DEVELOPMENT OF AN INTELLIGENT AGENT FOR SPEECH RECOGNITION AND TRANSLATION

(PEMBANGUNAN AGEN PINTAR UNTUK PENGECAMAN SUARA DAN PENTERJEMAHAN)

ABD MANAN BIN AHMAD

RESEARCH VOT NO:
74076

Jabatan Kejuruteraan Perisian
Fakulti Sains Komputer dan Sistem Maklumat
Universiti Teknologi Malaysia

2006

UNIVERSITI TEKNOLOGI MALAYSIA

BORANG PENGESAHAN
LAPORAN AKHIR PENYELIDIKAN

TAJUK PROJEK : **DEVELOPMENT OF AN INTELLIGENT AGENT FOR SPEECH RECOGNITION AND TRANSLATION**

Saya **ABD MANAN BIN AHMAD**
(HURUF BESAR)

Mengaku membenarkan **Laporan Akhir Penyelidikan** ini disimpan di Perpustakaan Universiti Teknologi Malaysia dengan syarat-syarat kegunaan seperti berikut :

1. Laporan Akhir Penyelidikan ini adalah hakmilik Universiti Teknologi Malaysia.
2. Perpustakaan Universiti Teknologi Malaysia dibenarkan membuat salinan untuk tujuan rujukan sahaja.
3. Perpustakaan dibenarkan membuat penjualan salinan Laporan Akhir Penyelidikan ini bagi kategori TIDAK TERHAD.
4. * Sila tandakan (/)

[ ]	SULIT	(Mengandungi maklumat yang berdarjah keselamatan atau Kepentingan Malaysia seperti yang termaktub di dalam AKTA RAHSIA RASMI 1972).
[ ]	TERHAD	(Mengandungi maklumat TERHAD yang telah ditentukan oleh Organisasi/badan di mana penyelidikan dijalankan).
[X]	TIDAK TERHAD	

TANDATANGAN KETUA PENYELIDIK
PROF. MADYA ABD MANAN AHMAD
Ketua Projek Vot: 74076
Fakulti Sains Komputer & Sistem Maklumat
Universiti Teknologi Malaysia Skudai.
Nama & Cop Ketua Penyelidik

Tarikh : 1. 11. 2006

CATATAN : **Jika Laporan Akhir Penyelidikan ini SULIT atau TERHAD, sila lampirkan surat daripada pihak berkuasa/ organisasi berkenaan dengan menyatakan sekali sebab dan tempoh laporan ini perlu dikelaskan*

ABSTRACT

Nowadays, speech researchers and software engineer make an effort to integrate speech function into internet applications. Intelligent Agent for Speech Recognition and Translation prototype has been build to achieve that purposed. This prototype also proposed a new framework for speech recognition, word translation and mobile agent technology. Intelligent Agent for Speech Recognition and Translation is a system that recognized human voice in spoken English, then translate English word to Malay word. Generally, this prototype is a merge of two areas, speech recognition and word translation and using mobile agent technology to integrate them. This prototype is a system executed in Local Area Network (LAN) environments and also called Distributed Speech Recognition (DSR) system. DSR system divided into two parts, front-end processing (feature extractions) and back-end processing (recognition). There is a server which connected with a number of client devices. At the client side, the system will received human voice in spoken English and front-end processing will be done here. Then the feature files will be transmits to the server side which executed the back-end processing. When the voice has been recognized, the result will be translated into Malay word in text. After translation process is done, the Malay word in text will be send back to the client computer. All these processes will be handled and monitor by agent. With contribution from mobile agent technology, the efficiency of speech recognition server can be improved compared with traditional distributed speech recognition server in term of time processing. By using mobile agent technology, speech recognition server can perform their task as parallel processing. Beside that, this approach also can reduce network traffic while handling the transfer data.

ABSTRAK

Kini, penyelidik suara dan juga jurutera perisian telah berusaha untuk mengintegrasikan fungsi-fungsi suara ke dalam aplikasi internet. Prototaip Agen Pintar untuk Pengecaman Suara dan Penterjemahan telah dibangunkan untuk bersama-sama memenuhi tujuan tersebut. Prototaip ini juga mengusulkan kerangka baru untuk pengecaman suara, penterjemahan perkataan dan juga teknologi agen bergerak. Prototaip ini adalah satu sistem yang mengecam suara manusia dalam sebutan bahasa Inggeris, kemudian menterjemahkan perkataan Inggeris tersebut ke dalam perkataan bahasa Melayu. Umumnya, prototaip ini adalah gabungan daripada dua bidang iaitu pengecaman suara dan juga penterjemahan perkataan serta menggunakan pendekatan teknologi agen bergerak untuk proses pengintegrasian. Prototaip ini dilarikan pada persekitaran Rangkaian Kawasa Setempat (LAN) serta boleh dinamakan sebagai sistem Pengecaman Suara Teragih (PST). Sistem PST dibahagikan kepada dua bahagian utama iaitu pemprosesan bahagian hadapan dan pemprosesan bahagian belakang. Terdapat satu komputer pelayan yang bersambung dengan sejumlah alat peranti pelanggan. Pada bahagian pelanggan, sistem ini akan menerima suara pelanggan dalam percakapan Inggeris dan pemprosesan bahagian hadapan akan dilaksanakan di bahagian alat peranti pelanggan. Kemudian, fail pengekstrakan ciri akan dihantar ke bahagian pelayan dimana bahagian pemprosesan belakang akan dilaksanakan. Setelah, proses pengecaman di lakukan ke atas suara tadi berjaya, hasil keputusan tersebut akan di terjemahkan ke dalam bahasa Melayu dalam bentuk teks. Setelah proses penterjemahan selesai, hasil keputusan akan di hantar ke alat peranti pelanggan. Proses-proses yang berlaku akan di kawal selia dan di awasi oleh agen. Dengan sumbangan daripada teknologi agen, tahap prestasi

pengecaman suara pelayan adalah lebih baik jika dibandingkan dengan pengecaman suara teragih tradisional dalam konteks pemprosessan masa. Secara kesimpulannya, dengan menggunakan pendekatan teknologi agen, pengecaman suara pelayan boleh melaksanakan tugas-tugasnya secara pemprosesan selari. Selain itu, pendekatan ini juga mampu untuk mengurangkan kesesakan traffik semasa pengendalian penghantaran data.

CONTENTS

CHAPTER II LITERITURE REVIEW

CHAPTER III METHODOLOGY

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ASR	-	Automatic Speech Recognition
AMS	-	Agent Management System
AUML	-	Agent Oriented Unified Modeling Language
CORBA	-	Common Object Request Broker Architecture
DCT	-	Discrete Cousine Transform
DF	-	Directory Facilitator
DSR	-	Distributed Speech Recognition
FFT	-	Fast Fourier Transform
HMM	-	Hidden Markov Models
IP	-	Internet Protocol
Jade	-	Java Agent Development
JDK	-	Java Development Kit
LAN	-	Local Area Network
LPC	-	Linear Predictive Coding
MaSE	-	Multiagent Systems Engineering
MESSAGE	-	Methodology For Engineering Systems Of Software Agent
WWW	-	World Wide Web

LIST OF APPENDICE

CHAPTER I

PROJECT OVERVIEW

1.1 Introduction

Intelligent networking is helping to advance the sharp rise and wide exploitation of agent technologies. Mobile agent technology, a profound revolution in computer software technology, is a brand new computing technique promoted for solving the complicated and dynamic applications of distributed intelligence. Followed by the rapid development of artificial intelligence and computer networking, especially for the popular applications of Internet and related technologies in recent years, the network has become a basic platform for people to publish and obtain information. Under this background, computing is no longer limited to few servers in the network, but all computers are required to get involved into this environment.

The current distributed computing model typically utilizes Remote Procedure Call (RPC), process immigration and Client/Server architecture, but all these computing models have limitations. The biggest one is that all the nodes involved in computing have to be available in the network at the same time during the interaction. If some required resources are not accessible, the whole computing process will fail.

1.2 Background

This research is about Development of an Intelligent Agent for Speech Recognition and Translation. Basically, input for this system is voice in English sound and the output is Malay text. Therefore, this system also can call it as English – Malay word translation system based on voice input and the result is in text. This system support utterance of technical terms based on Information Technology domain. This research involved three main parts: mobile agent, speech recognition and word translation. Generally, this system runs on Local Area Network (LAN) environments and it also called distributed speech recognition. User will record voice at the client-side while at the server-side, recognition and translation process will be done and the result will be send to the user. Translation part was based on translate word from English word to a Malay word. Those words were based on 10 confusable technical terms – in the Information Technology domain.

This system was developed based on mobile agent technology on networks to support distributed speech recognition system and English – Malay word translation as well. Agent development was based on Java Agent DEvelopment (JADE) framework, which contained management agent for both speech process and word translation process. While, speech recognition processed was based on CMU Sphinx4 Speech Recognition engine. MySQL 5.1 also has been applied to support prototype database, especially used in translation parts. Further description about these parts will be explains in the next section.

1.3 Problem Statement

Based on Antonio Cardenal-Lopez et. al (2004), the increasing use of both the Internet and Automatic Speech Recognition (ASR) systems makes Internet-based Distributed Speech Recognition (DSR) services very attractive. In DSR, speech feature

vectors are obtained at the client side and transmitted to the remote server for recognition. These services are based on a client-server architecture:

i. The client device quantizes and packetizes the speech features and transmits them over the communication channel to a remote ASR server.
ii. The remote ASR server performs the speech recognition task.

For this research, we also applied both of the services but not exactly based on client-server architecture. It is because client-server implementation was highly dependent on sufficient network bandwidth (Robert S. Gray, 2001). However, for running distributed speech recognition system, network bandwidth was not usually in a good condition. Differs from mobile agent technology, agent implementations saved network bandwidth at the expense of increased server computation. In this situation, we believed by using mobile agent technology to support distributed speech recognition system was an excellent idea. We also applied mobile agent parallel processing to support remote server computation to recognize the features files.

Based on experiments done by Robert S. Gray (2001), mobile agents can be beneficial in situations with low network bandwidth and plentiful server capacity. Indeed, in many environments it is easier to add more server capacity than to add network capacity. With those problems, we believed this project, Development of Intelligent Agent for Speech Recognition and Translation can solve the problems either to improve the accuracy of speech recognition performance or to come out with new architecture of mobile agent technology in DSR systems.

1.4 Aim

The aim of this research is to develop agent-based distributed speech recognition and English – Malay word translation system for information technology term, input by

spoken English and output as a Malay in text. In addition, the systems run on Local Area Network (LAN) environment.

1.5 Project Objectives

The purposed of this research was to develop agent-based distributed speech recognition and translation. The input was spoken English and the output was Malay word in text. The systems execute on LAN environment. There are several objectives for this project. There are:

i. To identify a framework of human speech.
ii. To develop an architecture of an intelligent agent for speech recognition.
iii. To develop a prototype of a speech recognition system for the improvement of an existing online ‘Sistem Penterjemahan Istilah Teknologi Maklumat’ from Dewan Bahasa dan Pustaka.

This research also wants to know the efficiency of mobile agent technology in distributed speech recognition environment. These efficiencies can be look deeper in two ways:

i. How mobile agent technology is competent to support distributed speech recognition application through network compare with client-server method?
ii. How mobile agent technology can optimize speech recognition engine server processes for each request and response from client devices?

Therefore, in the next chapter will discuss about these two issues that been highlighted in detail.

1.6 Scope Project

Development of Intelligent Agent for Speech Recognition and Translation prototype is based on several conditions and scopes. These kinds of scopes are lists below:

i. The prototype runs on Local Area Network (LAN) environments.
ii. The translation process is based on English – Malay word translations.
iii. The prototype used CMU Sphinx4 speech recognition engine as speech recognition engine server.
iv. The prototype used Java Agent Development Framework to develop agent functions.

