Dissertation

Master's Degree in Mobile Computing

Mobile Agents Applied to Smart Homes:

The Virtual Butler Case Study

Rui Branco

Leiria, March *2015*

Dissertation

Master's Degree in Mobile Computing

Mobile Agents Applied to Smart Homes:

The Virtual Butler Case Study

Rui Branco

Master's project performed under the guidance of Dr. Nuno Costa Professor at the School of Technology and Management of the Polytechnic Institute of Leiria

Leiria, March *2015*

To My Family

Acknowledgments

This project would not have been possible without the cooperation and goodwill of those that now apply. To all, my sincere thanks.

To Professor Nuno Costa advisor for this project that gave me all the necessary support and advise to improve my approach to the topic.

To Eng. Paulo Ferro co-worker who with his knowledge helped me to have a broader and realistic view on mobile agents.

Finally a note of thanks to my group of friends and co-workers who have always motivated and encouraged me to conclude this project and, particularly, to Carolina Fernandes that even not being from the area of computer science, was always available to listen to my doubts, help and motivate me.

Abstract

The concept of computer networks exists from a long time and has evolved until it became the Internet, a massive set of networks that allows everyone to communicate from all around the world, and access all kinds of information in a very short period of time. With this rapid growth of the Internet, technologies have also evolved, in such a way that a computer, something that previously was big and usually placed on a desk, is now so small that we can put it in our pockets, such as smartphones, tablets or simple smart wrist watches.

With the use of these devices it is now possible to communicate and control a large set of appliances in a house, car, office and other places. With the communication of these small devices and appliances, the number of systems that automate and assist in the daily tasks has grown like opening doors, and windows, turning on the lights and music, checking the groceries list, checking appointments book and a lot of other tasks. Last year, these systems have focused not only in the general public but in specific niches of the population like blind people, old people and people with some kind of disability, acting like some kind of virtual butlers in their daily lives.

In this work it was implemented a virtual butler based on mobile agents capable of speech recognition and speech synthesizing, by using a local network in the controlled environment of a house, as a case study to evaluate to what extent the mobile agents can be used to help people in their daily tasks. The work developed shows that it is possible to integrate mobile agents with other software and manage the addition and removal of features as needed, and provide a good interaction between the user and the agent.

In short, mobile agents concept works and could be very useful when used to solve specific problems, and with the actual knowledge and infrastructures, with some work this concept

could be reignited and great and useful solutions accomplished mainly in smart environments.

Key-words: mobile agents,network, speech recognition, speech synthesizing

Resumo

O conceito de redes de computadores existe desde há muito tempo e evoluiu até se tornar a Internet, um conjunto enorme de redes que permite que todos se comuniquem em todo o mundo, e aceder a todos os tipos de informações num período de tempo muito pequeno. Com este rápido crescimento da Internet, as tecnologias evoluíram muito, a tal ponto que um computador, algo que anteriormente era grande, e geralmente colocado sobre uma secretária, agora é tão pequeno que pode ser colocado nos nossos bolsos, assim como smartphones, tablets ou simples relógios de pulso inteligente.

Com o uso desses dispositivos é agora possível comunicar e controlar um grande conjunto de aparelhos em casa, carro, escritório e afins. Com a comunicação desses pequenos dispositivos e aparelhos, tem crescido o número de sistemas que automatizam e ajudam nas tarefas do dia-a-dia, como abrir portas e janelas, acender luzes e ligar a música, verificar a lista de compras, verificar a agenda e muitas outras tarefas. No ano passado, esses sistemas focaram-se não só no público em geral, mas em nichos específicos da população como cegos, idosos e pessoas com algum tipo de deficiência, atuando como uma espécie de mordomos virtuais nas suas vidas do dia-a-dia.

Neste trabalho foi implementado um mordomo virtual baseado em agentes móveis com a capacidade de reconhecimento e sintetização de voz usando uma rede local no ambiente controlado de uma casa, como caso de estudo para avaliar até que ponto os agentes móveis podem ser usados para ajudar as pessoas nas tarefas do dia-a-dia. O trabalho desenvolvido mostra que é possível integrar agentes móveis com outros software e gerir a adição e remoção de recursos, conforme necessário, a fim de criar uma boa interação entre o utilizador e o agente.

Em suma, o conceito de agentes móveis funciona e pode ser muito útil quando aplicado na

resolução de problemas concretos, e com o conhecimento e as infraestruturas existentes, com algum trabalho, este conceito poderia ser relançado e realizadas grandes soluções principalmente em ambientes inteligentes.

Palavras-chave:agentes móveis, rede, reconhecimento de voz, sintetização de voz

Index of Figures

Index of Tables

List of Acronyms

ACL – Agent Communication Language

AI – Artificial Intelligence

AMS – Agent Management System

API – Application Program Interface

FIPA – Foundation for Intelligent Physical Agents

IDE – Integrated Development Environment

HW– Hardware

IP – Internet Protocol

JDK – Java Development Kit

JVM – Java Virtual Machine

LAN – Local Area Network

MA – Mobile Agent

MAP – Mobile Agent Platform

MASIF – Mobile Agent System Interoperability Facility

SW – Software

TCP – Transmission Control Protocol

WORA – Write Once Run Anywhere

XML – eXtensible Markup Language

Index

Introduction

The concept of computer networks exists from a long time and has evolved until it became the Internet, a massive set of networks that allows everyone to communicate from all around the world, and access all kinds of information in a very short period of time. With this rapid growth of the Internet, technologies that also evolved, in such a way that a computer, something that previously was big and usually placed on a desk, is now so small that we can put it in our pockets, as smartphones, tablets or simple smart wrist watches.

As these technologies grow, the companies grow too and are becoming more technological. In the same way, people are using these technologies more and more to the point that nowadays almost everyone has a smartphone or a tablet that frequently use in their daily routine. With the use of these devices it is now possible to communicate and control a large set of appliances in the house, car, office and other places. With the communication capabilities of these small devices and appliances, the number of systems that automate and assist in the daily tasks like opening doors, and windows, turning on the lights and music, checking the groceries list, checking appointments book and a lot of other tasks has grown. Last year, these systems have focused not only in the general public but in specific niches of the population like blind people, old people and people with some kind of disability, acting like some kind of virtual butlers in their daily lives.

It is in this area that the several software approaches differ on the architecture used, being the most used probably the client-server model. However, in this dissertation another approach that uses mobile agents is studied.

1.1 Mobile Agents

Along the years lot of approaches to communicate among computers, search and transfer information through the networks were developed like peer-to-peer and client-server, being the client-server model the most used.

The client-server model is a software architecture used to support a large number of services in the context of distributed systems [14]. In this model, the server is a host that is responsible for providing a set of running services that shares their resources with the clients. These clients are usually other machines that send requests to the server and wait for their responses. The servers are machines with the data and applications that receive the requests sent by the clients, execute those requests, and send back the responses.

Mobile agents are small programs that can migrate through the network from one computer to another, taking with it the code, data and execution context. Using mobile agents the user that owns the agent does not have to stay glued to the computer to get the job done. The agent will do it faster and probably better.

To understand the mobile agent concept deeply, nothing better than reviewing a little more about their history. How they have appeared, when, and of course, what mobile agents are and how they work.

The term “agent” is largely used in a wide range of subjects and areas of knowledge, like Sociology, Psychology, Biology, and of course Computer Science [2]. In this last subject, the term was first introduced in mid-70s, in the area of artificial intelligence [1]. That has happened because, since artificial intelligence tries to create something that can mimic the human intelligence, it often uses metaphors of psychology and sociology to inspire and motivate the work. [2]

A Software Agent is a software entity that continuously performs tasks given by a user within a particular and restricted environment. A software agent can be a simple program, a software component or a simple object. However, a true software agent must be interpreted in a more general concept in which a software object is passive and agents are active. [1]

The definition of what constitutes a software agent has been debated for years by the research community and despite the fact that there is no consensus there is a common understanding of the features a software entity must exhibit to qualify as an agent. [1]

- Autonomy – agents receive their tasks by the user and according to it, they behave and operate without asking the user for confirmations of each step to complete the task or all task in general.

- Social behaviour – agents have the capacity to communicate with each other or even with the user using an agent communication language. Their communications can be simple for pure exchanges of information or sophisticated for, example, negotiations using protocols.
- Reactivity – with the interaction with some kind of sensors, the agents are able to react to some identified events.
- Proactivity –agents can react to the environment stimuli, but they can also take some initiatives in the process of completing a task. E.g. "*Automation and an agent are somewhat like the difference between a dog and a butler. If you send your dog to buy a copy of the New York Times every morning, it will come back with its mouth empty if the news stand happens to have run out of this specific newspaper one day. In contrast, the butler will probably take the initiative and buy a copy of the Washington Post, since he knows, that sometimes you read it instead.*" [1]

The idea of having code moving in the network from different hosts independent from the architecture was mentioned, possibly for the first time, by Rulifson in 1969. He and some of his colleagues create the Decode-Encode Language (DEL) that was published as RFC[1]. This language was created by the idea that, in the beginning of a session an interpretative program to communicate to the host would be downloaded, and it would be that same program (written in DEL) that would be able to control efficiently the communication with small bandwidth between the local and remote host. [1]

About ten years later at Linkoping University in Sweden, a group of researchers had the idea of developing a packet-oriented radio network called Softnet. In this network, whose packet was a script file, a program written in FORTH language in which each node that received the packet automatically executed it, allowing each user to be able to instruct the nodes to provide new services. [1]

The first experiments with mobile software were conducted at Xerox by Shoch and Hupp, when they were writing worms to search their network for idle processors. [1]

After the above explanation of how the term agent has appeared and what software agent is

[1] RFC (Request For Comment) is a standard for new or modified network protocols. When standards are proposed these standards are released for public comment so that they can be refined.

and how it works, all the necessary things to understand the precursors of mobile agents are set. These precursors were the first attempts of an idea, a concept that later evolved, originated the mobile agents and spread, creating the networks that are actually used in the whole world.

The remote evaluation concept was the first precursor. It appeared when the research community was facing the problem of providing client-specific interfaces to remote services across heterogeneous distributed systems, in contrast to offering a unique interface with multiple actions that suited the majority of the users. Joseph R Falcone wanted the clients to be able to create their own personalized interfaces using a new programming language NCL (Network Command Language). In this language, the client sends a NCL expression to a server, the server receives the expression and using a set of standard functions in the form of libraries, the result is sent back to the client (as an expression) which can start execution again. This is the primitive concept behind mobile code in both directions. [1]

The second step towards mobile agents was the change to the messaging concept where some kind of autonomy was added. This new concept refers to mobile objects that are often associated with Java RMI. The idea was to create active messages that could migrate to a remote host. All of these active messages were composed with data and some kind of code program that were executed in each hosts when the message arrived. In these messages the data were still the dominant part of the message. In opposition to this approach, the mobile agents typically migrate more than once, being this process initiated by the agent itself.

