform that was recently extended to support agent mobility.

5.5.2 Examples Of Services

5.5.2.1 External services

Some of the external services that the E-commerce PA may use to fulfil its task are :

- Payment Agent: Responsible for completing the payment step. This step consists in providing the credit number of the user. This agent obviously should have an enhanced security mechanism and uses a secure protocol to complete the electronic transaction such as SET (Secure Electronic Transaction),

- Fulfilment Agent: This agent is basically responsible for supervising the delivery and the service after sale stage.

- Negotiation Agent: This agent is responsible for negotiating with other buying or selling agents when some elements of the deal are not fixed. The negotiation parameters include the price, services after sale, and the delivery time. The negotiation can go from a simple exchange mechanism in retail markets to an enhanced negotiation protocols in other markets such as stocks, automobile, and fine arts,

- Buying or Selling Agent: Agents that contact the Personal Assistant to sell or to purchase some goods or services. They can be surrogate mobile agents created by the personal assistants of other merchants or organisation,

- Other agents: include agents that are specialised in providing specific services such as a personal travel assistant described in Chapter 2.

5.5.2.2 Internal services

Some examples of internal services include:

- Database Management: allow the user to manage his database such as adding some items for sale or purchase, deleting, and modifying.
- Address Book: contains the address of different merchants.
- E-mail, Calendar.

Examples of external software that can be integrated include Fax and Cyber cash.

5.5.3 Negotiation Protocol

The negotiation protocol that we adopted is simple and stems from the negotiation mechanism that we can find in the traditional retail market where buyers and sellers try to increase and decrease the prices until they reach an agreement. Indeed, it is based essentially on price comparisons. Other enhanced negotiation mechanisms that include other features such as the delivery options, services after sale and other added value services can be offered by using specialised external agents. In the following we explain the negotiation protocol followed by both the Personal Agent and surrogate mobile agents.

5.5.3.1 Between mobile agent and other buying/selling agents

The negotiation protocol between the mobile agent and different buying/selling agents is simple. The mobile agent gets the desired price and the higher/lower acceptable price from the

Persona Agent and starts moving in the network. When the mobile agent arrives at a site it asks the buyer/seller to propose a price for a specific item. The buyer/seller gives its suggestion. The mobile agent then compares it to the highest/lowest acceptable price given by the human user. If the offered price is lower/higher it keeps it in a persistent variable that represents the lowest/highest price obtained. Otherwise it asks the buyer/seller to give a better price. When the buyer/seller comes up with a lower/higher price it compares it again with the lowest/highest price it has and so on. If the buyer/seller refuses to lower the price the mobile agent moves to the next site. If the obtained price is lower/higher than the desired price the mobile agent sends a notification to the Personal Agent to complete the deal.

5.5.3.2 Between personal assistant and other buying/selling agents

The human user specifies the desired price, the highest/lowest acceptable price that indicates how high/low the agent can go and the unit by which the user should lower or increase the price. When the Personal Agent is contacted it proposes the highest/lowest price and then starts lowering or increasing the price depending on the demand of the buying/selling agent until it reach the highest/lowest acceptable price. When the buyer/seller accepts the offer the Personal Agent validates the offer with the Information Management Agent and sends a notification to the human user.

CHAPTER 6: CONCLUSION

6.1 Summary

The concept of a personal assistant helping his boss to achieve his routine tasks does already exist in our human world. However, with the advances in computer technology, the principle of a personal assistant assisting the user to monitor complex functionality, performing time-consuming research, learning advanced applications, and using more interactive interfaces, starts its infiltration in the software world. Indeed, there is a large effort from international organizations to standardise the personal assistant concept and to define a universal model that would allow more open communication and more interoperability.

There are already many designs and implementations of personal assistants from different researchers on artificial intelligence and software mainstream. However, most of the existing personal assistants are not collaborative and do not act on behalf of the user. Their main tasks are to assist the user in particular applications. On the other hand, few of them use or take advantage from the use of agents' technologies to build an open model of personal assistants; which will be able to communicate with other agents and profits from the services that they offer.

In our model, the personal assistant has many characteristics. First of all, in addition to assist the user to accomplish his daily task, it is able to act and take some decisions on his behalf; Furthermore it is collaborative. Which means that it is able to co-operate with other agents in order to offer a better service to the human user. Other characteristics include an enhanced user interface, openness, and extendibility.

In the context of this thesis, we presented an approach for building a model of personal assistants based on agent technology, and using the FIPA-OS platform. Our model is composed of three agents; namely the User Interface Agent which is responsible for translating the human language to the agent language, the Information Management Agent that is responsible for controlling the system and storing the system information, and finally, the Personal Agent which is responsible for directing tasks to the adequate agents, and also for communicating with the external world. We have also integrated and tested our model in the context of the V-Team Collaboration project. We showed, as well, how our model of a personal assistant can be applied in E-commerce.