1.7 Thesis Arrangement

i. Project Overview.

First chapter that should be done is Project Overview. This chapter describes project overview: Development of an Intelligent Agent for Speech Recognition and Translation. It contains of general overview about three main parts in this research project: Mobile Agent, Speech Recognition Engine and Word Translation, our research target and project objectives.

ii. Literature Review

After project overview has been defined, our project's problem definitions and literature review will be done. After problem definition and literature review done, the suitable methodology for this research project will be describes further in chapter 3.

iii. Methodology

In this chapter, it will describe about the methodology, method and techniques that will be use in this system development life cycle. This chapter will be divided into two parts: project development methodology and system prototype development methodology.

iv. Project Design and Implementation

In this chapter, it consists of two main process development : mobile agent system development and speech recognition engine development. This chapter will describe a model to develop the system and the methodology based on agent development process and also methodology based on CMU Sphinx4 Speech recognition engine.

v. Result and Conclusion

This chapter will describe our result that we get from our system, speech recognition engine and word translation. After that, we will make a conclusion about the whole process development that we have done.

CHAPTER II

LITERATUR REVIEW

2.1 Introduction

This research project developed to captures voice in spoken English and recognized the voice in text. After some recognition processes is done, the recognized text will be translate from English word to Malay word in text. This research project consists of three main parts: mobile agent, speech recognition and word translation. Mobile agent technology is the main part in this research project. Mobile agent is used to control and manage the interaction and the processes around the system. Mobile agent is the middleware between speech recognition engine and word translation engine. Speech recognition engine is responsible to recognize the human speech and word translation engine responsible to translate from source word (English) to target word (Malay) in text. The further description about those parts will be describes in the next sections.

2.2 Mobile Agent

Mobility is an orthogonal property of agents. That is, not all agents are mobile. An agent can just sit there and communicate with its surroundings by conventional means, such as various forms of remote procedure calling and messaging. We call agent that do not or cannot move *stationary agent.*

Stationary Agent-A stationary agent executes only on the system where it begins execution. If it needs information that is not on that system or needs to interact with an agent on a different system, it typically uses a communication mechanism such as remote procedure calling (RPC) (Danny B. Lange and Mitsuru Oshima, 1998).

In contrast, mobile agent is not bound to the system where it begins execution. The mobile agent is free to travel among the hosts in the network. Created in one execution environment, it can transport its *state* and *code* with it to another execution environment in the network, where it resumes execution. By the term state, it typically means the attribute values of the agent that help it determine what to do when it resumes execution at its destination. By the term *code*, in an object-oriented context, the class node necessary for the agent to execute.

Mobile Agent - A mobile agent is not bound to the system where it begins execution. It has the unique ability to transport itself from one system in a network to another. The ability to travel allows a mobile agent to move to a system that contains an object with which the agent wants to interact and then to take advantage of being in the same host or networks as the object (Danny B. Lange and Mitsuru Oshima, 1998).

An agent is usually defined as an independent software program that runs on behalf of a network user. It can be characterized as having more or less intelligence and it has the ability to learn. Mobile agents add to regular agents the capability of traveling to multiple locations in the network, by saving their state and restoring it in the new host. As they travel, they work on behalf of the user, such as collecting information or delivering

requests (Herve Paulino, 2002). This mobility greatly enhances the productivity of each computing element in the network and creates a powerful computing environment. Mobile agents require a software infrastructure that provides them security and data protection. This infrastructure includes protocols, rules for safe mobility and, directions and directories with information about all available hosts.

The use of mobile agents for distributed applications has several potential benefits (Stefan Fünfrocken and Friedemann Mattern, 1999). Those benefits are as follows:

i. Asynchronous task execution: while the agent acts on behalf of the client on a remote site, the client may perform other task.
ii. More dynamic: It is not necessary to install a specific procedure at a server before-hand and to anticipate specific service request types; a client or a service provider may send different types of agents to a server without the need to reconfigure the server.
iii. Reduced communication bandwidth: If vast amounts of server data have to be processed and if only a few relevant pieces of information have to be filtered out, it is more economical to transfer the computation to the data than the data to the computation.

Generally, based on those advantages of mobile agent above, they can reduce network traffic, provide an effective means of overcoming network latency, and perhaps most importantly, through their ability to operate asynchronously and autonomously of the process that created them. Mobile agent also can help programmers to construct more robust and fault-tolerant applications (Yariv Aridor and Danny B. Lange, 1998).

2.2.1 Network Computing Paradigm

Mobile agents provide a powerful, uniform paradigm for network computing. Mobile agents can revolutionize the design and development of distributed systems. There are several programming paradigm for distributed computing such as client-server, code-on-demand and mobile agents (Danny B. Lange and Mitsuru Oshima, 1998).

i. Client-Server Paradigm

In the client-server paradigm (see figure), a server advertises a set of services that provide access to some resources (such as database). The code that implements these services is hosted locally by the server. The server holds the *know-how*. Finally, it is the server itself that executes the services and thus has the processor capability. If the client is interested in accessing a resource hosted by the server, the client will simply use one or more of the services provided by the server. Clients need some "intelligence" to decide which of the services it should use. The server has it all: the *know-how*, resource and processor. So far, most distributed systems have been based on this paradigm. We see it supported by a wide range of technologies such as remote procedure calling (RPC), object request broker (CORBA) and Java remote method invocation (RMI).

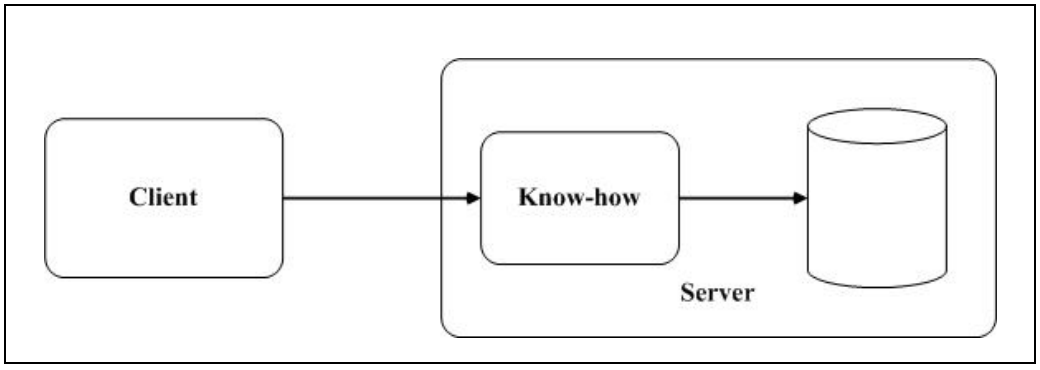

Figure 2.1: Client-Server Paradigm

ii. Code-On-Demand Paradigm

According to the code-on-demand paradigm (see figure), you first get the *know-how* when you need it. Suppose that the client initially is unable to execute its task because of a lack of code (*know-how*). Fortunately, a host in a network provides the needed code. Once the code is received by the client, the computation is carried out in the client. The client holds the processor capability as well as the local resource. In contrast to the classical client-server paradigm, the client does not need preinstalled code because all the necessary code will be downloaded. The client has the resource and processor, and the host has the *know-how*. Java applets and servlets are excellent practical examples of this paradigm. Applets get downloaded in web browsers and execute locally, whereas servlets get uploaded ti remote Web servers and execute there.

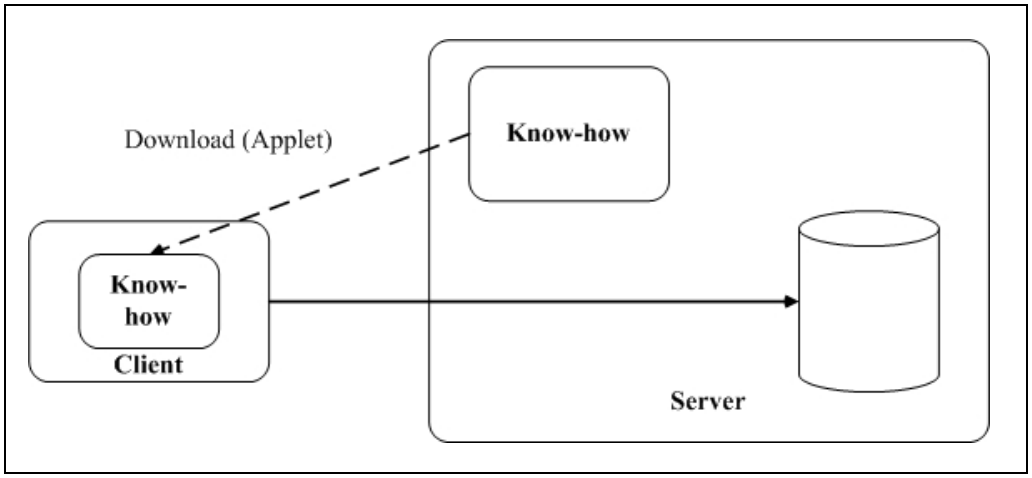

Figure 2.2: Code-On-Demand Paradigm

iii. Mobile Agent Paradigm

A key characteristic of the mobile agent paradigm (see figure) is that any host in the network is allowed a high degree of flexibility to possess any mixture of know-how, resources and processors. Its processing capability can be combined with local resources. Know-how (in the form of mobile agents) is not tied to a single host but rather is available throughout the network.

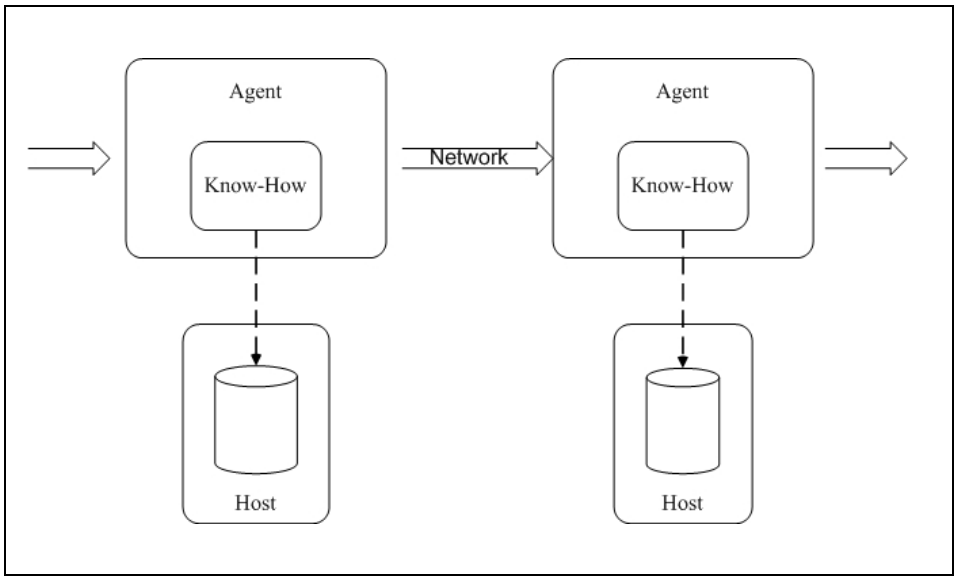

Figure 2.3: Mobile Agent Paradigm

The comparison between these three paradigms is the chronological trend toward greater flexibility. The client and server has merged and become a *host*. The applet and the servlet, while serving as client and server extenders, respectively, have been combined and improved with the emergence of mobile agents.