Mobile processes are the third precursor of mobile agents and these mobile processes gave the mobile agents the ability to capture the execution state of the processor on virtual machine that they are currently using. This idea was developed in the area of the distributed systems. In this concept, a process that is being executed on a computer can be moved to other computers to create a load balance of the distributed system.

All these precursors gave birth to this new idea of mobile agents. Mobile agents are small software entities that can travel through the network, in order to translate the user specification – the “what” – into a possible solution. They take care of how the solution is found (they do the browsing, comparing and negotiating). The mobile agent has the capacity to work in an asynchronous and autonomous mode, as long as the basic infrastructure they

need is available. One of the most important characteristic of the MA is that they will visit the network nodes directly, and aren't limited to downloaded information from a specific platform. Imagine them as a sophisticated search engine that works on demand and is able to go where the information is, adding flexibility to the search process. Agents are fast, smart if necessary, and can start their task from small platforms and simple interfaces. While migrating through the networks, the agent will use only the available computer power. [1]

To better understand what mobile agents are and how they work, it´s important to remember two definitions:

- Pervasive computing – it means that in distributed systems everything can become a node. Computers have become smaller and smaller and can be found not only in cars or houses but in small things like refrigerators, wrist watches and glasses. [1]
- Nomadic computing – it means that users move around places while working, logging into systems from different computer domains. In this concept, the users want to see nearly the same working environment and applications, and most importantly the same data. Moreover the nomadic users also demand a seamless integration of different devices, creating the possibility to change the working environment from a computer to a mobile device. [1]

Mobile agents are a special type of mobile code. Mobile Code is a technique that transfers the code from the computer that stores the code file to the computer system that will execute the code. (E.g. Java Applets) [1]. Mobile agents are programs that can migrate from a starting host to many other hosts in a network of heterogeneous computer systems and fulfil a task specified by its owner. They work autonomously and can communicate with host systems and with other agents. During the migration the agent, typically carries with itself (Figure 1): [1]

- Code – typically a script file with a classes and methods representing a computer program (Java, C, Perl, etc.).
- Data – all the variables of the agent that contain some kind of information (attributes of the object in OOP).
- State – information about the execution state of the agent.

Figure 1 - Mobile Agent representation.

A mobile agent in the context of distributed programming can be considered an additional design paradigm and a useful supplement of traditional techniques like client-server architecture. Considering the mobile agents as a new design paradigm for distributed systems and recalling their definition, it is possible to identify four characteristics:

- Mobile agents are typically used in large and heterogeneous networks (Figure 2) where no assumptions can be made regarding the security or reliability of the computers.
- The migration process of the mobile agent is always started by the agent. More precisely the programmer or the user.
- The migration process is used to access resources and information only available at other hosts in the network.
- Mobile agents have the ability to migrate more than once (ability also known as multi-hop), e.g. the mobile agent migrates to one first machine and might migrate again to another in order to continue their task. [1]

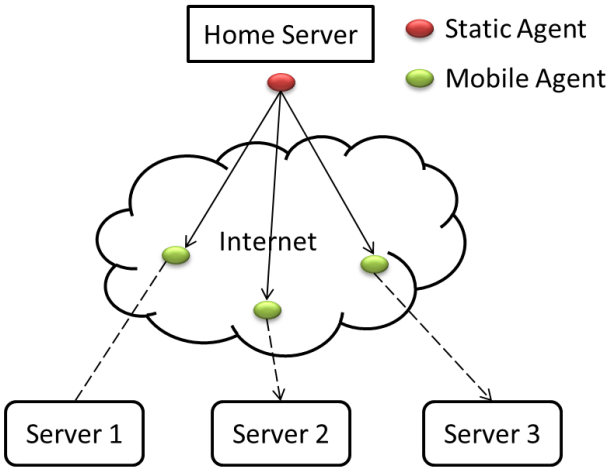

Figure 2 - Mobile agent migration through heterogeneous networks [10].

The figure above (Figure 2) shows the capacity of the agents to be sent through the network to

other computers (servers or home machines) and perform their tasks at that place and return with the result of their tasks.

After presenting the concept of mobile agents, it is really important to understand their architecture and how they work. To this, take the Figure 3 as an example of mobile agent architecture.

In Figure 3, as it is possible to see, there are two sites (A and B) representing two environments with a device (hardware and operating system) and a virtual machine that contains a mobile agent framework that allows receiving and sending mobile agents. In this scenario, the mobile agent is sent by the framework, having their code, data and execution state serialized. Then the serialized stream of data moves through the network and it is received by site B in which the serialized data stream is deserialized allowing the framework to launch the mobile agent in this site. After being launched, the mobile agent can continues its task as it was at the time that the mobile agent decided to move to the other location.

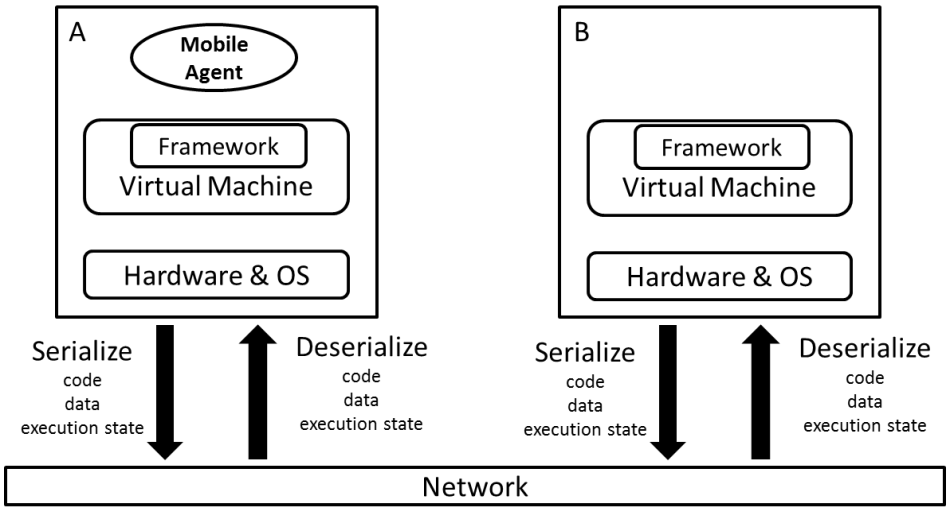

Figure 3 - Mobile agent architecture.

1.2 Contextualization

This dissertation falls under the final project of a master's degree on Mobile Computation. It deals with the topic of mobile agents more specifically "*Mobile Agents Applied to Smart Homes: The Virtual Butler Case Study*" applied in a local environment. This dissertation aims to evaluate a case study which uses mobile agents as an approach to implement the role of a butler and so assist people with their tasks, with special emphasis on human/virtual butler

interaction with natural language.

1.3 Motivation

Nowadays mobile computing is an area in slow expansion and due to that there is the need of new or refreshed ways to deals with data. Mobile agents are an old concept but with this grow of mobile computing its seems to have lost life. For that reason, this work analyses and evaluates the development of a test case that mobile agents could still be used and represent a viable solution or alternative to deal with the mobile computing data communications.

1.4 Objectives

When the mobile agents concept first appeared it had a lot of attention and strong partners to work in their development. However this hard work was vanishing through the years with the arising of the high bandwidth and the low access costs. Despite that, we argue that there are some applications in which the use of mobile agents make sense and could be valuable asset and it is that fact we want to demonstrate in this work.

This dissertation has the objective to demonstrate that mobile agents still have potential in several areas despite everything, and for that, knowing that in Portugal the population is getting older [12] and obviously needs care and assistance in their daily lives, it was implemented a virtual butler with speech recognition and speech synthesizing capabilities, working in a local environment that assists in the day-to-day tasks.

1.5 Organization

The next chapter presents the work related with the mobile agents, speech recognition and speech synthesizing present in the market nowadays. Chapter 3 presents the system architecture of this case study, chapter 4 presents the implementation describing all the decisions made, following chapter 5 with the tests and results of this case study. Finally, chapter 6 shows the conclusion that sums up everything that has been concluded while exploring the theme as well as the problems and barriers encountered and eventual future implementations.

Related Work

This chapter presents and explains the work related with the purpose of contextualizing and somehow shows the evolution in the field of mobile agents projects, speech recognition and speech synthesizing.

2.1 Mobile Agents Projects

As referred above, mobile agents appeared a long time ago and since then there have been investigations and developments in the area which leads to the appearance of some implementations of this concept. However, this concept was never really fully adopted but instead used as an base of knowledge to create other more simple concepts that solve some specific small problems, being today largely used to the point that is almost unknown that they exist, once they are given as granted and fundamental by the world in general.

Mobile agents as previously referred, weren´t massively spread as it was initially idealized and as far as this research goes, there aren´t in the market systems or solution that fully implement this concept. Almost all the implementations were developed for academic case studies or to make proof those mobile agents were a good or viable solution to respond to some sort of problem in a more efficient way. These academic works could be divided in three areas of work: Mobile Agents in Remote Operations, Mobile Agents versus Client/Server Performance and Mobile Agents and Security Issues, which will be addressed next.

2.1.1Mobile Agents in Remote Operations

In the last few years with the growing use of devices and the explosion in the integration of chips and mini computers in the house appliances, there has been an arising of the need to remote communicate with everything, and remotely control everything. To use this remote connection, the Internet, the largest network available, is the base of all communications.

However, today, with the continuous growth in the volume of traffic and even the space exploration, the communications struggle against some difficulties (latency and weak signals). The mobile agent paradigm is presented as having the necessary characteristics to reduce some of the difficulties encountered.

2.1.2 Mobile Agents versus Client/Server Performances

Nowadays, everyone uses the Internet for everything, since playing games, viewing movies and TV, buying tickets or reserving plane sits. And to do this everyone use accounts that store the preferences of every single user. However to maintain this great quantity of data, the companies have or rent servers that can hold not only the data, but also the accesses in real time. For the users to access and use these systems there are paradigms that manage these problems, being the most used the client-server paradigm. But this paradigm presents some problems as latency due to the great quantity of data that is always transmitted through the network.

The mobile agents are really small pieces of software that migrate through a network until accomplishing the task given by the user. As such, mobile agents could be a good add in to the client-server paradigm in the area of the Internet of things. Thinking about this scenario consider a house where every single object has some kind of sensor. Currently with the client-server paradigm each and every one of the sensors send information to the server, which results in two consequences, such as the growth of the network traffic and the quantity of data that the server has to process. If this example expands to the level of a university campus or even a city, the number of sensors increases exponentially and so increase the network traffic and the server volume of data to be processed by the server.

Considering the same example, with the use of the agents, where one agent can room the different sensors, gather information and deliver it filtered in one piece to the server and then the client server process resumes its work, the network traffic would decrease and the quantity of data to be processed by the server would be minimal comparatively. Mobile agents are referred in all the articles that cover this aspect as a replacement for the client-server paradigm but instead as a way to improve and reduce problems once with mobile agents the data is processed in the remote location and only travels back when the task is accomplished, reducing the quantity of data roaming the network and reducing latency.