6.2 Suggestion for future work

The future directions and the ongoing work that we suggest for this research may be summarised and focused in the following issues:

- Interoperability: The protocol of communication between the personal assistant and external agents was not deeply studied. In our implementation we were handling the communication scenario case by case. A more enhanced protocol of interaction can be developed; Especially how does external agents know about the personal assistant and how they offer and present their services to them.

- Trust: It deals with the degree or the extent to which the user can trust his personal assistant and allow him to take some decisions on his behalf. In our application, we assumed that the user trusts completely his assistant. We can imagine a scenario where the human user gives responsibilities to his personal assistant increasingly according to

his history and how much he succeeds to perform the required tasks and to which extent its decisions were correct.

- User-Agent interaction: Although we provided an enhanced visual Graphical User Interface via the Iconic Modelling Tool component, we believe that more enhanced interactions between the user and the personal assistant can be realised, as we think that the use of voice interaction modality will make the communication with the personal assistant easier.

REFERENCES

[1] M.N. Huhns and M.P. Singh, Eds., "Readings in agents". Morgan Kaufmann, San Francisco, CA, 1997.

[2] P. Maes, Ed., " Designing Autonomous Agents", MIT Press, Cambridge, MA, 1990.

[3] " The Agent Society"
Home page: "http://www.agent.org/".

[4] M. Wooldridge and N.R Jennings, " Intelligent Agents: Theory and Practice", in Knowledge Engineering Review, 10(2), pp 115-152, 1995.

[5] J. Dale, "A mobile agent architecture for distributed information", Ph.D. dissertation, University of Southampton, UK, September 1997.

[6] Y. Shoham, "Agent-Oriented Programming", in Artificial Intelligence, 60(1), pp. 51-129, 1993.

[7] H.S. Nwana, " Software Agents: An Overview", in Knowledge Engineering Review, 11(3), pp. 205-244, 1996.

[8] R. Tkito, " SHIPMAI, Secure and High Performance Mobile Agent Infrastructure", M.S. thesis, University of Ottawa, April 2000.

[9] E.H. Durfee, V.R. Lesser and D.D. Corkill "Trends in Cooperative Distributed Problem Solving." in IEEE Transactions on Knowledge and Data Engineering, 1(1), pp 63-83, March 1989.

[10] Katia P. Sycara, " Multi Agent Systems", in AI Magazine 19 (2), pp 79-92, 1998.

[11] " UMBC CSEE"
Home page: " http: //www.cs.umbc.edu/"

[12] C.G. Harison, D..M. Chess and A. Kersherbaum, " Mobile agents, are they a good idea?", Technical Report, IBM Research Division, T.J. Watson Research Center, Yorktown Heights, New York, March 1995.

[13] P. Wayner, "Free Agents", in Byte, 20(30), pp. 105-14, March 1995.

[14] Y. Lashkari, M. Metral and P. Maes, " Collaborative interface Agents", in Proc. Of the twelfth National Conference on Artificial Intelligence, vol. 1, AAAI Press, Seattle, WA, August 1994.

[15] P. Maes, "Agents that Reduce Work and Information Overload", in Communications of the ACM, 37(7), pp. 31-40, 1994.

[16] " FIPA 97 Specification", Geneva, Switzerland October 1997.
Home page: "http:// www.fipa.org/".

[17] N. Chauvat, "Narval, the Intelligent Personal Assistant", in Lunix Gazette, issue 59, November 2000.

[18] P. Klark and U. Manber, "Developing a Personal Internet Assistant", in Proc. Of ED-Media 95, World conf. On Multimedia and Hypermedia, Graz, Austria, 1995, pp. 372-377.

[19] B. Falchuk, " A platform for mobile agent-based data Access, Retrieval, and Interaction", Ph.D. dissertation, University of Ottawa, ON, Canada, 1997.

[20] A. Karmouch and H. Harroud, "Toward a collaborative virtual team", Internal Report, Multimedia Information Research Laboratory (MIRL), University of Ottawa, Canada, 2001.

[21] A. B. Johnston, "SIP, Understanding the Session Initiation Protocol", Artech House, ISBN 1580531, January 2001.

[22] T. Lindholm and F. Yellin, "The Java Virtual Machine Specification (2nd Ed) (Java Series)". Addison-Wesley, ISBN 0201432943, April 1999.

[23] " FIPA-OS Platform Tutorials"
Home page: "http://www.fipa-os.sourceforge.net/tutorials.htm".

[24] E.R. Harold, "XML Bible", IDG Books Worldwide, ISBN 0764532367, July 1999.

[25] A. Chavez and P. Maes," Kasbah: An agent marketplace for buying and selling goods", in Proc. Of the First International conference on Intelligent Agent and Multi- Agent Technology, London, UK, April 1996, pp 75-90.

[26] P. Dasgupta, N. Narasinhan, L.E. Moser and P.M Mellian-Smith, " Magnet: Mobile Agent for Networked Electronic Trading", University of California, 1999.

[27] ."Grasshopper - The agent Platfor;"
http://www.grasshopper.de/