2.2.2 Contemporary Mobile Agent Systems

What kind of mobile agent systems are available? Fortunately, Java has generated a flood of experimental mobile agent systems. Numerous systems are currently under development, and most of them are available for evaluation on the web (Danny B. Lange and Mitsuru Oshima, 1998). Some of Java-based mobile agent systems are as follows:

i. **Aglets.** This systems, mirrors the applet model in Java. The goal was to bring the flavor of mobility to the applet. The term aglet is a portmanteau word

combining *agent* and *applet*. One of the aglet purposed is to make aglets an exercise in clean design. So that, applet programmers will appreciate the many ways in which the aglet model reflects the applet model.

ii. **Odyssey.** General Magic Inc. invented the mobile agent and created Telescript, the first commercial mobile agent system. Based on proprietary language and network architecture, Telescript had a short life. In response to the popularity of the internet and later the steamroller success of the Java language, General Magic decided to re-implement the mobile agent paradigm in its Java-based Odyssey. This system effectively implements the Telecsript concepts in the shape of Java calluses. The result is a Java class library that enables developers to create their own mobile agent applications.

iii. **Concordia.** Mitsubishi's Concordia is a framework got the development and management of mobile agent applications that extend to any system supporting Java. Concordia consists of multiple components, all written in Java, which are combined to provide a complete environment for distributed applications. A Concordia system, at its simplest, is made up of a standard JavaVM, a server and a set of agents.

iv. **Voyager.** ObjectSpace's Voyager is a platform for agent-enhanced distributed computing in Java. While Voyager provides an extensive set of abject messaging capabilities, it also allows objects to move as agents in the network. You can say that the voyager combines the properties of a Java-based object request broker with those of a mobile agent system. In this way, Voyager allows Java programmers to create network applications using both traditional and agent-enhanced distributed application techniques.

2.2.3 Agent Design Pattern

Based on (Emerson Ferreira et. al, 2003), a different approach to increase the usage of mobile agent in real applications is the definition of design patterns. The basic

idea of design patterns is to define general solution models for common problems found in a given context. In fact, some problems can be faced many times during different phases of the development process. It would be worthwhile if good solutions for these problems could be reused instead of the necessity to develop a new one from scratch. Design patterns make easier the development of applications, increasing flexibility and promoting reuse.

The advantage of using mobile agent design patterns in JADE framework has been shown in (Emerson Ferreira et. al, 2003). The use of agent design patterns has generally increased due to the advantages they can bring to applications development, like reuse and a better understanding of their project. These advantages can also be obtained when developing mobile agent-based applications by using mobile agent design patterns. These patterns present solutions that can be reused, avoiding loss of time and effort to investigate problems that have already been solved.

The patterns so far can be divided into three classes: traveling, task and interaction. This classification scheme makes it easier to understand the domain and application of each pattern, to distinguish different patterns, and to discover new patterns. We describe the three classes of patterns as well as the patterns in each class.

2.2.3.1 Traveling Pattern

The itinerary pattern is an example of a traveling pattern that is concerned with routing among multiple destinations. An itinerary maintains a list of destinations, defines a routine schema, handle special cases such as what to do if the destination does not exist, and always to know where to go next. Objectifying the itinerary allows you to save it and reuse it later in much the same way that you save URLs as bookmarks.

The ticket pattern, an enriched version of the concept of a URL, embodies requirements concerning quality of service, permission or other data. For example, it can include time-out information for dispatching the agent to a remote host. Thus, instead of naively trying to dispatch to a disconnected host forever, the agent now has the necessary information to make reasonable decisions while traveling.

2.2.3.2 Task Pattern

Task patterns are concerned with the breakdown of task and how these tasks are delegated to one or more agent. In general, task can be dynamically assigned to general purpose agents. Furthermore, a given task can be accomplished either by a single agent or by multiple agent working in parallel and cooperating to accomplish it (such as in the case of a parallel search).

The fundamental task pattern is Master-Slave pattern, which allow the master agent to delegate a task to a slave agent. The slave agent moves to a destination host, performs the assigned task, and sends back the result of that task. The more complex *Plan* pattern adopts a workflow concept to organize multiple tasks to be performed in sequence or in parallel by multiple agents. The plan encapsulates the task flow, which is then hidden from the agent. The agent merely provides the mobility capabilities needed to perform activities at specific destinations. The plan promotes reusability of tasks, dynamic assignment of tasks to agent, and even composition of tasks. Figure 2.4 show the scenario of master-slave agent technique.

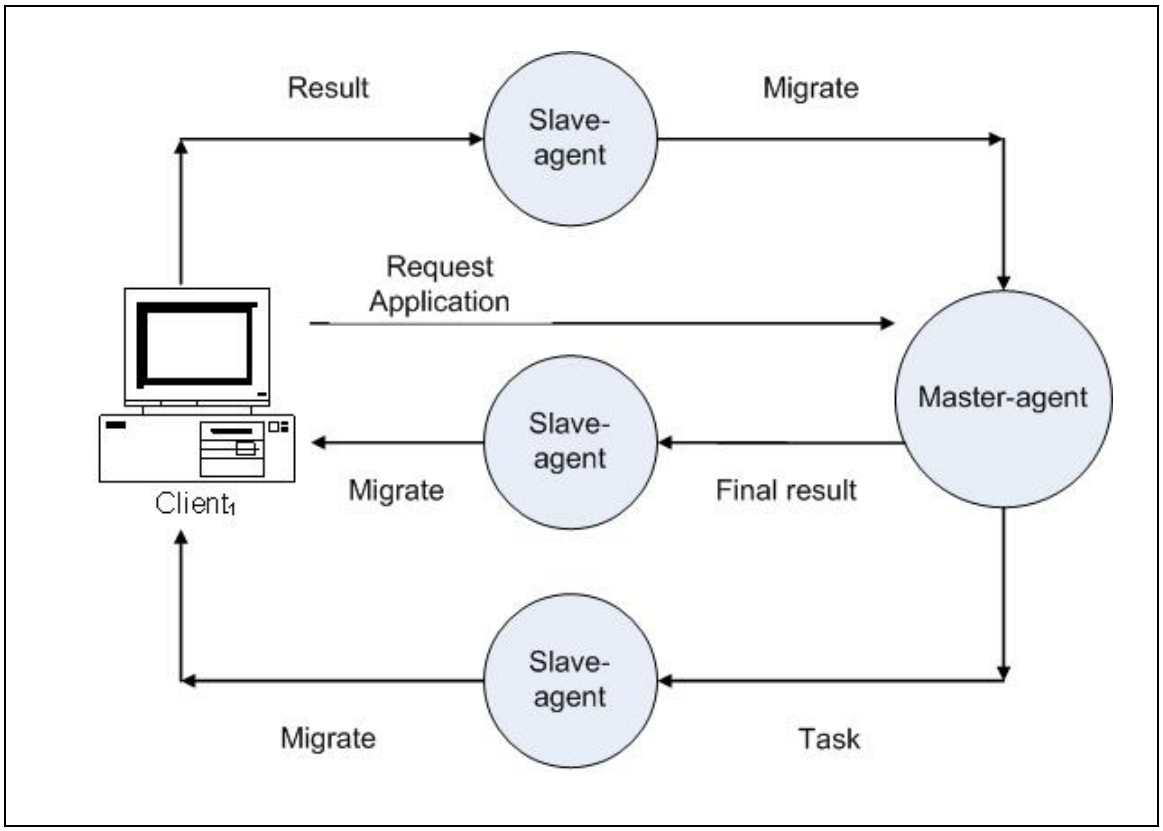

Figure 2.4 : Scenario of master-slave agent technique

2.2.3.3 Interaction Pattern

The ability of agent to communicate with one another is vital for cooperation among agents. The interaction patterns are concerned with locating agent and facilitating their interactions. The *Meeting* pattern is an interaction patterns that provides a way for two or more agent to initiate local interaction at a given host. It abstracts the synchronization in time and place that is required for local interactions. Agent can dispatched themselves to a specific destination, called a *meeting place,* where there are notified of the arrival of their counterparts and can engage in local interaction. Agent can exploit the *Locker* pattern to temporarily store data in private. In this way, they can avoid

bringing along data that for the moment are not needed. Later, agent can return and retrieve the private data stored in the locker. For example, in an agent-based purchasing system, an agent may visit a vendor's host locker before leaving the company network. The result is a reduction of network traffic and improved data confidentially.

Agent can establish remote communication by using the *Messenger* pattern, which objectifies messages in the form of agent that carry and deliver messages between agents. For example, if a slave agent wishes to report a possibly intermediate result back to the master agent, it can send the result by a messenger agent while continuing with its current task.

The Finder pattern describes a naming and locating service for agents. It is so often convenient to assign a symbolic (meaningful) name to an agent in order to locate it later. For example, an information-gathering agent may continuously move in the network and other agents may from tome to time wish to retrieve updates from the information-gathering agent without actually knowing its present location.

Table 2.1: Agent Design Patterns

Patterns	Description
Traveling Patterns	
Itinerary	Objectifies agent's itineraries and routing among the destination.
Forwarding	Provides a way for host to forward newly arrived agents automatically to another host.
Ticket	Objectifies a destination address and encapsulates the quality of service and permissions needed to dispatch an agent to a host address and execute it there.
Task Patterns	
Master-Slave	Defines a scheme whereby a master agent can delegate a task to a slave agent.
Plan	Provides a way of defining the coordination of multiple tasks to

	be performed on multiple hosts.
Interaction Patterns	
Meeting	Provides a way for two or more agents to initiate local interaction at a given host.
Locker	Defines a private storage space for data left by an agent before it is temporarily dispatched (sent) to another destination.
Messenger	Defines a surrogate agent to carry a remote message from one agent to another.
Finder	Defines an agent that provides services for naming and locating agents with specific capabilities.

2.2.4 Parallel Processing using Mobile Agent

This Intelligent Agent for Speech Recognition and Translation prototype executed in LAN environments. There is a server waiting and answer a request from the computer client that request for speech recognition and word translation applications. This server connected with a number of computer clients through network or internet. Therefore, there is a need for the server to perform all requests from the computer client. However, there is a burden jobs for the server to completely process both parts recognition and translation process. For recognition process, there is a need to process a complex computation before the speech data can be recognized. Based on Wei Qi Zhang et. al (2000), there decided that the recognition server should use some parallel program method to support the recognition server. Their suggestions are to use some parallel program method such as multithread, variable-share and COOP (Concurrent Object-Oriented Program) to efficiently use these computers. It is because DSR server has to be constructed by multiple computers and parallel program to support multiple requests.

Wei Qi Zhang et al, (2000) also described implementation of recognition server which they used CORBA method and the core components in CORBA is SRP. The data/control flow inside SRP is shown in figure 2.5. In this figure, when a request arrives on SRP, SRP creates a new thread *Thread2* to manage the connection and receive the following feature data. When the data completely received, the handle of *Thread2* is put into *Thread* Queue, which is a buffer used to contain all the requesting thread handles. When *Thread2's* handle reaches the top of the queue, this handle is deleted from the queue if there are available engine servers. Otherwise, *Thread2's* handle has to wait.

Moderator decides which engine server is available according to a workload schedule scheme. A workload schedule scheme is typically geared to meet certain performance metrics, which include CPU efficiency of engine server, number of jobs (threads), response time of each request (user), amount of processed data, communication throughput and memory requirement.