2.1.3 Mobile Agents and Security Issues

Security issues have always been a concern when using the network and remote connections. For the companies this is a daily concern that sometimes has teams dedicated to investigate and resolve these issues. However, the computers are objects of the everyday and the security attacks are becoming smarter and smarter. As a way to investigate these threats, the mobile agent paradigm is studied because the agents can be programed with intelligence and can learn as they roam through networks which represent a great asset in the motorization of the companies networks and detection of potential threats. Despite these security issues this dissertation uses the mobile agents in a local network which makes the security easier to handle.

2.2 Mobile Agents Frameworks

Mobile agents alone cannot work or "become alive". They need some kind of environment, which is usually called virtual machine. This virtual machine consists not only of the interpreter of the agent programming language but also the execution environment for the agents which is called agent server or agency. [1]

The agency is responsible for receiving, host and executing the agents and offers them an environment that allows the access to local services, communicate with each other and obviously migrate to other agencies. The agency also controls the order of execution of the agents and protects the machine software in order to prevent eventual threats by malicious agents. [1]

The mobile agents alone don't mean anything and in the same way an agency alone doesn´t make sense, even in a network an agency means almost nothing if there aren't mobile agents roaming the network using its services to accomplish some task. [1]

All the existing agency toolkits differ widely in some aspects such as implementation, architecture, security features, communication and mobility. The reason why they are so different even if they are based on the context of mobile agents is because each of these toolkits were developed for a specific objective inside the thematic of mobile agents. As a solution to have each of the different toolkits working together there was a need to create and follow some standards of development that weren't well defined and consequently did not

spread through the frameworks as they were supposed to.

Since the late ninety's many mobile agent platforms were developed. Some of them were abandoned and others continue to receive updates to correct some bugs and add or improve some features. After this initial boom there were some more platforms developed which were completely new or some kind of fresh start of some of the abandoned ones. Some of those mobile agent platforms are Aglets, Voyager, Grasshopper, Tryllian, JADE, Tracy, and SPRINGS. This quantity of platforms, make the decisions more difficult for someone who want to benefit from this technology [1].

This section explains and details some of these mobile agent platforms, chosen specifically because of their characteristics. The chosen platforms are TCL (probably the first framework that can be considered as a mobile agent platform), Aglets (probably the most known of the mobile agent platforms), JADE (a newer and popular platform with an active community), Tracy (a platform that has led to the publication of the most recent book on mobile agents [1]), and SPRINGS (a very recently developed and maintained platform for academic purposes).

2.2.1 TCL

TCL is a scripting language created in 1987 that was very popular at the time and is now maintained at Dartmouth College. When TCL was developed the mobile agent concept were in his early days and there were few frameworks using the concept and those same frameworks had flaws, which served as motivation for the development of TCL [13].

The initial systems that are referred above were Tacoma and Telescript. Tacoma weaknesses for example, were the fact that it requires the programmer to explicitly capture the state information before starting the migration and the inexistence of security mechanisms. On the other hand, Telescript wasn't opened for developers, required powerful hardware and was limited to a single language [13].

TCL appears with the goal to address the weaknesses of the mobile agent systems. It could run in standard hardware, support multiple languages and transport mechanisms (ex: TCP/IP) and offer a security and fault-tolerance. On top of this, TCL could be embedded in other applications allowing the applications to use its functionalities, proving its simplicity and

simplicity to any programmer [13].

Despite its simplicity and intention to address some flaws present in the other platforms, TCL itself has some disadvantages. TCL is inefficient if compared with other languages, it isn´t object-oriented and doesn´t allow code modularization aside procedures, making really difficult the debug process if working with large script files [13].

Until the early ninety's the problem of a non-object-oriented and unstructured language wasn't really a problem since the agents themselves were small and made use of other tools to process information. However, with the increasing of the information the agent started to grow and other languages like java started to be considered [13].

2.2.2 Aglets

Probably Aglets is the most popular mobile agent platform developed so far. It was initially developed in 1997 by IBM, becoming an open source project since 2001 and maintained by this community until February 2002 (last version: 2.0.2). [2]

Aglets platform has some strong points that must be noted as the use of the MASIF[2] specification and the use of proxies as a form of abstraction to refer to other remote agents. The Aglets mobile agent platform communication infrastructure is based on a message passing system that allows both synchronous and asynchronous messages, built around a single-thread model. [2]

The Aglets, due to their popularity, have made a significant contribution to the field of mobile agents, despite the level of activity related to this platform nowadays being quite low. To this decrease has contributed not only the arising of other platforms but also some disadvantages native to the platform itself. The use of proxies despite being useful because of the abstraction offered, were also a disadvantage due to not being dynamic. Once the agent they point move to another location the proxy can no longer point to it, forcing the programmer to obtain an updated proxy before using it again [2].

[2] MASIF (Mobile Agent System Interoperability Facility) were the first standardization approach to mobile agent toolkits formerly known as Mobile Agent Facility (MAF). "MASIF is a set of definitions and interfaces for interoperable mobile agent toolkits." [1]

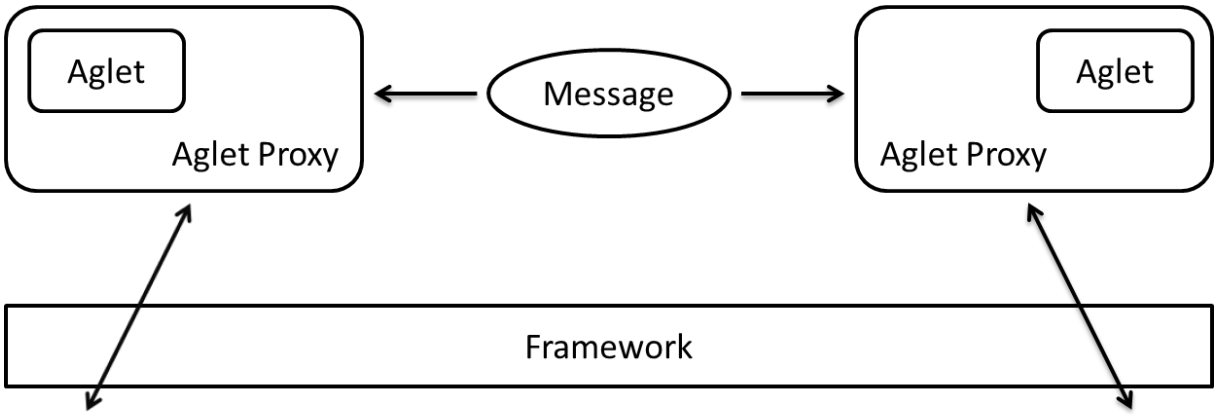

Figure 4 - Aglets representation.

Another disadvantage is the fact that every agent is assigned to a single thread, which can cause deadlocks if two agents decide to send a message to each other at the exact same time (each agent stays blocked unable to receive any message until the other has received its message).

Aglets are completely developed in Java, both agents and the platform itself, which offers a great portability between systems. The Aglets platform is available as a set formed by three components: mobile agent platform, a standalone server named Tahiti, and also a library that allows the programmers to include mobile agents in their programs.

2.2.3JADE

JADE is a mobile agent platform developed by Telecom Italia in 1998 that became an open source project in the year of 2000. Its last version is JADE 4.3.1 that was released in December 2013. This mobile agent platform is probably the most popular FIPA[3]-compliant platform.

Once again this platform was also developed with its one purpose and has some benefits (others than being FIPA-compliant), that allow it to stand out against other platforms. One of its benefits is the large variety of tools offered like the possibility to manage and monitor the agents remotely. Another benefit is that JADE can be integrated with other software. In this

[3] FIPA (Foundation for Intelligent Physical Agents) was founded in 1996 at Swiss. This set of specifications represents a collection of standards created to promote the interoperation of heterogeneous agents and all their services.

group of software's that can integrate JADE, there is one that is worth to mention, Jess. Jess is a rule engine where rules can be declared and allow JADE agents to interoperate using the knowledge contained in the rules. With the cooperation of this and other similar software, it's possible to represent the knowledge of the agents with the development of ontologies.

JADE also has disadvantages and probably the most notorious is that mobility isn't one of the principal concerns. However this mobile agent platform focuses in other features like the fact that the platform supports the use of containers in the same platform, allowing the agents to move among containers of the same platform.

2.2.4Tracy

Developed at the University of Jena in Germany, Tracy is a mobile agent platform that has a plugin-oriented architecture where last version (1.0.1-52) was released in April 2005. Due to having a plugin-oriented architecture, Tracy allows a wide range of plugins to be added to the agencies in order to add more features to the agencies. Examples of these features are security features, migration, and of course, communication among agents. As this mobile agent platform allows the addition of functionalities through plugins, it is possible to use different migration strategies form the agents. [2]

The key disadvantage of this mobile agent platform is that Tracy does not support the remote communications among agents, which forces one agent to migrate to the platforms where the other agents are, in order to communicate with the other agent. [2]

Tracy was most probably developed as an environment to test and evaluate migrations and class loading strategies could be evaluated, and this meritorious work results in writing and publishing the most recent book about mobile agents. [1][2]

Once this work had been an experimental research task, this platform has few documentation even with the book that only covers the general aspects of mobile agents, which has probably contributed for its low level of activity [2]

2.2.5SPRINGS

SPRINGS is one of the few platforms that still has maintenance and support once it has been

developed by the Distributed Information System Group at the University of Zaragoza (Spain). This mobile agent platform has its strong points in scalability and reliability, which makes it good in ambient with relatively high number of mobile agents. Once it has been developed only recently and by a university, SPRINGS has inspired itself in features of other platforms like Voyager and Grasshopper. [2]

As used by other platforms, the hierarchical architecture of SPRINGS is based on regions. With this characteristic, it also offers full location transparency for movements, which allow the programmer not to specify the network address of the remote destinations, he only needs the name of the destination. Above all, this mobile agent platform has mechanisms to attempt to minimize the live lock problems that can occur when agents move too quickly. [2]

To this mobile agent platform, one of the features that count as a disadvantage is perhaps the lack of support for agent communication using the standard FIPA and the fact that the security mechanisms provided aren't sophisticated. As a conclusion, it is also interesting to refer, that this platform doesn't offer any graphical tool to the user, as due to being an academic implementation, only available if requested, there isn't much documentation to support its development. [2]

2.3 Voice Recognition and Synthesizer

Home automation systems, among others, currently started to have some kind of voice interaction with the users. As such, this section presents a brief review about some of the systems available on the market and in research community.

2.3.1Windows Speech Recognition

Microsoft has been working since 1993 in the speech recognition algorithms and processes trying to accomplish the tasks speech to text and text to speech with the most accuracy possible which results in the Speech API (SAPI).

Windows Speech Recognition is an application created by Microsoft and it is part of their operating systems since Windows Vista was released in 2006. This application allows the user to interact with the computer by voice commands like "open calculator" in order to make the computer do some task. But it does not only support some specific voice commands, it also

allows the user to interact with other programs using the little numbers and letters that can be usually seen in the menus when pressed the key “Alt”.