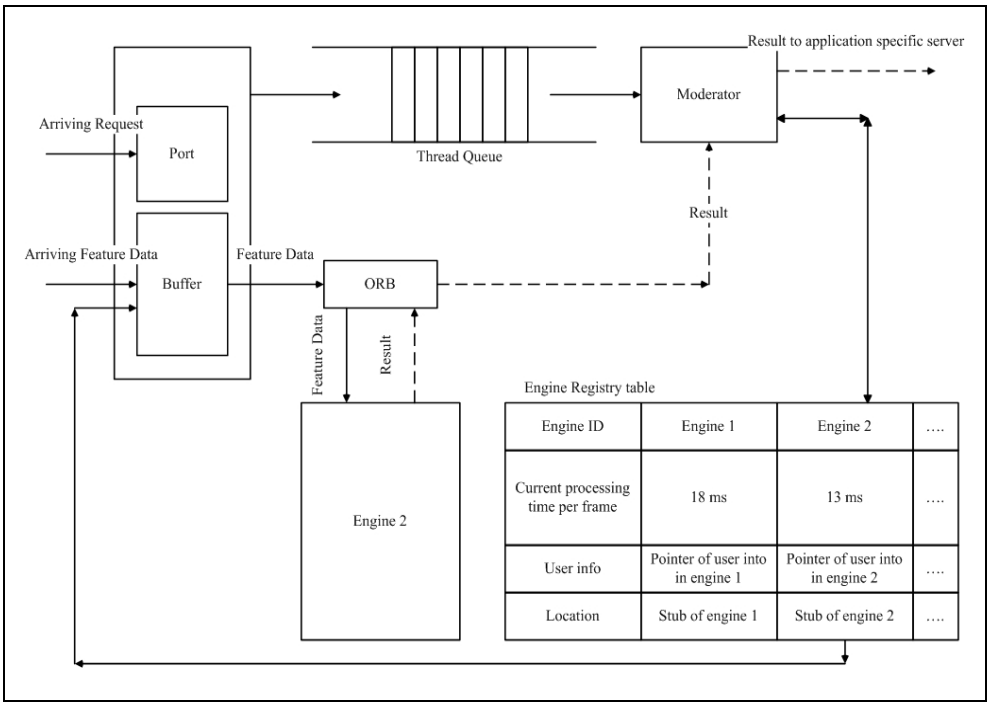

Figure 2.5: Block diagram of recognition management in SRP

This prototype intelligent agent for speech recognition and translation applied mobile agent technology to support parallel processing in speech recognition server. Mobile agent has a method that called agent design pattern. This method has been described before at section 2.1.3. This method has three classes and task pattern class has been chosen to support parallel processing in speech recognition server.

In class task pattern, it provided Master-Slave pattern, which allow the master agent to delegate a task to a slave agent. The slave agent moves to a destination host, performs the assigned task, and sends back the result of that task (Danny B. Lange and Mitsuru Oshima, 1998). Based on this pattern, a parallel processing for speech recognition server has been build and implemented. A master-slave responsible to manage, monitor, create and control each agent that executed and connected with this system. Master-slave will delegate the speech and translation process to each slave-agent that he created. For every slave-agent, they have their own thread. With that thread, they can perform the given task without have to queue.

2.2.5 Multi-Threaded Application

Multi-threaded programming is an important concept in Java networking, as networking clients and servers must often perform several different tasks at a time (for example, listening for incoming requests and responses, processing data, and updating the text or graphical user interface for the user). Multi-threading is a strategy to split tasks into manageable units (David Reilly and Michael Reilly, 2002). There are three types of multi-threaded programming:

i. Single-Threaded Programming
ii. Multiprocess Programming
iii. Multi-threaded Programming

2.2.5.1 Single-Threaded Programming

Traditional software written in procedural languages is compiled into a machine-readable format, which is called machine code. This code is read by a central processing unit (CPU), which executes programming statements one after another, in a sequential manner (see figure 2.6). The time taken to execute each statement may vary (due to the nature of the operation, such as comparing two bytes for quality or adding two number together), but until a statement is completed, no further statement will run. This is single-threaded execution.

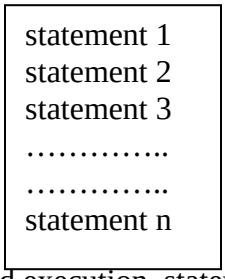

Figure 2.6 : In single-threaded execution, statements are executed sequentially

2.2.5.2 Multiprocess Programming

Multiprocess programming can be described as each application runs as a process, with memory allocated for program code and data storage (Multiple processes would run on the same machine. The operating system would allocate CPU time to each process, suspending a process when its time was up and allowing another to take its place. Sometime, a process will become blocked (waiting an I/O), or may voluntarily choose to yield it CPU time.

Multiprocesses programming can create new processes, having one part of the program performing a task while another part does something else (figure 2.7). This type

of programming is useful, as it means that work can be performed even if one part of the program becomes stalled (for example, waiting for input).

Although multiprocess programming works well, there are disadvantages to its use. First, when a process branches into two, there is overlap between the data storage of one process and another. Because two copies of data are being kept, more memory than is needed is consumed. Second, there isn't an easy way for one process to access and modify the data of another.

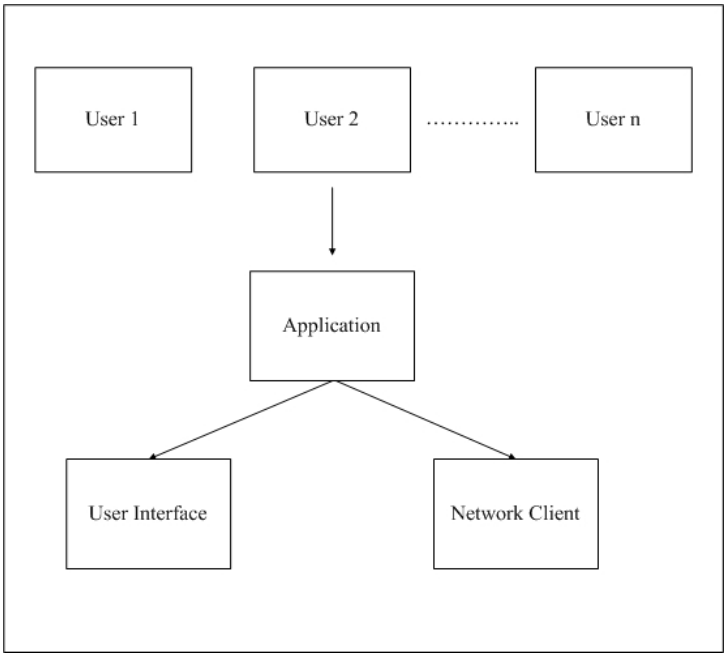

Figure 2.7: Application process can fork into two, having one or more sub-processes perform useful work

2.2.5.3 Multi-threaded Programming

Multi-threaded programming requires a different way of looking at software. Rather than executing a series of steps sequentially, tasks are executed concurrently-that

is, many tasks are performed at the same time, rather than one task having to finish before another can start. Multi-threading, also known as multiple thread of execution, allow a program to have multiple instances of itself running, while using the same shared memory space and code (David Reilly and Michael Reilly, 2002). An application can be performing many different tasks concurrently and thread may access shared data variables to work collaboratively.

2.3 Speech Recognition

Speech recognition is an alternative to traditional methods of interacting with a computer, such as textual input through a keyboard. An effective system can replace, or reduce the reliability on, standard keyboard and mouse input. This can especially assist the following:

- People who have little keyboard skills or experience, who are slow typist, or do not have the time to resources to develop keyboard skills.
- Dyslexic people, or others who have problems with character or word use and manipulation in a textual form.
- People with physical disabilities that affect either their data entry, or ability to read (and therefore check) what they have entered.

Speech and understanding voice message by human is a complex process. Factors like height, weight, sex, teeth and lips can give an impact to their speech. Voice processing by human can be simplified as below.

Processing sequence in the human auditory system.

- Fixed filter which represents the transfer from free field to eardrum and through the middle ear.
- A bank of logarithmically spread bandpass filters in cochlea.

- Dynamic compression, when mechanical energy is transformed to neural signals by the hair cells.
- Periodicity estimation at each band.
- Integration of the band-wise processed signals and further calculations. This takes place in the central nervous system (brains).

Human perception of speech starts with receiving signal by the ear. It will then pass the membrane basilar in the inner ear where the signal will be analyzed. The analyzed signal will pass to neural tranductor that convert the signal into activity signal on the auditory nerve and the brain will translate and understood the speech. Figure 2.8 show the scenario of human speech recognition process.

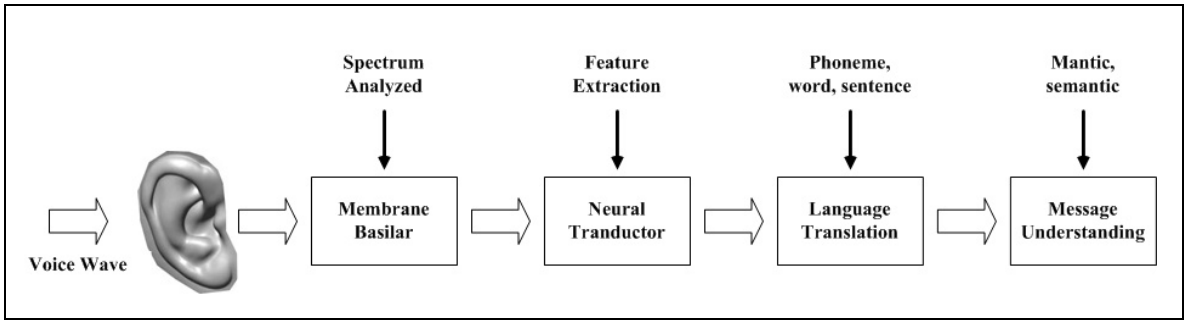

Figure 2.8: Human speech recognition process

A speech recognition system consists of the following:

- A microphone, for the person to speak into.
- Speech recognition software.
- A computer to take and interpret the speech.
- A good quality soundcard for input and/or output.

At the heart of the software is the translation part. Most speech recognition software breaks down the spoken words into phonemes, the basic sounds from which syllables and words are built up. These are analyzed to see which string of these unit best

“fits” an acceptable phoneme string or structure that the software can derive from its dictionary.

It is a common misassumption that such a system can just be used “out of the box” for work purposes. The system has to train to recognize factors associated with the users voice e.g speed, pitch. Even after this training, the user often has to speak in a clear and partially modified manner in order for his or her spoken words to be both recognized and correctly translated.

Most speech recognition software is configured or designed to be used on a stand-alone computer. However, it is possible to configure some software in order to be used over a network. We can classify speech recognition tasks and systems along a set of dimensions that produce various tradeoffs in applicability and robustness. A speech recognition system can be used in many different modes (speaker dependent or independent, isolated / continuous speech, for small or large vocabulary). Figure 2.9 show the general speech recognition architecture which it contains two main components, Features Extraction and Speech Recognizer. This architecture received speech voice as an input and text as an output.

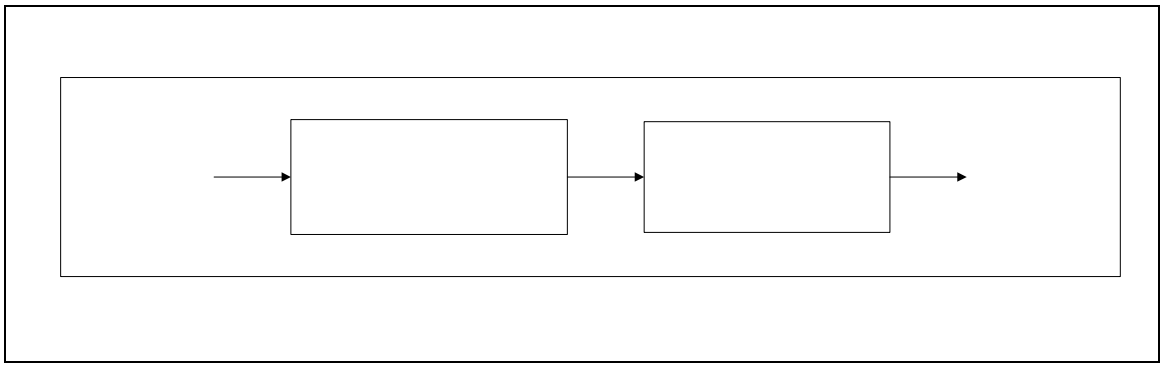

Figure 2.9: General Speech Recognition Architecture

Isolated word versus continuous speech: Some speech systems only need identify single words at a time (e.g., speaking a number to route a phone call to a company to the appropriate person), while others must recognize sequences of words at a time. The isolated word systems are, not surprisingly, easier to construct and can be quite robust as they have a complete set of patterns for the possible inputs. Continuous word systems cannot have complete representations of all possible inputs, but must assemble patterns of smaller speech events (e.g., words) into larger sequences (e.g., sentences).

Speaker dependent versus speaker independent systems: A speaker dependent system is a system where the speech patterns are constructed (or adapted) to a single speaker. Speaker independent systems must handle a wide range of speakers. Speaker dependent systems are more accurate, but the training is not feasible in many applications. For instance, an automated telephone operator system must handle any person that calls in, and cannot ask the person to go through a training phase before using the system. With a dictation system on your personal computer, on the other hand, it is feasible to ask the user to perform a hour or so of training in order to build a recognition model.

Small versus vocabulary systems: Small vocabulary systems are typically less than 100 words (e.g., a speech interface for long distance dialing), and it is possible to get quite accurate recognition for a wide range of users. Large vocabulary systems (e.g., say 20,000 words or greater), typically need to be speaker dependent to get good accuracy (at least for systems that recognize in real time). Finally, there are mid-size systems, on the order to 1000-3000 words, which are typical sizes for current research-based spoken dialogue systems.

Some applications can make every restrictive assumption possible. For instance, voice dialing on cell phones has a small vocabulary (less than 100 names), is speaker dependent (the user says every word that needs to be recognized a couple of times to train it), and isolated word. On the other extreme, there are research systems that attempt to transcribe recordings of meetings among several people. These must handle speaker independent, continuous speech, with large vocabularies. At present, the best research systems cannot achieve much better than a 50% recognition rate, even with fairly high quality recordings.

2.4 Distributed Speech Recognition

Based on (Imre Kiss et. al, 2003), in the Distributed Speech Recognition (DSR) system, the recognition process is split between the terminal device and the network server. Feature extraction (front-end) followed by parameter compression is carried out at the terminal end (server). The compressed recognition parameters are subsequently transmitted to the network server where the actual pattern matching stage (back-end) is performed. (Imre Kiss et. al , 2003) also mentioned that the development of DSR has been motivated by two main reasons. By computing the recognition parameters at the terminal end and sending them as data over the channel, it is possible to have less distorted input for the recognition process. This arrangement helps to improve the robustness against channel errors, as well as distortions introduced by the speech coding algorithm. Secondly, DSR is efficient from the channel occupation point of view, as only a handful of recognition parameters need to be sent to the network end. Figure 2.10 show the general architecture for distributed speech recognition.

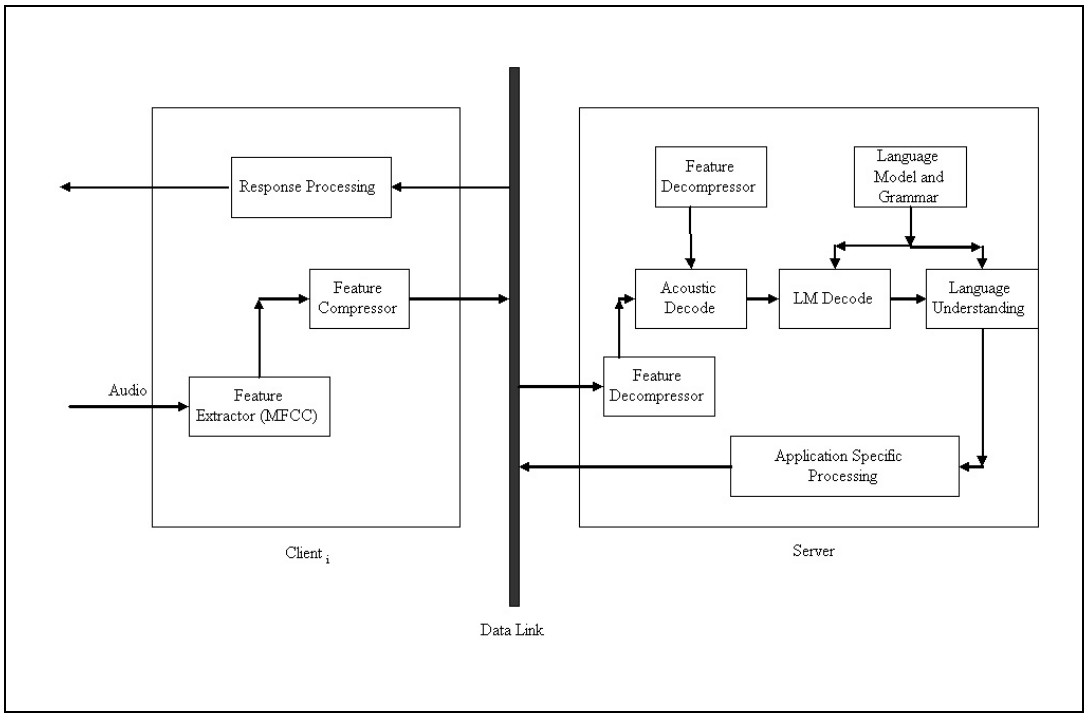

Figure 2.10: General architecture for distributed speech recognition system

2.4.1 Fundamental Architecture of DSR

Generally there are three alternative strategies in the design of DSR architectures (Wei Qi Zhang et. al, 2000). Those three architectures are as follows:

i. Server-only processing.
   All processing done at the server side and the speech signal is transmitted to the server either through the internet by using speech coding or via a second channel like telephone (Domonic Vaufreydaz et. al. 1999; Zhemin Tu and Philipos C. Loizou, 1999).

ii. Client-only Processing
   Most of speech processing is done at the client-side and the results are transmits to the server (Aldebaro Klautau, 2000).

iii. Client-server processing.
   In this model, front-end processing of speech features are transmitted to the server and finally the processing of speech decoding and language understanding are performs at the server-side. However, all the distributed speech recognition strategies were based on client-server model.

These types of architectures gave some advantages to the DSR systems. By using DSR systems:

i. By maintaining language models, dictionaries and user interface components on the server, DSR is less constrained by memory limitations on wireless devices.
ii. DSR lets the developer add voice interfaces to a variety of mobile devices without significant hardware requirements.
iii. It's much easier to update services, content and code when the majority of the recognition application is on the server.

For this development of an intelligent agent for speech recognition and translation, we used the third architecture: client-server processing. We divided speech recognition components into two parts. One part runs at the client-side (front-end processing) and the other part runs at the server-side (back-end processing). At the client-side, the user will speak a word in English and record the voice into the interface layer. The interface layer can receive and record voice from the user. After the voice is recorded, it will send to the speech encoder. The speech encoder processes the input and extracts the features of the voice. The extracts data then send to the speech decoder at the server-side. At the server-side, speech decoder will recognize the extract data and covert it into a text.

2.4.2 Advantages of DSR

Based on (David Pearce, 2000), the main benefits of DSR are as follows:

i. Improved recognition performance over wireless channel and network.
   The use of DSR minimizes impact of speech codec and channel errors that reduce the performance from recognizers accessed over digital mobile speech channels.
ii. Ease of integration of combined speech and data applications.
   Many new mobile multimodal applications are envisioned; such as the use pf speech to access wireless internet content. The use of DSR enables these to operate over a single wireless data transport rather than having separate speech and data channels
iii. Ubiquitous access with guaranteed recognition performance levels.
   There are currently 4 different major digital mobile systems each using several different codec. These all produce different effects on recognition performance. Mismatches in the channels used for training and recognition can result in severe degradations in performance, while models that have been

trained over man network give a compromise performance. DSR in the other hand offers the promise of guaranteed level of recognition performance over every network. It uses the same front-end and there is no channel distortion coming from the speech coded and its behavior in transmission errors.

2.5 Word Translation

This prototype involved two main areas, speech recognition and word translation. It will receive a wav file as an input and a text file as an output. After the prototype received a wav file, the file will be processes by speech recognition engine until get a result. The result is in text file and in English. The purpose of this prototype is to translate the English word to Malay word. So, the translation engine will translate the source word (English) to the target word (Malay). For this prototype, several technical terms has been used for both processes training and testing. The lists of the technical terms are shows at table 2.2.

Table 2.2: List of technical terms

No	Utterance (English)	Utterance (Malay)
1.	Access Time	Masa Capaian
2.	Access Right	Hak Capaian
3.	Download	Muat Turun
4.	Upgrade	Tatar
5.	Upload	Muat Naik
6.	Artificial Intelligent	Kepintaran Buatan
7.	Central Processing Unit	Unit Pemprosesan Berpusat
8.	File Transfer Protocol	Protokol Pemindahan Fail
9.	Local Area Network	Rangkaian Kawasan Setempat
10.	Wide Area Network	Rangkaian Kawasan Luas

2.6 Development Tools

In this research project, “Development an Intelligent Agent for Speech Recognition and Translation” we selected and used several tools that we found it suitable for us to develop the systems. Tools like Java Agent Jade for mobile agent development and Sphinx4 Speech Recognition engine for speech recognizer are the best framework that we have chosen.

2.6.1 Java Agent Development Framework (JADE)

JADE is completely written in Java and JADE programmers work in full Java when developing our agents. An overview of the JADE platform, the description of it architecture, main functionalities and the outline of the conceptual model is shown in (Willlie Walker et al ,2004). There are two major aspects in Jade: distributed system topology with peer-to-peer networking and software component architecture with agent paradigm. JADE is an enabling technology, a middleware for the development and run-time execution of peer-to-peer applications which are based on the agents paradigm and which can seamless work and interoperate both in wired and wireless environment.

JADE is the middleware developed by TILAB for the development of distributed multi-agent applications based on the peer-to-peer communication architecture. Both the intelligence, the initiative the information, the resources and the control can be fully distributed on mobile terminals as well as on computers in the fixed network. The environment can evolve dynamically with peers, that in JADE are called agents, that appear and disappear in the system according to the needs and the requirements of the application environment. Communication between the peers, regardless of whether they are running in the wireless or wireline network, is completely symmetric with each peer being able to play both the initiator and the responder role.

JADE is fully developed in Java and is based on the following driving principles:

- Interoperability – JADE is compliant with the FIPA specifications. As a consequence, JADE agents can interoperate with other agents, provided that they comply with the same standard.
- Uniformity and portability – JADE provides a homogenous set of API's that are independent from the underlying network and Java version. More in details, the JADE run-time provides the same APIs both for J2EE, J2SE and J2ME environment. In theory, application developers could decide the Java run-time environment at deploy-time.
- Easy to use – The complexity of the middleware is hidden behind a simple and intuitive set of APIs.
- Pay-as-you-go philosophy – Programmers do not need to use all the features provided by the middleware. Features that are not used do not require programmers to know anything about them, neither add any computational overhead.