Microsoft also pushes the functionality of voice recognitions further and it is possible to speak to the computer and the speech is converted to text which is especially useful to people who have some degree of disability.

The initial Speech API created by Microsoft evolved and it is now in version 5.3 which serves as a base of the application used nowadays in Windows. Currently Microsoft Speech Recognition is very accurate and offers support for different languages like English, Spanish, German, Japanese and Chinese among others.

2.3.2 Google Speech Recognition

Google as we know, is always in the frontlines of technologies and with the creation of the android more and more services, libraries and API’s where developed and distributed for everyone who wants to develop which helped this initial software to grow. However not all the official libraries by google are official, some of them are like experiments that are under development or tests and google leak it to the Internet.

One of these libraries developed and released to be used by the general public is the Speech Recognition which we can use in google webpage or in the android devices. But since google primary focus is the browser and the android devices, almost all libraries are developed for that purpose, making more complex for someone to use them in native desktop applications.

Thus the fact that the libraries are often released to general public, some developers managed to create new ways to use the speech engines created by google, one of these is J.A.R.V.I.S. Speech API.

Despite google API offers advantages such as speed in processing and Multilanguage speech recognition, these engines obligate the users to have Internet connections and there are not much documentation to use due to the fact that desktop API are “pet” projects developed by enthusiast developers that make use of the google web engines. Nowadays, to use the google engines a key is required and must be obtained by the developers.

2.3.3CMUSphinx4

CMUSphinx also known as only Sphinx is a speech recognition system developed by Carnegie Mellon University and that was made open source in 2000. CMUSphinx4 was developed in partnership with Sun Microsystems laboratories, Mitsubishi Electric Research Labs, and Hewlett-Packard's Cambridge Research Lab. This version 4 is an evolution of the previous CMUSphinx systems in terms of modularity, flexibility and algorithms. This new version is entirely developed in Java which makes it highly portable. Sphinxs4 accepts different grammars and languages and acoustic models, due to the fact that it uses Java as its main language, it also allows multithreading.

Sphinx4 offers the developers a simple way to define the grammar that the system uses, which makes possible to create a very easy grammar or a very complex and specific one and all of these without using an Internet connection which is perfect to be used in local and particular implementations of speech recognition.

2.4 Works with Voice Recognition and Synthesizer

The area of voice recognition and synthesizer is an area that has been researched for a very long time. Developers have always tried to interact with the computers by voice commands which results in a large number of software and libraries available today. Some of these libraries are old and work locally, others requires Internet connections and use remote engines to convert the voice to text and vice-versa. Lately, with the arising of smartphones and tablets the interaction with the devices using voice has been a major concern and a focus of development which lead to the development of new software and libraries. This makes these voice recognition libraries to act as little personal assistants, being some of them well known by the users. It is the case of SIRI from Apple, Google Now from Google and Cortana from Microsoft.

2.4.1SIRI

Siri is a software application that works as a personal assistant that allows the user to ask questions which SIRI will answer or tasks that SIRI will perform. This system was created in 2007 by Dag Kittlaus and his team, being acquired by Apple in 2010 and included in the iPhones operating systems (IOS). This software recognizes natural language in different languages and uses artificial intelligence (AI) and a set of web services to choose and

elaborate the best answer to give to the user request.

2.4.2 Google Now

Google Now is also a software application that works as a personal assistant that listens to the user natural voice and answers questions or trigger tasks by user request. This application is an evolution of Google Voice Search, although keeping the voice recognition capability. Google Now has support for a large number of languages and increasing, and has the advantage that it can be used not only in mobile devices but also in computers through the use of a browser.

2.4.3 Cortana

Being Microsoft one of the three great players in the market of smartphones and tablets and having the other two (Apple and Google) already developed for their devices their own personal assistants with the capability of natural voice recognition and synthesizer, Microsoft developed Cortana. The capability to recognize natural voice in Cortana, has support for a set of languages all over the world without the need of previous configurations, just talk and Cortana will recognize the language and answer using the language perceived.

2.5 Synthesis

After the research to find or discover software that even in just a small part could use mobile agents, it's safe to say that there isn't or, if they exist are so small in the market that are unknown. However, the great use and investigation of mobile agents isn´t to replace anything that we already use nowadays, but instead to reduce their flaws and optimize tasks. It is important to understand that mobile agents themselves don´t have any advantage to the other paradigms that are nowadays used; however, if we analyse all the joint advantages, they are really strong because the alternatives are good for unique advantages but none of them can support the same set of functionalities as mobile agents [7].

This research has shown that on the other hand, the speech recognition and synthesizer software has been used for a very long time and have been greatly developed due to the mobile devices. Being used in mobile devices, and being the mobile devices mostly property of three large companies (Apple, Google and Microsoft), it has got the infrastructures which offer the engines support for a large and growing number of languages.

In short, it is possible to understand that mobile agents are starting to gain a second life with the arising of mobile devices. At the same time, speech recognition are at the top being more and more a key feature for the users. Thus, if these two systems can work together, probably something new and useful could be developed.

System Architecture

This chapter describes the system architecture of the case study with all the components forming part in the test case. First, it is presented a macro view of the architecture which is explained and detailed piece by piece, finishing with a closer view of the test case architecture used.

3.1 Architecture

The next image (Figure 5) shows a macro view of the architecture of the test case and all the components that take part. This architecture shows the mobile agent as the center of the architecture and it is him who is responsible for making things happen.

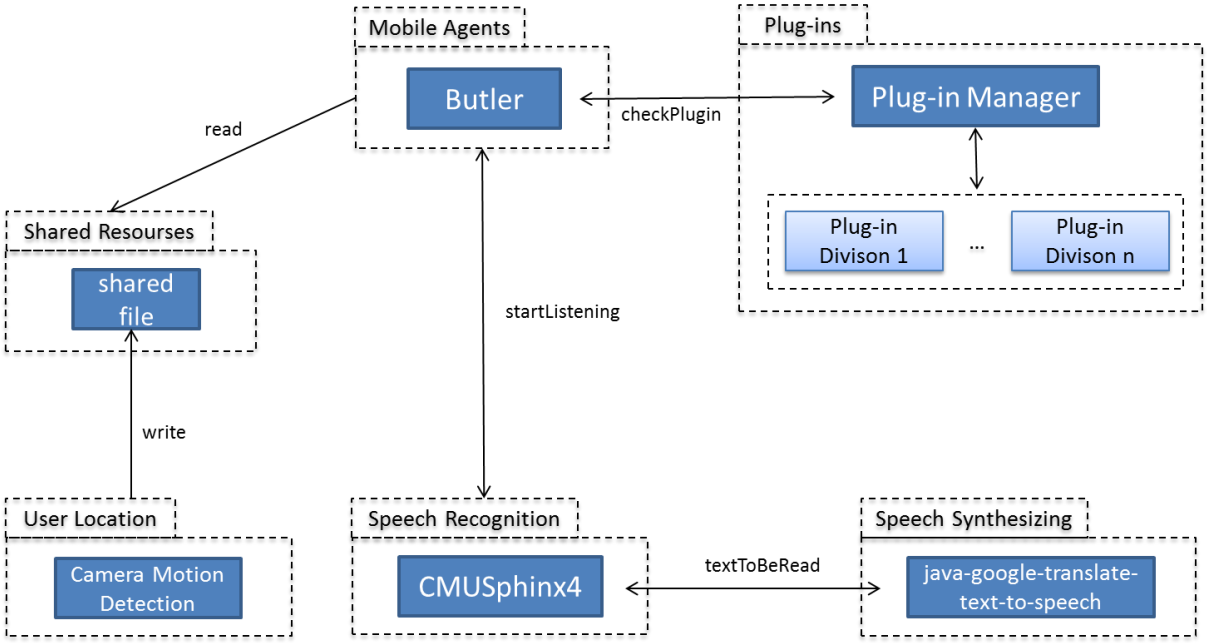

Figure 5 - Architecture (macro view).

As shown in the above image (Figure 5), the camera motion detection detects the presence of the user and writes to the shared file that will be read by the mobile agent. The mobile agent

will then migrate to the user location activating the speech recognition to be listening to the user commands. The commands given by the user are then analysed and an answer sent to the speech synthesizing that will play it.

3.1.1 User Location

The user location showed on the above image represents all the systems that can detect where the user is in real time so that the agent could be informed and move to the user location. In this prototype it was used camera motion detection to detect the position of the user and inform the agent. To do that the camera motion detection when detects movement writes to a shared file the IP address of that specific division of the house overwriting any other IP address that could exist in the shared document.

Instead of the camera it is also possible to use all kinds of sensors to detect the user and even combine different kinds of sensors and cameras to a better and more accurate detection.

3.1.2 Shared Resources

Shared resources were used as a simple way to register the information that the camera motion detection needs to pass to the agent. In this prototype the shared resource is a simple text file that register the IP address of where the user was detected and that allows the agent to read it in order to know where he needs to move. It is probably not the best way to inform the agent to where he needs to move, but since it was not the principal concern of this project, shared resources were the easiest way to send the information to the agent.

3.1.3 Mobile Agent

The mobile agent is the center of the architecture and the main subject of this test case. The mobile agent is the principal piece that has to room through the house to communicate and control the other pieces of the prototype in order to make all of them work together and accordingly to the needs of the user.

In this prototype, the agent checks the shared file from time to time with the single purpose of discovers the next location to move. Having a new location to move the agent migrate to the destination taking with him all the other pieces contained in the prototype. Arriving at the new location the agent instantiates the speech recognition classes to starts the speech recognition

to be listening for voice commands.

3.1.4Plug-ins

In this prototype the plug-ins were one addiction made in order to allow the agent to evolve with the minimum impact for the user. This piece allows the user to add plug-ins and with the minimum effort having them installed and running. The plug-ins are xml files with a specific structure and name that allows the "plug-in manager" to discover and parse the files.

In this prototype it is the agent, by user request, who checks if there are plug-ins to install and launch a "plug-in manager" that has the function to parse all the plug-ins (xml files) and accordingly their parameters generate a new java file that will overwrite the existing one and, this way, make the changes contained in the plug-ins available.

3.1.5Speech Recognition

The agent can be more useful as many interactions it allows with user, and the speech recognition are one of the best ways to make the interaction between the user and the agent natural. Speech recognition is an important piece that allows the user to command predefined actions to the agent.

In this prototype the speech recognition is started by the agent to be listening to user commands. When it is detected a user command the command is put to text and analysed to verify if the command is contained in the grammar file that contains all the possible commands and variations, then the speech recognition launch the speech synthesizing and sends the text, as a function parameter, to be converted to audio and played by the speech synthesizing.

3.1.6Speech Synthesizing

The speech synthesizing is the final piece of this prototype and it is responsible to communicate with the user using audio responses. These responses are also predefined to each and every command given by the user.

In this prototype when the speech synthesizing is launch and receives a text parameter, it detect the language of the received text and accesses a Google service through an HTTP GET

request to convert the text to an audio file. This audio file is received and played to the user. It is these audio responses are the ones that allow the user to perceive if his commands have been perceived or not.

3.2 Test Case Architecture

In this section the technical part of the project is planned since the definition of the operating system to the libraries that will be used for speech recognition and voice synthesizer.

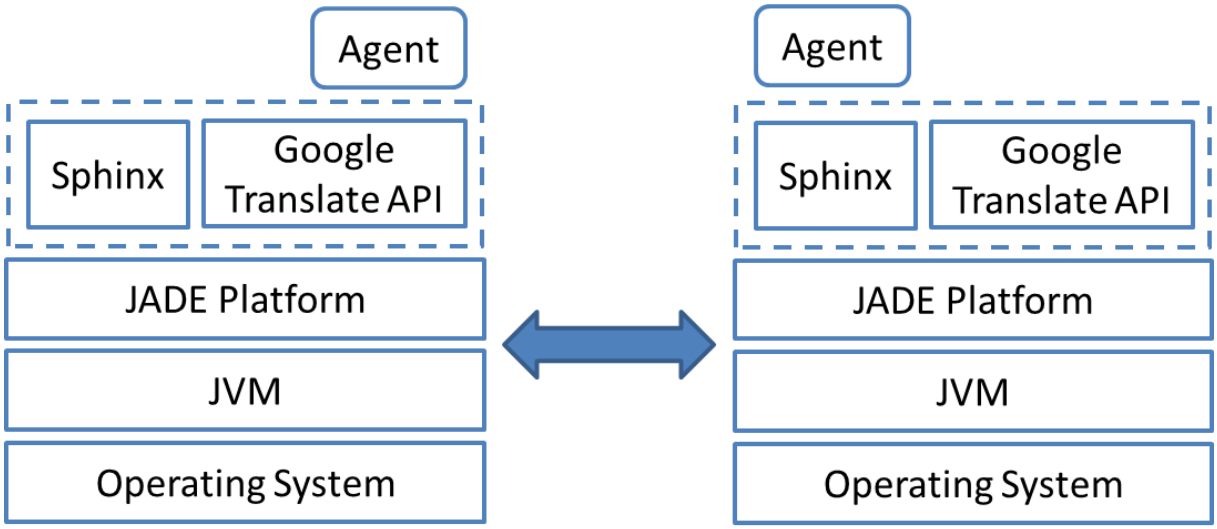

Figure 6 – Test case architecture.

The image above (Figure 6) shows the architecture of the functional project. The project executes over an operating system which represents the first level of the architecture.

For the development of the project it was used Java and, as a consequence, to develop in Java it is necessary to the Java JDK installed and which will offer the JVM (Java Virtual Machine) necessary to run Java applications. In top of the JVM it is JADE platform which is the platform that offers to the different agents the possibility to room the network between different computers. As it is noticeable, the agents run over this platform that is installed in every computer of the local network. Although the software/libraries for speech recognition (Sphinx) and speech synthesizer (Google translate API) doesn´t need to be installed in each computer of the local network that will receive the agents once it goes with the agent itself.

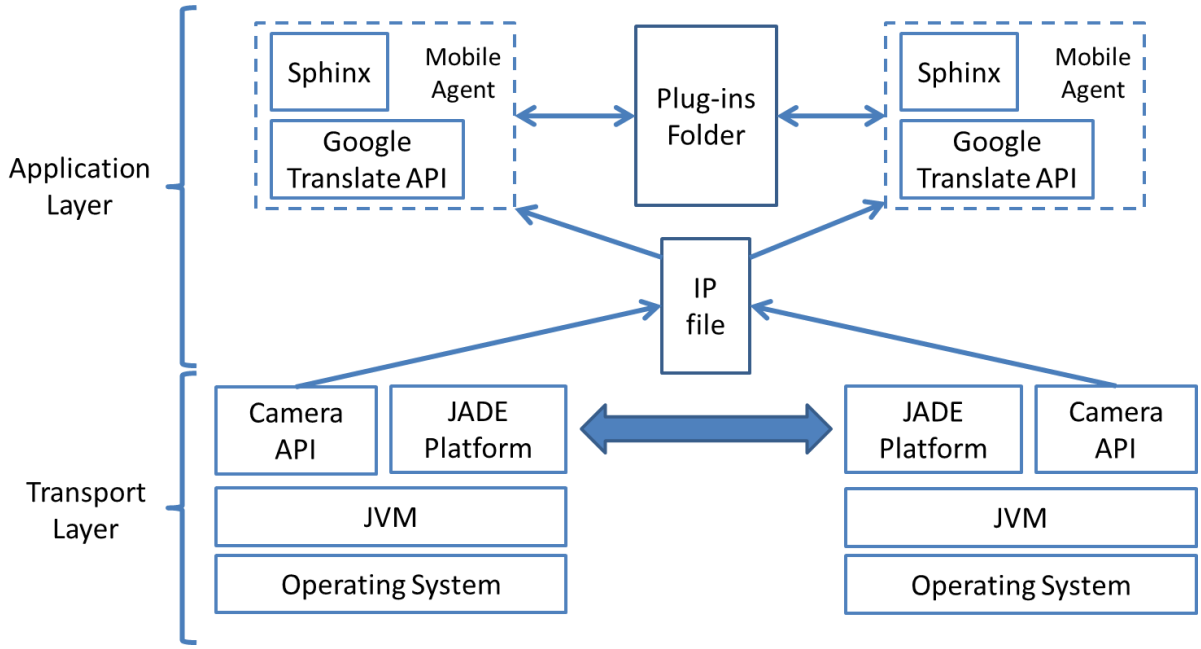

Figure 7 - Detailed Architecture.

At last, the agent itself is independent from all the libraries that offer new functions to the project, and this way, it stays simple, agile and fast while rooming the network.

Implementation

Following all the work and research carried out, a simple functional test case was developed in which mobile agents can be applied and used with efficiency. This test case is oriented to a house as scenario and a local network to allow the agent to communicate between the different devices in the house, in which the main purpose is to simulate a digital butler capable of voice interaction with the users.

4.1 Software and Hardware

For the test case environment it was used as hardware two normal laptops without extra performance with very common characteristics as shown in Attachment 1.

The software used for the test cases was chosen in order to obtain a digital butler in a local environment. There was a lot of software that was investigated in this work and probably there is other software that can accomplish the same goals in a simpler or at least different way. However, with the search conducted, some decisions were taken that will be explained next.

It was used Java was used as a main programming language in this prototype for various reasons. For starters Java programming language is not as other conventional programming languages where the code is compiled natively. Java uses a Java Virtual Machine (JVM) to run a bytecode that is generated when Java files are compiled. This particular characteristic of Java allows the developers to write once and run anywhere (WORA) which makes Java the ideal programming language to run in any operating system without the need to make changes or adjust the developers code. On top of that, currently almost all sensors accept some kind of basic Java as programming language or offer some kind of plugin to deal with their events, and if we remember the definition of mobile agents, it says that mobile agents must run in any environment and operating system, which makes Java the logical choice as programming

language to create agents and make them communicate with the available sensors.

As previously explained, mobile agents is a concept that is not new and that had a lot of exploration and evolution for years. However, the network, programming languages and frameworks themselves have become obsolete and without the capacity to evolve further, being lately discarded and adopted the Java programming language as the main programming language due to the characteristics already described.

Thus the platform to choose would need to use Java as programming language and there were some that were really good and already been used in other academic works; however, in a better research it was concluded that the great majority of these frameworks are currently open source projects and no longer under maintenance which lead to JADE.

JADE is a Java platform to manage mobile agents, which is currently under maintenance and has a very active community that participate in the JADE forum helping anyone who works with JADE. In addition, this platform has a lot of tutorials, examples and documentation to anyone that starts the development of mobile agents with JADE. On top of that, JADE is currently perfecting its own variation of the platform to allow it to run in mobile devices which will allow the agents to migrate from computers to mobile devices (Android uses Java as native programming language) and come back to computers, and this particularity is really important once it represents a significant growth of the agents available environment to migrate and accomplish tasks.

As an IDE the choice was netbeans over others, like eclipse because it is simpler to use and more intuitive, and for the creation of graphical interfaces it does not need to install any additional plug-in. In addition to this, netbeans is smaller when installed and consumes less memory while running. Another reason was the fact that all the examples and tutorials of the software researched shown netbeans as IDE, which makes it simpler to understand tutorials and the configuration of settings to launch the agents and also to interact with the voice recognition software used and configured to work in their test case.

As described in the chapter Related Work, there was a revision of some solutions to implement a basic speech recognitions program and finally the CMUSphinx4 was chosen. The decision was not really easy to make and among the three software explained in the

above chapter, Windows Speech Recognition as the name indicates is Windows and only works natively in Windows which is against the mobile agent concept where the agent must be capable to migrate to different environments. Despite that Windows speech recognition could be a good choice once it already supports different languages and has a good accuracy and noise suppression algorithms.

Another reviewed software was Google Speech Recognition and this one was really the option that was considered as the best of all. This google engine has one of the major range of languages in speech recognition if not the major itself, which would be a must have feature to add, and the use of these library seems to be really easy there is only the need to speak to the microphone, the stream would be stored locally and sent to the google engines to be processed and returned the most accurate text possible, that would be used to launch some events. However, this solution revealed some difficulties that lead to discard this option, as the fact that to communicate with the google engine it was obviously required an Internet connection, and being the test case environment a house with local network, it was preferred the usage of a library that works locally. In addition to this choice was the fact that the documentation is scarce and this library is not an official google product, but a lab version instead.

At last, choice was CMUSphinx4 which initially seemed to have the disadvantage compared to the other choices, but at the end and due to the test case, it became the best choice to make. CMUSphinx4 was entirely developed in Java which allows it to run in any platforms and this characteristic works perfectly with the mobile agent concept. It is under ongoing maintenance and has a lot of examples and documentation available to help understand how it works. This system does not require Internet connections, working entirely offline without any problem while its main disadvantage is in our opinion the lack of ready to use recognition languages.

Despite this lack of native multilanguage recognition, CMUSpinx4 was developed with some complexity but that offers the developers relatively simple ways to add new languages to be recognized and define the syntax and semantics of each language which makes it ideal to configure and parameterize events and actions to be done when a specific command is received or define eventual responses to make the system more similar to a human and not a machine vocalizing words.

Logically, if there is a speech to text, it is nice to have text to speech, and even if this is not

the main focus of this research work, a quick research was made and ended up with two possible solutions. The first is the google voice that can be used with a simple url where we concatenate in a variable the text to be read and when we run the url the mechanic voice of google will read the given words or text. The second one was eSpeack. This software can run locally without Internet connections and it is high configurable and tunnable and it is multiplatform which allows it to run in any environment. However this has the disadvantages that its voice is more machine-like and not human like, but for this research project it has the necessary requirements.

In the end it was used a library that offers the best of the initial two options, Google Translate API. This Google library can run offline without any Internet connection and already has a large range of languages supported and working with the change of a simple parameter of the configurations. This decision was made looking to the future, as this test case may support other languages beyond English.