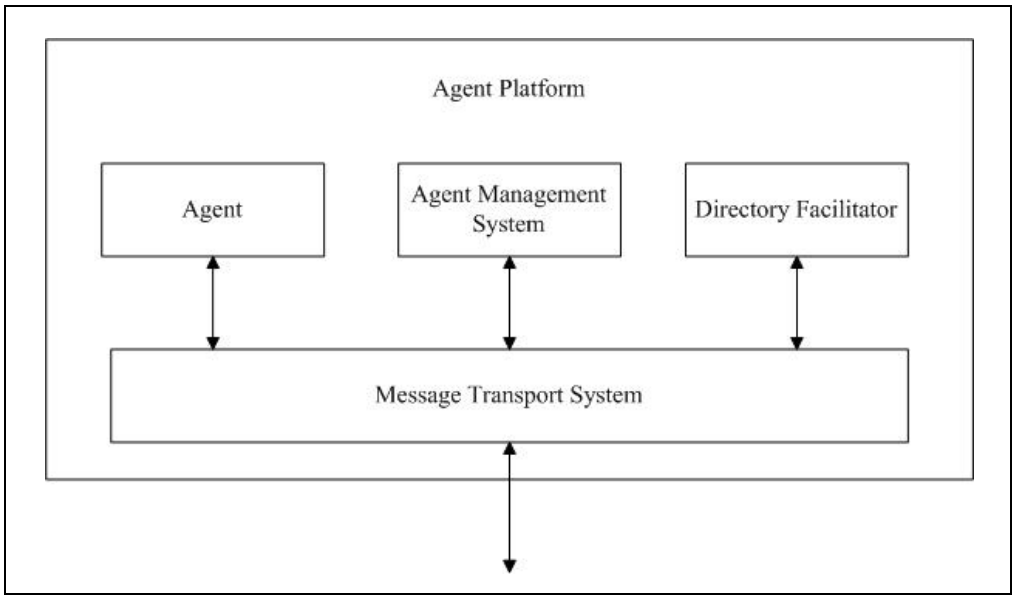

Figure 2.11: Jade platform

2.6.2 CMU Sphinx4 Speech Recognition Engine Framework

The Sphinx-4 framework has been designed with a high degree of flexibility and modularity. Figure 2.12 shows the overall architecture of the system. Each labeled element in Figure 2.12 represents a module that can be easily replaced, allowing researchers to experiment with different module implementations without needing to modify other portions of the system.

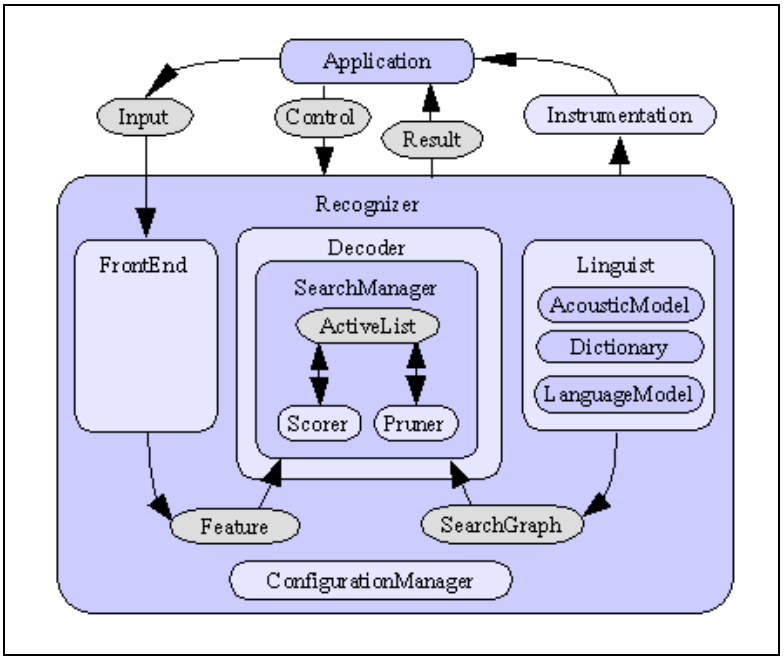

Figure 2.12: Sphinx4 Framework

There are three primary modules in the Sphinx-4 framework: the *FrontEnd*, the *Decoder*, and the *Linguist*. The FrontEnd takes one or more input signals and parameterizes them into a sequence of *Features*. The Linguist translates any type of standard language model, along with pronunciation information from the *Dictionary* and structural information from one or more sets of *AcousticModels*, into a *SearchGraph*. The *SearchManager* in the Decoder uses the Features from the FrontEnd and the SearchGraph from the Linguist to perform the actual decoding, generating *Results*. At any time prior to

or during the recognition process, the application can issue *Controls* to each of the modules, effectively becoming a partner in the recognition process.

CHAPTER III

METHODOLOGY

3.1 Introduction

Defining the project's methodology is an important task, as it can give guidelines about activities that need to be performed in order to successfully develop a system. Moreover it helps to achieve the project's objectives and vision, as well as solving the background problems. This chapter discusses the methodology of the research project: Development of an Intelligent Agent for Speech Recognition and Translation. This chapter will gives a clear view on the methodology used by describing the framework of mobile agent technology and sphinx4 speech recognition engine which implemented agent-based distributed speech recognition and translation. This chapter attempts to provide clear guidelines on how the project goal and objectives are accomplished.

3.2 Research Framework

This framework is used as a guideline through out the research for studying the appropriate system to manage agent-based distributed speech recognition and translation system environment. This framework provides a solution roadmap that goes through all the necessary phases in achieving the research goal. Figure 3.1 below represents the framework that mapped to the research scope. It provides an overview of the key initiatives taken in achieving Development of an Intelligent Agent for Speech Recognition and Translation objectives. There are four main stages involves in this research framework; Planning and Selection, Design, Operation & Implementation, and the last phase; Validation. Figure 3.1 shows the research framework for development of an intelligent agent for speech recognition and translation.

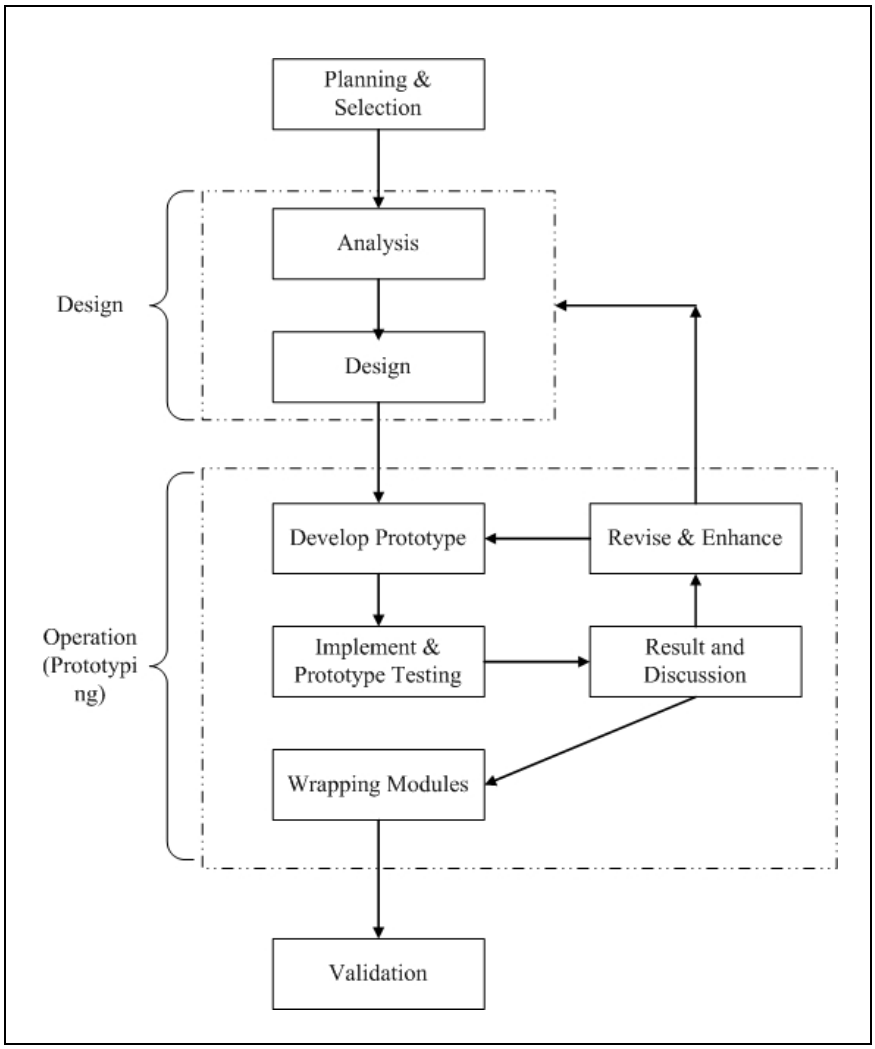

Figure 3.1: Research framework for intelligent agent for speech recognition and translation.

3.2.1 Planning and Selection Phase

Before any technical or analytical processes are performed, firstly, the system needs to be sketched down. In Planning and Selection stage, the needs are prioritizes and translates into a written plan. The planning includes making up hypothesis, defining objectives and scope, generating possible solutions, and determining how the research

should be executed. Generally, it summed up to a manageable groundwork towards the research goal.

In this stage, a broad view of the current situation within intelligent agent, mobile agent, distributed speech recognition and word translation is examined. Request for a new system springs from desire for enhancement in current system which still has large rooms for improvements are extracted. From the extracted information, possible candidates of solutions and improvements are gathered and clustered by their weakness and advantages. The best cluster of possible solutions is carried out to the next stage for analysis. The next task in this process is investigating the current of mobile agent, distributed speech recognition, and word translation system and the proposed solutions. Furthermore, the investigation leads out to the requirements of justifications and specifications for the research study.

In this phase also, research planning must be clarified. Careful planning at the beginning of the project is perhaps the single most important factor that distinguishes success from failure. From the available model and approach comparison, mobile agent technology approach is chosen for agent development while Sphinx4 speech recognition engine for recognition performances.

3.2.2 Design Phase

In the design phase, there are two stages; analysis and design. Each of the stages has their own responsibility in process to develop the system. In the analysis stage, the requirements of the proposed system are determined. Candidate solutions are discussed and compared in selecting the best alternative which meets the system needs. The best solution selected is then studied detailed down to its structure before implemented within the research study.

Before developing a working model of the research, a design is created. This is to ease development process forward in this stage. A design based on Multiagent System Engineering (MaSE) methodology is being used. MaSE methodology is similar to traditional software engineering methodologies is but specialized for use in the distributed agent paradigm (Deloach, 1998). MaSE methodology takes an initial system specification, and produces a set of formal design documents in a graphically cased style. The MaSE will guide a designer through the software lifecycle from a prose specification to an implement agent system. The MaSE will chart out and organized the proposed prototype model which is easy to developed and maintained. The next step is the development process of the model designed.

3.2.2.1 Analysis Stage

In the analysis stage, the requirements of the proposed system are determined. Candidate solutions are discussed and compared in selecting the best alternative which meets the systems needs. The best solution selected is then studied detailed down to its structure before implemented within the research study.

The analysis process does not end with the selection of the solution. The analysis is the continuing by exploring the environments of the current both mobile agent and distributed speech recognition systems. The environment are analyzed in determining the others contributions factors that has effects to the distributed speech recognition performance either positively or negatively. Fundamentals of mobile agent are then analyzed bottom up from the architectures and mobile agent models towards the policies used in distributed speech recognition and translation placement strategies. Exploration and analysis done in this stage are the essential acquaintance for the success of the next stage of this research; operation & implement.

3.2.2.2 Design Stage

At this level, design stage will define (or reuse) the agent architectures for each individual agent type. The agent architecture defines the components within each agent and how they interact. The agent level is documented. At this stage also, some of the process must be defined such as mapping actions identified in agent conversations to internal components, defining data structures identified in agent conversations and defining additional data structures, internal to the agent. At this stage also the overall of system design is specified using an AgML deployment diagram. To ease the process of the design, some steps should be followed. Those steps are: i) selecting the agent types that are needed, ii) determining the number of agents required of each type and defining. The output of this design stage will be use for system development at the next phase.