This dissertation implements mobile agents in a local networks which is configured in a house, as such, for the agent know where to move it is necessary the use of some kind of sensors that could detect the presence and movement of the users. As such, it was chosen the camera motion detection that provides the necessary capabilities required to detect the presence of the user in the house.

4.2 Mobility Environment

As already mentioned, the project created is the development of a generic butler based on mobile agents with speech recognition and voice synthesizer capabilities and the environment in which this butler would operate is a house with a local network (Figure 8).

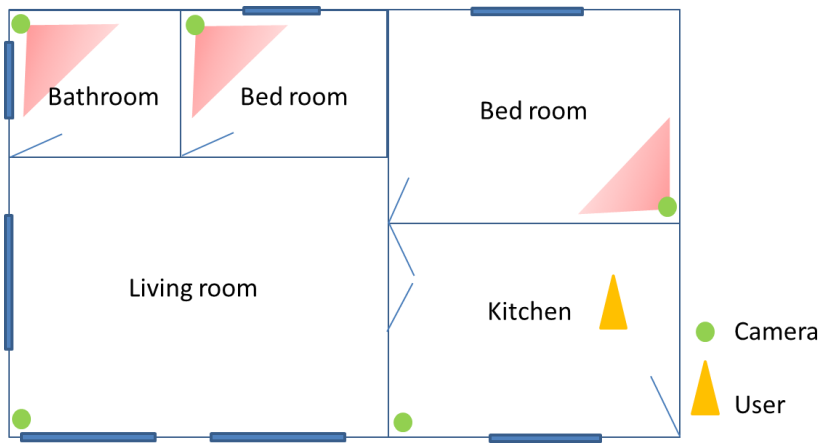

Figure 8 - Home plan.

This test case can be mounted in two separated parts: the hardware and the software. The hardware used was two laptops that were already described in Attachment 1 and two microphones that, in this case, are embed in the laptops. Each of these laptops represents a division of the house (ex: one represents the kitchen and other the bedroom).

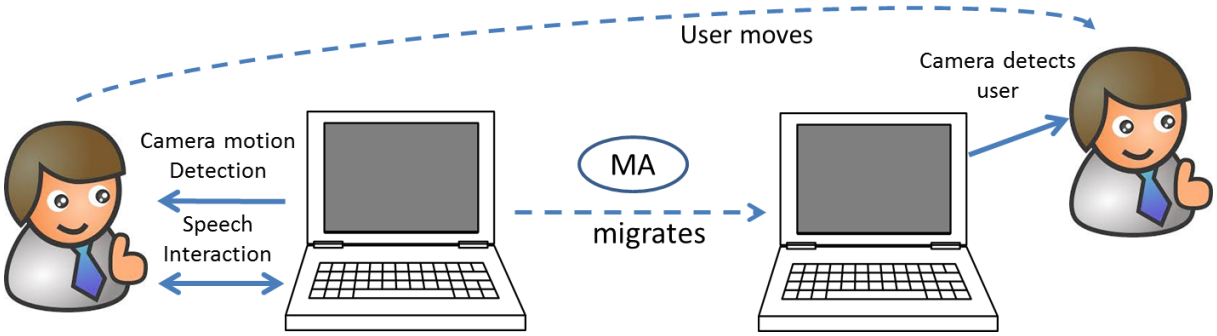

Figure 9 - Test case representation.

First of all it was created a Java project with graphical interface only to have some kind of avatar image so that the user feels closer to the virtual butler. This test case will hold all the classes and libraries necessary to the project development. To this clean new project were added the necessary libraries for each feature of the developed test case, which are: mobility of the mobile agent, speech recognition and speech response.

4.2.1Mobility of the Mobile Agent

For the implementation of the mobile agent and its mobility were chosen the JADE Platform and so the library "jade.jar" is required to the development so that the agent could move between the computers set in the local network. The agent is essentially coded in two files, one a Java class for the code itself of the agent and its actions and the other a JFrame for its graphical representation to the users so that they can relate with the system instead of speaking to something that they could not see.

At this point the basics for the agent were set but the agents need a platform to send and receive them. This platform is offered by the JADE Platform and needs to be running so that the agent could move. In order to make this work in the project were needed some configurations and are described in Attachment 3. At this point and having a JADE platform running in another machine in the same network, the agent could already migrate between machines taking with him the source code and maintaining all its functionalities.

4.2.2Speech Recognition

Another feature of this test case is the speech recognition and for that it was chosen the CMUSphinx4 software. To include this software in the test case there were some libraries needed to be added in order to interact with the agent by voice commands:

- Jsapi.jar
- sphinx4.jar
- TIDIGITS_8gau_13dCep_16k_40mel_130Hz_6800Hz.jar
- WSJ_8gau_13dCep_16k_40mel_130Hz_6800Hz.jar

Having the libraries added to the main project it is possible to implement the speech recognition. For that and to maintain the project organized, it was created a new package to hold all the files required to the speech recognition. There were then three files added to this new package, a java class file, a grammar file and a configuration file, so that the speech recognition could be parameterized according to the needs.

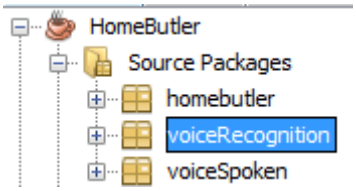

Figure 10 - Project packages.

The first file parameterized was the grammar file in which more words are defined to the dictionary of words of CMUSphinx4 that are not defined in the default acoustic models. For that it was created a new file with the extension ".gramm" and started to define new words and sentences needed in the implementation. To work with this file it is mandatory to respect a specific writing structure of the Java Speech Grammar Format (JSGF).

```
1  #JSGF V1.0;
2
3  grammar simpleExample;
4
5  public <greet> = ( Hello | Hey | Hi );
6  public <completeGreet> = <greet> ( World );
```

Figure 11 - Grammar file example.

These JSGF files represent one grammar and are divided in two parts, the grammar header and the grammar body. The header required the tag “#JSGF” and the optional annotation of the version or char-encoding locale. After the header, it is mandatory to define the grammar name which will allow calling the grammar from the java files. To define the name it was just used the key word “grammar” followed by the name of the grammar as shown in the figure above (Figure 11). With the grammar header and grammar name defined, it was defined the grammar body which also has a specific structure to define the new rules of the vocabulary.

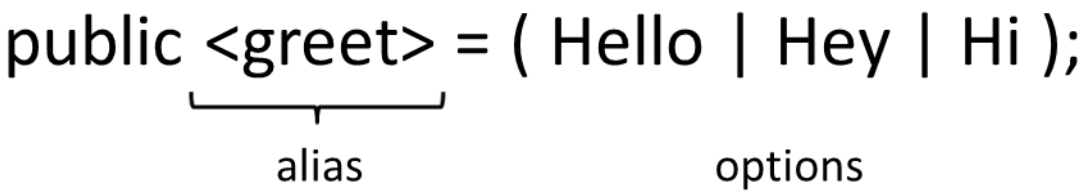

Figure 12 - Grammar rule example.

The structure of each rule is really simple. First it is necessary to define an alias as shown in the figure above (Figure 12), and it must be unique inside the grammar once if for some reason it is used again it will be overwritten. After the alias, it is necessary to define the real words or sentences that must be recognized by CMUSphinx. This definition can have a great number of possibilities that can be organized in order to make the communication really close to a real conversation. The words must be declared in brackets and the different possibilities inside a rule must be separated by pipes (|), as shown in the above Figure 12.

The second file is the configuration file and it is a mandatory file that holds the configurations of the inputs and outputs of CMUSphinx4, with the references for the four libraries used in the process of speech recognition and respective acoustic models, that could be refined if necessary for improvements.

The third and last one of the three files is the Java class file in which it is coded all the logic since the initial voice detection to the voice recognition result and what that results will trigger. In this file it is first defined what grammars to use, and next started the microphone recording to be listening to the environment waiting for the vocabulary defined in the grammar to be used. This constant listening will be sent to analysis any word detected and continues listening even during the analysis. After being analysed, the results are sent back

and will enter in a sequence of validations to verify if that specific result triggers any event or action. The record, analysis and validation of the result are a real fast task, and even so the microphone continues always listening and starting this process continuously.

With all this integrated in the same project, mobile agent and voice recognition, it was everything set to start personalizing the agent and the responses to the voice controls.

4.2.3Speech Response

The other feature included in the butler was the speech response so that the butler can communicate with the users. This speech response is essentially a text-to-speech action triggered as result of an interaction with the users through speech recognition. For this it was used java-google-translate-text-to-speech which is an unofficial API that offers all the main features of Google translate. To include this API in the project there was the need to add two libraries:

- gtranslateapi-1.0.jar
- jl1.0.jar

With the addition of these two libraries and an Internet connection it was possible to code the text-to-speech feature, in order to keep the code organized, it was created another package inside the project to hold the files required.

Only one simple file was needed, a Java class file where it was defined the spoken language of the feature and with the action of playing a text that would be received when the program was running. At this point the three major features where implemented and ready to work.

4.2.4Camera Motion Detection

The camera motion detection feature, was implemented to show how to make the agent move through the house network following the user wherever he is. As such, this is an independent piece of software completed unbound with the rest of the project presented above.

This little independent piece of software has as only purpose to detect the presence of the user in each room of the house as the user moves. For that, it was used the Webcam Capture software. In order to make this software work with the agent each room was given a fixed IP

address and so, having a little program configured for each room, when it is detected movement through the camera, the program writes for a shared file in the network the IP address of the room.

The agent itself is checking this shared file in a brief time interval in order to verify if it was changed since last time the agent checked it. If it has been changed, then the agent will move to the room of which the IP written on the shared file corresponds.

4.2.5Plug-in Architecture

Having the test case working, became possible to notice that it was limited to the initial parametrizations defined which means that it would have only a limited series of words/phrases with correspondent event/action. Noticing that, it was developed a simple system that works as plug-ins which allow to expand the series of words/phases and respective event/action.

This plug-in architecture is really simple. It consists basically in a shared folder that contains two kinds of files: grammar file and an xml file.

The grammar file or files are the files in which the words that will be recognized in the speech recognition are defined. Having this file(s) in this folder, it is possible to add, remove and edit the grammar and expand it to include more and more words for all kind of scenarios, and this way make the speech recognition more accurate.

The xml file is a file that holds the event/action to do for each and every single word or phrase existing in the grammar file. As such, both of the files must be updated together in order to accomplish accurate question-answer interactions with the user. In this xml file the event/actions could be java code or command-line commands to be executed.
Imagine the following example: the user needs to open the windows calculator and next exit the program. These are two actions that could be performed and, as shown in (Figure 13) the action of the opening of the calculator could be accomplished using a Windows console command while the second needs a java code sentence to be executed. For this purpose, it is needed a file like the one showed in Figure 13.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Commands>
    <Command>
        <Type>console</Type>
        <Keyword>calculator</Keyword>
        <Action>cmd /c start calc</Action>
    </Command>
    <Command>
        <Type>java</Type>
        <Keyword>exit</Keyword>
        <Action>this.dispose();</Action>
    </Command>
</Commands>
```

Figure 13 - Event/action XML file.

The butler when launched spreads to this shared folder, reads both files and generates the necessary code to execute the event/actions, overriding any other code that could exist before. With this the butler has all the newest information to work and room the local network interacting with the users.

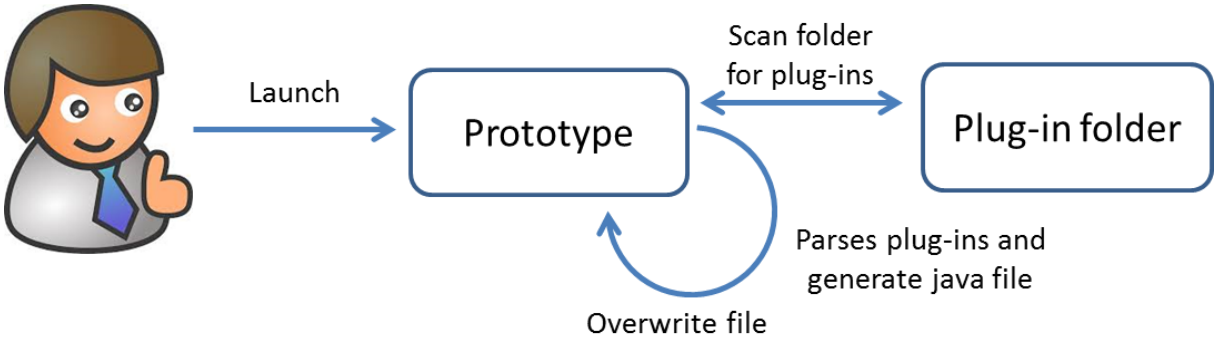

Figure 14 - Generation of the java file after reading the plug-ins (xml files).

When the butler read the files (plug-ins) it parses its xml content and, according to the xml parameter, it is inserted in a new java file a new piece of code according to the reading parameters. After it is finished, the new java file is added to the jar file containing the butler source code and the butler re-launches himself.

With this plug-in architecture, it is possible to add or remove files from the shared folder according to the needs without any special effort and with the only flaw that the project has, to be launched again in order to reload the files on the shared folder.

4.3 Challenges

The development of this research project involved a lot of research, examples implemented

and experiences tested out to accomplish the objectives of having and mobile agent moving though the local network with the ability to receive voice commands and respond back in order to assess it. However this was no simple task and there were some challenges encountered. First of all it was the decision of choosing which software to use. Java is a language that works in all platforms and as such there are a massive amount of libraries and pet projects to accomplish similar objectives, developed by a wide range of people since a good programmer to a high school kid that developed something as pet project. Due to this fact, the documentation on the majority of these works are scarce, there are no examples and the code is no longer maintained.

The other challenges were faced while developing the project. After having the entire three feature working together, it was time to implement the objectives and refine the solution. However, when making the code more flexible allowing reading the actions to be executed from an xml file one problem was encountered: the code only worked if the actions were command-line commands (ex: MS-DOS commands). To solve the problem of the xml file, the xml were modified to include one more element indicating if the actions are Java commands or command-line commands.

The next challenge appeared after resolving the previous one, and it was how a piece of code could be included in an existing Java class file that was already created inside the .jar file resultant of the project compilation. To overcome this problem the agent parses the xml file it creates a new Java class file with all the code created according to the xml file. After the class created, the agent executes a command-line command which allows this Java class file to be added do the .jar file overriding the existent one (Figure 14).

The last but not the least challenge was to create some kind of running file that could be distributed to clients. Developing in Java, when the project is compiled, it is generated a .jar file that works as a running file containing all project; however, due to the fact that there are some configurations necessary to launch the JADE Platform so that the agent could be executed, for some reason, this .jar file does not work. The JADE Platform is not launched and so the agent cannot be executed. To solve this problem it was created a base application with the unique tasks of launching the JADE Platform and next the prototype itself, which allows to have a running file that could be distributed.

Evaluation

As known all tests cases are made to evaluate a theme and detect errors, limitations and adjust necessary to prove the viability of a theme. For this dissertation it was created a test case scenario, a smart home, with two laptops facing opposite directions so that the cameras could not overlap theirs detection range. Three user from different ages (18, 54, 76) were also selected and from both genders to test the test case working. The idea behind this test case to be used as a virtual butler inside and smart house was explained individually to each user.

5.1 Evaluation of the Virtual Butler Movements

One of the principal functionalities of the virtual butler is its capacity to move through the local network following the user location in the house. For that, as explained above, it was used a camera to detect the presence of the user. In the test scenario that was used to test the prototype two computers were used back to back for the cameras range could not overlap and having the prototype running the users were asked to change their position from the front of one computer to the front of the other. As expected, the prototype moves from one computer to another following the users position. With this test it was possible to verify that the usage of a simple camera despite working still has some flaws, like the luminosity and the fact that it does not distinguish people from animals. However, it is what is required to making the agent change their location following the users.

5.2 Evaluation of the Speech Interaction Response

Using the test scenario described above it was also tested the interaction with the speech system used in the prototype. The speech system could answer to two questions in English:

- What time is it?
- What date is today?

For each of these questions the prototype would answer with an synthesized voice answering

with the respective time or date. The speech system also accepts the addition and removal of events, actions possible through the commands (in English):

- Add event
- Remove Event

In both cases it was possible to conclude that the synthesizer works perfectly at the user commands, however, the speech recognition needs more work and/or better hardware because with the built in microphone of the laptops used detected a lot of the environment sounds which sometimes leads to similar sounds and of course actions not needed and out of time. Despite the hardware there is some work that could be done in the speech recognition algorithms in order to make the recognition more accurate possible.

5.3 Evaluation of the Abnormal Situations

This prototype is for users of all ages and with all sort of handicaps, and for that reason it must be prepared for abnormal situations. In the prototype it was configured one abnormal situation, as when the user is idle without interaction for more than fifteen minutes the prototype asks “are you ok?” three times, each 30 seconds. If, after the three times, the user does not answer back, the prototype shows the message “DANGER” in the screen. In the test scenario the action worked as a single task; however, when used as a real scenario, the functionality showed that it needs to be adjusted and there is the need to receive information from other sensors besides the user voice to detect abnormalities, otherwise the user will be constantly asked if he/she is ok if he/she stays in silence for more than fifteen minutes. Another abnormal situation that the prototype has parameterized is the question “Can you repeat?” that is used every time the prototype does not understand the voice commands. In this case it was surprisingly to perceive that a simple accent could make a command unrecognizable for the prototype.

5.4 Opinion of the Test Users

To test the prototype and once the test scenario wasn’t a real implementation in a house, three test users already mentioned in this chapter were selected. After they have experienced the usage of the prototype, they were asked 3 questions.

Question: What do you think about the prototype and its utility inside the house?

Answer1 (18): "It recognizes very few voice commands but it´s cool that it moves with me through the house. The plug-ins are great for making it speak more languages."
Answer2 (54): "It´s still very limited in the voice recognition but the feature of programming things is useful I could program the schedule of my mother medication that has Alzheimer.
Answer3 (76): "Ok, I recognize that this is a prototype and it is still in and embryonic state, but despite that, I like the fact that I could have a personal assistant with me all the time; however, I am old and it could be difficult for old men of my age get used to this modern stuff."

With these answers it is possible to understand that the prototype is useful despite the need to improve its dictionary of recognition commands. It is also possible to understand that, for different ages the needs change and, of course, the old ages will be the more difficult to convince to use these systems despite being the age group that has more needs and could benefit more.

Question: Do you have any suggestion of tasks that the prototype could do?