3.2.3 Operation & Implement Phase

Operation defined here as a process which the functions are defined and developed. As this stage is based on prototyping approach in software development, the prototyping process will lead out from this point though out to the end of the stage.

3.2.3.1 Develop Prototype

Based on the MaSE designed in the early phase, a working model prototype will be developed. The MaSE design will be the master plan of the development of Intelligent Agent for Speech Recognition and Translation. The interfaces defined within the AgML,

the expect inputs and outputs, the modules functions, and general design are then being developed accordingly.

3.2.3.2 Implement and Prototype Testing

The developed model is then implemented to a real system. There are two types of implementation done to test the developed model. There are alpha and beta testing. The alpha testing will be testing among team development (organization) where the outputs are the results of input generated within the team development.

The mistakes and errors found within these testing phases will be addressed in the next stage Result & Discussion for process evaluation.

3.2.3.3 Result & Discussion

After Implement & Prototype Testing stage has been done, the result will be addressed to Result & Discussion stage. This stage is a process to ensure all the result has match and fulfill all the system objectives and also to ensure the prototype is ready to work in a real environment. This stage also, the result will be discussed. The outcomes of the discussion, solid or poor result will be addressed to Revise & Enhance stage for revise and enhancement process until the prototype system ready to work in a real environment.

i. System Performance

This section is responsibility to measure system performance of Intelligent Agent for Speech Recognition and Translation. System performance can be divided into several measurement processes such as accuracy of word translation, system stability, agent communication stability and agent parallel processing accuracy.

ii. Malay acoustics model for speech recognition

This section is a process to ensure the accuracy of Malay acoustic model for speech recognition performance is ready to be implemented in a real environment. Intelligent Agent for Speech Recognition and Translation used Malay acoustic model based on phoneme acoustic model unit. Once this phoneme accuracy is higher that 95%, the Malay acoustic model ready to use. However, if the phoneme accuracy for Malay acoustic model is below than 95%, some modification or parameters setup in training process for acoustic model should be done to ensure the accuracy must higher than 95%.

iii. Objective Verification

After making the final conclusions on the findings from the analysis, the objectives of this project are verified to make sure that research has fully accomplished the vision of this project. All the objectives have to be met in order to clarify that this research project is success. The purpose of this section is to ensure all objectives, vision can be achieved and all the problems that stated earlier are solved at the end of research.

3.2.3.4 Revise & Enhance

The revise & enhance stage will ensure that all the errors and mistakes found within the working model is taken care and corrected. A suitable patch for each error will be produced if error occurred in previous phase. It is a continuing loop for the Revise and Enhance stage with Design phase and Developed Prototype stage in the Operation & Implementation phase until there are no more errors found within the working model.

If the major mistakes or errors are found within the module which needs restructuring within the MaSE designed, necessary action will be taken either involving certain changes with the initial MaSE designed or just have a changes in implementation

process. As a research study and a new adapt technology, the Intelligent Agent for Speech Recognition and Translation cannot run away from these changes as it is involving exploring new areas of solutions.

3.2.3.5 Wrapping Modules

After the two previous phases has been done, the working prototype model is then being wrapped up as a library for the use in the validation process. The module is an interactive module which it has inputs and outputs. The module structures are still designed accordingly to the MaSE designed.

At this stage, it is a wrap for the development of the Intelligent Agent for Speech Recognition and Translation prototype. The working prototype then passed from Operation & Implementation phase to the Validation phase.

3.2.4 Validation Phase

The validation phase requires implementation on a remote site where a real environment involves in the validation scene. Implementation involves the use of previous techniques and the current Intelligent Agent for Speech Recognition and Translation. This phase is similar to the beta testing used before, except no errors are expected from the execution of the prototype. Data that show performance from previous techniques and the new Intelligent Agent for Speech Recognition and Translation are collected and analyzed. Comparison is done to validate the new proposed Intelligent Agent for Speech Recognition and Translation technique. This phase is a sum of all works done in previous phases and stages. Result of the validation stage will be the benchmark that defines the advantages of the Intelligent Agent for Speech Recognition and Translation in speech recognition and translation area and also for distributed application.

3.3 Research Methodology

Intelligent Agent for Speech Recognition and Translation prototype has been developed based on research methodology that been produced. This methodology contains the process from the beginning of the executed system to the end or until prototype was shut down. Figure 3.2 shows the research methodology for developing Intelligent Agent for Speech Recognition and Translation.

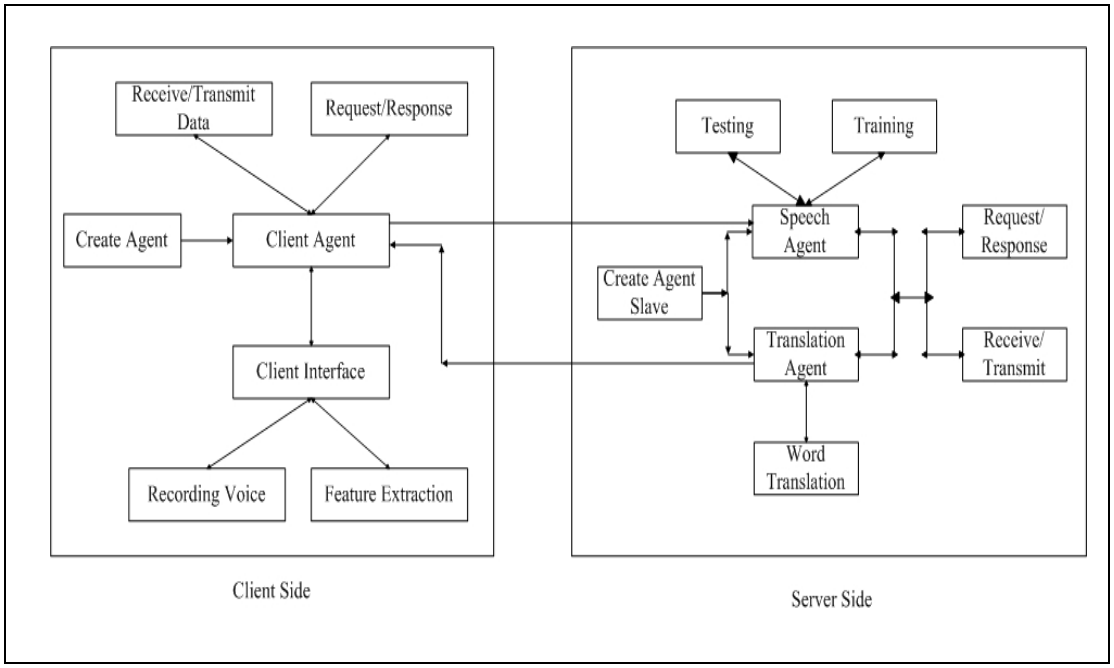

Figure 3.2: Research Methodology for Intelligent Agent for Speech Recognition and Translation.

This research methodology can be divided into two main parts, client side processing and server side processing. The important process for both parts is to create agents, *ClientAgent* (client side), *SpeechAgent and TranslationAgent* (both at the server side). These three agents have their own tasks and responsibility. The process by each part will be described separately.

At the client side, there is *ClientAgent* that control and handling processes at the client side. Once *ClientAgent* has been run, it will create three components, Client

Interface, Request/Response management and Receive/Transmit Data. Client Interface responsible for interacting between user and system, while Request/Response management responsible for request and response any process either from server side or itself. Receive/Transmit Data responsible to handle any kind of data that will send from client to server and also received data from server to client.

Client Interface will be act like connection between user and system. In distributed speech recognition system, front-end processing will be executed at the client side. Therefore, Client Interface will performs this tasks and will be guided and monitor by *ClientAgent*. Two processes can be described here in front-end processing which are recording voice and feature extraction. User voice will be recorded using Sphinx4 recording component and will save as wav files. This file will be an input for this system. Once user voice has been recorded, the feature extraction will be done. In feature extraction, the wav file will be converted from analog signal to digital signal. It's important because computer just process digital signal not analog signal. This digital signal then will be process to be a file of features using some kind of technique like *Linear Predictive Coding* (LPC), *Fast Fourier Transform* (FFT) and *Filter Bank*. This process is important because feature voice that been accepted will be use to represent reference pole and pole that want to be recognized at the recognition stage.

These components all at the client side, which these components responsible just at the client side only. On the other hand, at the server side, there also a number of components that integrated together to perform a number of tasks at the server side. At the server side, the important process still the same as at the client side, create agents. However, the numbers of agents are differences. At the server side, there are *SpeechAgent* and *TranslationAgent*. Both of them perform different responsibility. *SpeechAgent* performs a tasks related with speech recognition process, while *TranslationAgent* performs related with English-Malay word translation process. Beside that, there still some other components

At the server side, there are some components that have the same responsibility as components at the client side. Components like Request/Response and Receive/Transmit still needed at the server side. Both components will be used by both agents, *SpeechAgent* and *TranslationAgent*. However, at the server side, there still another components at the server side such as Create Agent Slave, Recognition, Training, Testing and Word Translation.

When *SpeechAgent* and *TranslationAgent* have been created, there is another component that can create slave agent, called *SpeechSlave* and *TranslationSlave*. This component called Create Agent Slave. This component will create new agent for both agents, *SpeechAgent* and *TranslationAgent* when needed. The purpose of this component is to perform concept of master/slave agent in design agent pattern techniques.

SpeechAgent related with several components such as Create Agent Slave, Receive/Transmit, Request/Response. One more components is Recognition. Recognition component is responsibility to recognize the feature file that it got from client side (front-end processing). To develop Recognition component, it consists of two processes, Training and Testing. Both processes contain data in wav files. These data were collected from a number of speakers. In this prototype, 50 speakers voice have been collected. They consist of 25 male speakers and 25 female speakers. For each speaker, they will speak 10 times of each word. There are 10 confusable technical terms in the Information Technology domain. Generally, for 50 speakers, total of collected voice is 5000 wav files. These 5000 wav files will be used for training and testing. The recognition process will used the result from these two processes. Further information will be described in the next section.

For *TranslationAgent*, there is component called Word Translation. This component is responsible to translate English word to Malay word. The English word is an input from the result of Recognition process. For each feature file that been recognized, the result is in English word and in text format. Therefore, the responsibility of Word Translation components is to translate that English word to Malay word in text

format. The ease this process, this prototype use MySQL 4.1 database to keep all the translate words. Once Word Translation components got a task to translate a word, it will search a number of data from this database. When, the translated word has been found, the result will be send back to the client side for user's view.

3.4 General Framework of Intelligent Agent for Speech Recognition and Translation.

This is the general framework for Intelligent Agent for Speech Recognition and Translation that present the working ground of mobile agent application, speech recognition engine and word translation. This framework shows where mobile agent interacted with speech recognition components and word translation engine in distributed application environment.