Answer1 (18): "Yes it could change the TV channel and program the recordings."
Answer2 (54): "No, nothing in particular, but the process of adding events is boring and time consuming, it would be nice if I could create the events in the computer and sync it with my agenda."
Answer3 (76): "Yes it could be more like a person, that way I could always talk and have an answer."

As showed, there are a great number of features that could be added but it is possible to agree that the major feature that can be included is a better integration in the house and the home environment itself.

Question: Would you use the prototype in the daily tasks?
Answer1 (18): "Yes no doubt."
Answer2 (54): "Yes, it would be a good help in the management of the house and its tasks and also it is better to have a friendly reminder of something than an irritating alarm on the smartphone".
Answer3 (76): "I don´t know, maybe but it would be difficult to adapt."

With this last question it is possible to understand that the level of acceptance depends on the age group and that, for each one of the age groups, the features must receive adjustments to better fulfil their tasks.

Conclusion and Future Work

In this work it was implemented a virtual butler based on mobile agents capable of speech recognition and speech synthesizing using a local network in the controlled environment of a (smart) house. It was a hard journey to accomplish the present work for various reasons, including the lack of or outdated works and documentation on the subject of mobile agents. Thus, the topic of mobile agents being old and somehow relinquished to second plan for some years it could be used as an alternative to the actual concepts.

The work developed shows that mobile agent concept could be useful in some environments and that with some work and improvements could present itself as a viable solution for personal assistant software implementations, in houses, stores or offices. As demonstrated it is possible to integrate mobile agents with other software in order to create a simple interaction with the users and simple addition and removal of features, which are good assets.

Despite de success in these implementations, there is still a lot of work that could be done in the future in order to improve the project. Once it already supports plug-in additions to improve its speech interaction with the users, a plug-in store could be developed to offer the users updates and new interactions. For example, a user with some degree of deficiency would probably need a different kind of interactions as well as old people. This plug-in store, working like Play Store®, could allow free and paid plug-ins in order to generate incomings. At last, it would be necessary a continuous work to improve the integration with sensors and obviously with mobile devices to make this easier to implement with lower costs.

Despite not being the objective of this dissertation, the great vantage of a butler based on mobile agents is his own learning ability, that always travels with him and that in the limit having him accompanies the user out of the house and makes use of the acquired knowledge and habits that he learned so far.

In short, mobile agents subject works and could be very useful when applied to solve concrete problems, and with the actual knowledge and infrastructures, with some work, this concept could be reignited and accomplish great and useful solutions.

Bibliography

[1] P. Braun and W. Rossak, Mobile Agents: Basic Concepts, Mobility Models and the Tracy Toolkit, San Francisco (USA): Morgan Kaufmann, 2005.

[2] R. Trillo, S. Ilarri and E. Mena, "Comparison and Performance Evaluation of Mobile Agent Platforms," in *Third International Conference on Autonomic and Autonomous Systems*, Athens (Greece), 2007.

[3] A. Silva and J. Delgado, "Agentes de Software: Conceitos e Tecnologias," Lisbon(Portugal), 1997.

[4] L. C. G. Rista, "Agentes Móveis - Seu uso no auxílio para a avaliação de desenpenho em um ambiente distribuido," Santa Catarina (Brazil).

[5] P. M. Reddy, "Mobile Agents: Intelligent Assistants on the Internet," India, 2002.

[6] V. H. Barros, "Sistemas Baseados em Agentes: Agentes Móveis," Porto (Portugal), 2008.

[7] C. G. Harrison, D. M. Chess and A. Kershenbaum, "Mobile Agents: Are they a good idea?," New York (USA), 1995.

[8] Y. Singh, K. Gulati and S. Niranjan, "Dimensions and Issues of Mobile Agent Technology," *International Journal of Artificial Intelligence & Applications (IJAIA),* vol. 3, 2012.

[9] W. Vieira and L. M. Camarinha-Matos, "Agentes Móveis na Operação Remota," Lisbon (Portugal), 2000.

[10] D. Horvat, D. Cvetković, V. Milutinović, P. Kočović and V. Kovačević, "Mobile Agents and Java Mobile Agents Toolkits," in *Proceedings of the 33rd Hawaii International Conference on System Sciences*, Hawaii (USA), 2000.

[11] F. Bellifemine, G. Caire, A. Poggi and G. Rimassa, "JADE A White Paper," 2003.

[12] "Pordata," [Online]. Available: http://www.pordata.pt/Portugal/Indicadores+de+envelhecimento-526. [Accessed 4 October 2014].

[13] R. S. Gray, "Agent Tcl: A transportable agent system," in *Proceedings of the Fourth Annual Tcl/Tk*

Workshop, Hanover, New Hampshire, 1995.

[14] C. Consel and L. Réveillère, “A programmable client-server model: Robust extensibility via DSLs,” in *Proceedings of the 18th IEEE International Conference on Automated Software Engineering*, Montreal, Canada, 2003.

Appendices and Attachments

Attachment 1

For this project, were used as hardware two laptop computers with very common characteristics that can be encountered easily on the market, with an average performance as shown in **Erro! A origem da referência não foi encontrada.**.

Characteristic	Computer A	Computer B
Model	Sony VAIO VGN-FZ21M	DELL Latitude D620
Operating System	Windows 7 Professional (32bits)	Windows 7 Professional (32bits)
Processor	Intel Core 2 Duo (T7250) 2GHz	Intel Core Solo (T2400) 1.83GHz
Memory (RAM)	2 GB (DDR2)	4 GB (DDR2)
Hard Drive	200 GB	200 GB
Graphic Card	NVIDIA GeForce 8400M GT	Intel Graphics Media Accelerator 950

Table 1 - Test case hardware characteristics.

Attachment 2

This attachment explains all the installation process of the chosen software used in this project.

For the project were needed:

- Java JDK
- JADE platform
- Netbeans IDE
- CMUSphinx4

Java JDK can be obtained at the Oracle site (http://www.oracle.com/technetwork/java/javase/downloads/index.html). After the download

of the JDK it must be installed using the installation wizard of the executable. Once the installation concluded it was necessary to check if the path for the installation folder had been added to the property path in the environment variables of the system, because it is this property that allows java to compile and run files. To check if the java JDK is correctly installed open command-line and run the command "java -version".

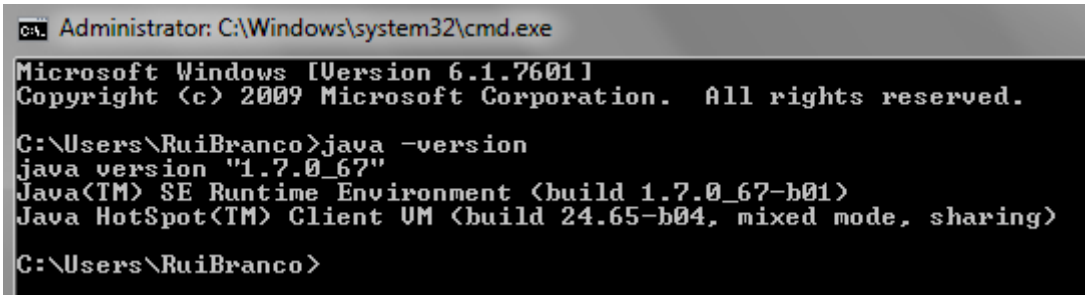

Figure 15 - Check if Java is installed and its version.

As shown in the above image the installation was successful and the current version of Java is version 1.7.0.67.

Having Java installed which is the base of all the project due the fact that is the main language and libraries, it must be downloaded JADE Platform from its own site (http://jade.tilab.com/download/jade/). JADE is downloaded as a package that has to be extracted, and within the package are examples, documentation and, of course, the library "jade.jar", which will be later added to the project.

Next software to be downloaded is CMUSphinx4 from the project site (http://cmusphinx.sourceforge.net/wiki/download/). This software also came in a package that had to be extracted and the content was a Java project that could be imported to an IDE as an example project, but instead all that is needed are four of the libraries that this package had.

- Jsapi.jar
- Sphinx4.jar
- WSJ_8gau_13dCep_16k_40mel_130Hz_6800Hz.jar
- TIDIGITS_8gau_13dCep_16k_40mel_130Hz_6800Hz.jar

These libraries will later be added to the main project.

At last the download of the IDE only to facilitate the integration and use of the different libraries in the same project and the coding and debugging process. For that choice were netbeans IDE for Java from the netbeans project site (https://netbeans.org/downloads/).
In the end all this software and libraries downloaded were included in one netbeans project where all the necessary code will be developed.

Attachment 3

After the project creation and all the libraries added, to run the project it is necessary to configure some parameters in the project run properties.

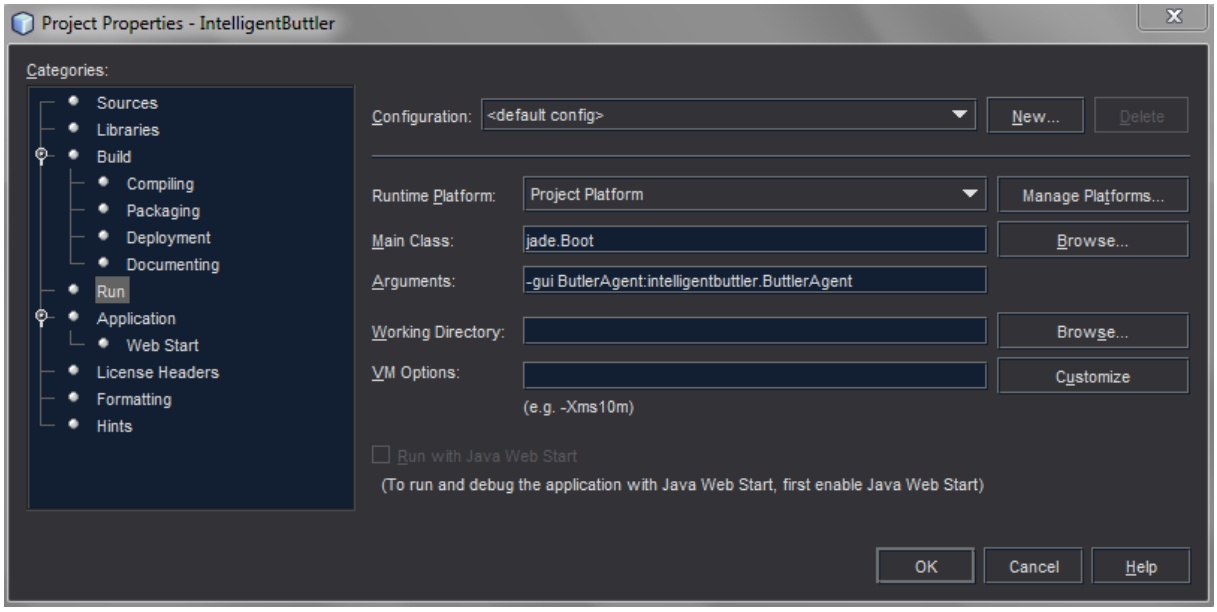

Figure 16 - Project properties.

To run this project it was necessary to define the main class of the project as "Jade.Boot". This is a class part of the library of JADE and this configuration will result in the launch of the JADE platform which is required to receive and send the mobile agents. This main class still receive some arguments as shown in the image (Figure 16), "-gui ButlerAgent:intelligentbuttler.ButtlerAgent". This argument is interpreted by the class in parts:

- -gui – tells to the library that the JADE platform will be launched with graphical interface.
- ButlerAgent – is the alias that will be shown in the JADE platform for the agent.
- Intelligentbuttler – is the package of the project that contains the agent to be launched.

- ButtlerAgent – is the name of the class in the package that will be launched.

Being everything configured, the project is launched and the graphical interface of JADE and the graphical interface of the agent are shown simultaneously as presented in the figures below

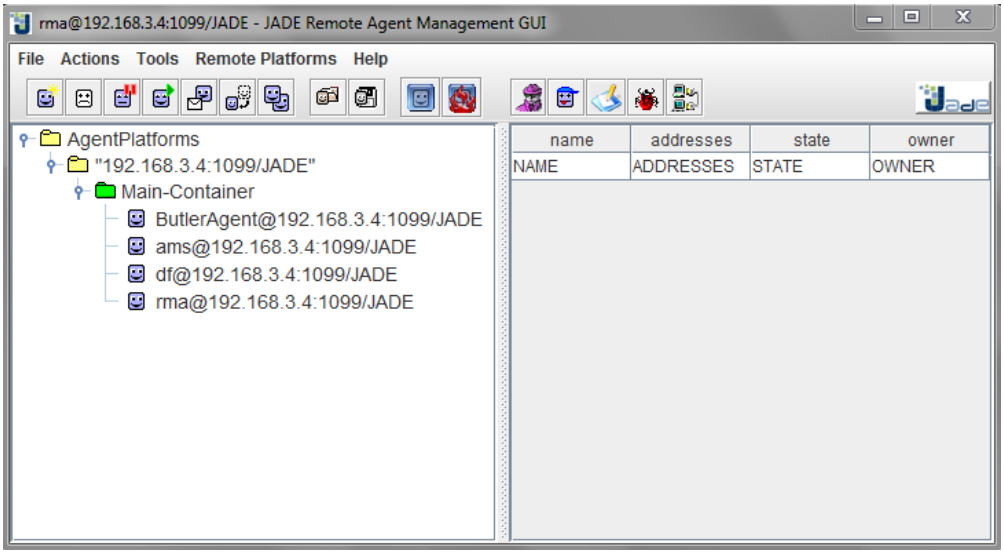

Figure 17 - JADE Platform interface.

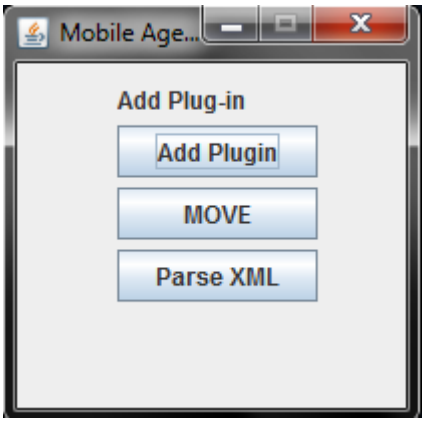

Figure 18 - Agent interface.

At this point and having a JADE platform (Figure 17) running in another machine in the same network, the agent can already migrate between machines.