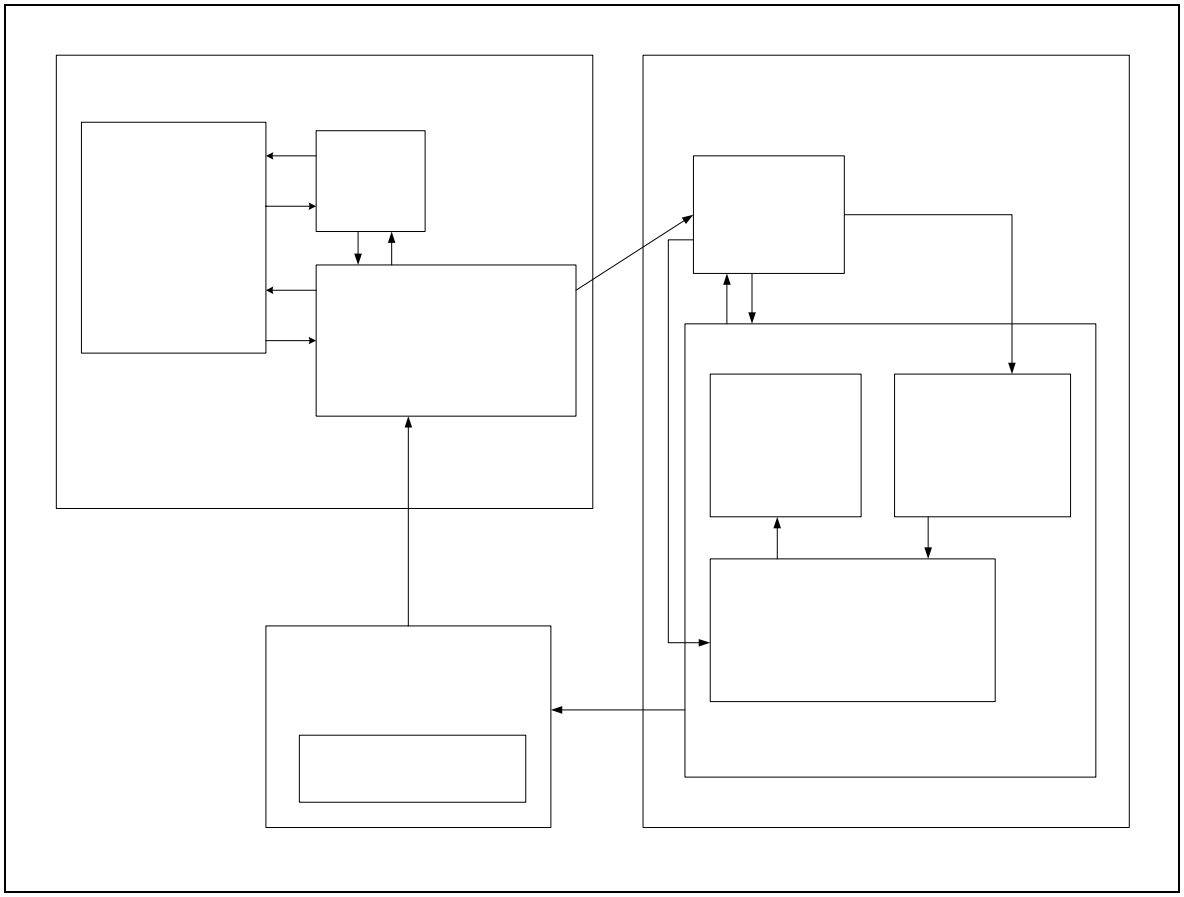

Figure 3.3: Framework of Intelligent Agent for Speech Recognition and Translation.

There are two sides of processing, server-side and client-side. At the client-side, voice will be recorded and also front-end processing will be done. The output from front-end processing is a feature file. Once the feature files ready, it will be transmitted to the server-side for recognition performance. All this process at the client-side will be handle together with agent. Agent acts like manager, manage the conversation among agent in one purpose to complete the given task.

At the server-side, recognition process and translation will be performed. The recognition process will be used the feature files that been transmitted to do that. The outcome of the recognition process is text in English word. Then, the English text will be addressed to English-Malay word translation engine by agent. Lastly, the English word will be translated to Malay word before the end result is send to the client-side.

3.5 Data Source

For data source section, it can represent as a used data for prototype development. Data source can be described as training and testing data. Further information about this data will be described below.

3.5.1 Training and Testing Data

Environment data is a raw data that can be divided into two types, raw data for speech recognition engine and a number of data for translation process. Raw data for speech recognition engine process that represents human voice in format wav files. These kinds of data will be used in training and testing phase in speech recognition engine life cycle. To create this raw data of human voice, a number of speakers had been chosen. A total of 20 speakers were used for training. 20 speakers consist of 10 male speakers and 10 female speakers. Each speaker said 10 confusable technical terms in the Information Technology domain and each word will be repeated 10 times. Therefore, there will be

2000 wav files will be collected after all speakers finish their jobs. Table 3.1 shows the list of words that speakers will speak. Data was collected by using GoldWave with data format, sampling rate: 16000 and mode: mono.

However, raw data for translation process is a data that consists of a number of words and put aside in a database. There are two columns, Malay word and English word. The database that been chose in this prototype is MySQL 4.1. Table shows the English and Malay words that keep in MySQL database.

Table 3.1 : English and Malay words in MySQL database.

No	Utterance (English)	Utterance (Malay)
1.	Access Time	Masa Capaian
2.	Access Right	Hak Capaian
3.	Download	Muat Turun
4.	Upgrade	Tatar
5.	Upload	Muat Naik
6.	Artificial Intelligent	Kepintaran Buatan
7.	Central Processing Unit	Unit Pemprosesan Berpusat
8.	File Transfer Protocol	Protokol Pemindahan Fail
9.	Local Area Network	Rangkaian Kawasan Setempat
10.	Wide Area Network	Rangkaian Kawasan Luas

3.6 Analyze Requirement

To develop a prototype of Intelligent Agent for Speech Recognition and Translation, this prototype needs two requirements, software and hardware to achieve the process development.

3.6.1 Software Requirement

There are several types of software involved in this research prototype.

i. Java Development Kit (JDK)

The development of this prototype used JDK 1.5.2 that been provided by Java developer. To execute this prototype, the system needs JDK 1.5.2.

ii. JADE 3.3 (Java Agent DEvelopment)

JADE is written in Java language and is made of various Java packages, giving application programmers both ready-made pieces of functionality and abstract interfaces for custom application dependant tasks. Java was the programming language choices because of its many attractive features, particularly geared towards object-oriented programming in distributed heterogeneous environment. JADE is an enabling technology, a middleware for the development and run-time execution of peer-to-peer applications which are based on the agent paradigm and which can seamless work and interoperate both in wired and wireless environment (A. Fuggeta et al, 2004). JADE also provide the control agent platform includes FIFA specified mandatory agents (ACC, AMS and DF). All agent communication is performed through message transfer. Message representation is based on the Agent Communication Language (ACL) (V.Gyurjyan, 2003).

iii. NetBean 4.0

This software is provided by Sun Microsystem's Netbean to support developer who works in Java language. It also provides an interface programming, so that the interface can be a middleware between user and system.

iv. CMU Sphinx4 Speech Recognition Engine

This is the main engine that been used in Intelligent Agent for Speech Recognition and Translation. The prototype speech function and agent functions will interact with this engine. Process recording function also related with this engine where the developer used it class function to record the voice.

v. MySQL 4.1

The development of this prototype used MySQL 4.1 version for database purpose. For word translation function, the prototype will save and search the exact word that want to be translate in MySQL database.

3.6.2 Hardware Requirement

Hardware also is a main component parts in prototype development. Generally, there are two computers (server and client) applied in prototype development. This hardware specification is as below:

i. Personal Computer 1 (Server)

a. Pentium IV 1.8 GHz Processor
b. 40 GB Hard Disk
c. 512 Mb RAM
d. Keyboard and Mouse
e. Monitor 17"
f. Sound Card

ii. Personal Computer 2 (Client)

a. Pentium IV 1.6 GHz processor
b. 35 GB Hard Disk
c. 512 Mb RAM
d. Keyboard and Mouse
e. Monitor 17”
f. Sound Card and Microphone

iii. Minimum requirement for Computer Client

a. Pentium 800 MHz processor
b. 128 Mb RAM
c. Keyboard and Mouse
d. Monitor
e. Sound Card and Microphone

3.7 Summary

A well designed and robust methodology is needed to run such an intensive research as the Intelligent Agent for Distributed Speech Recognition and Word Translation. The methodology designed and followed is shown to be the best fit for this research study. The next chapter will discuss the detail data and discussion about this prototype.

CHAPTER IV

DATA & DISCUSSION

4.1 Introduction

This chapter describes about two topics, the data and discussion. The data will be explained in two parts, data for DSR and agent development, and data for prototype testing. A discussion about the whole prototype system including the result also will be explained in detailed.

4.2 Data

There are two types of data that has been used in development of an Intelligent Agent for Speech Recognition and Translation. First, the data is for DSR and agent development purpose and the second one is for prototype testing. The detail description will be explained in the next section.

4.2.1 DSR & Agent Development Data

Data is one of the most important parts in this prototype development. This prototype has been building based on two main parts, DSR and agent development. Both of them need a data to ensure the development process is going well. DSR need a data to ensure the recognition engine can recognize the input voice, while a data for agent development is based on when the agent prototype system is running.

4.2.1.1 DSR Data

The data for DSR will be found in database which it called speech database. (Claudio Becchetti and Lucio Prina Ricottu, 2002) say that speech databases are essential to allow the ASR to grow on performance. The speech data may come from databases (in training step) or from a microphone. Since these data comply with different standards, the database/signal interface processes data to supply the successive blocks with a consistent input.

The speech database is a collection of recorded speech accessible on a computer and supported with the necessary annotations and transcriptions. The speech database will record different voice (data) from different people. It's important to provide the database with all possible documentation about the recorded material. The documentation produced during the database design and realization should describe recording techniques, number and type of speakers, linguistic content, etc. in the most detailed manner, to help in subsequent use by other interested people.

To develop this speech database, we collected a number of speakers to record their voice. There are several types of speech database such as Databases with few speakers, Databases of less than 50 speakers and Databases of more than 50 speakers. Based on Intelligent Agent for Speech Recognition and Translation prototype, this

prototype is a speaker-independent and continuous word. To support this speaker-independent and continuous word, Database of more than 50 speakers is suitable for that purpose. In this case, a large variability in speech styles and in recording quality, resulting from a proper distribution of speaker age and sex, is required.

Development of speech database, 50 speakers are collected. This number of speakers has been divided into several groups. These groups are based on age and male. One group consists of 25 male, and other one 25 female. This process also divided those speakers into different age groups because this process is preferable in developing a database. For instance, the age groups divide into two sub-groups, speakers under 20 years and adult from 20 to 30 years.

A total of 50 speakers were used for training to develop speech database which called Database of more than 50 speakers. 50 speakers consist of 25 male speakers and 25 male speakers. Each speaker said 10 confusable technical terms in the Information Technology domain. Table 2.2 in the previous chapter shows the list of the technical terms. Data was collected by using GoldWave with data format, sampling rate: 16000 and mode: mono. For each technical term, they will say 10 times.

At the end of this process, a small vocabulary speech database is developed. This small vocabulary database contains 5000 wav files which that enough to develop small vocabulary of Intelligent Agent for Speech Recognition and Translation prototype.

4.2.1.2 Agent Development Data

For agent development prototype, almost of the data will be develop when the prototype is running. Each agent will have their own memory of data. However, they also have permission to retrieve a data contain in JADE service such as Agent Management System (AMS) or Directory Facilitator (DF). This data is called environment data.

Those data usually help agents to ease the process of agent development and when the system is running. The data can help agent to know about other agent detail, other services that provide by other agents.

4.2.2 Prototype Data

This prototype, Intelligent Agent for Speech Recognition and Translation is a distributed applications and run on LAN or internet environment. There is a server that receives a request from multiple computers or clients that that request this recognition and translation services. For this prototype, data is use to execute the testing process. The data is a wav files contains human voice. The process just like how the training data are collected. However, the numbers of files are different from the training data.

4.3 Data Format

For every voice that been recorded, there are a format for name it. Figure 4.1 show the example of transcripts and naming files that been applied in speech database for Intelligent Agent for Speech Recognition and Translation.

```
<s> ARTIFICIAL INTELLIGENCE </s> (11AM4TR)
<s> ARTIFICIAL INTELLIGENCE </s> (11BM4TR)
<s> ARTIFICIAL INTELLIGENCE </s> (11AM1TS)
<s> ARTIFICIAL INTELLIGENCE </s> (11BM1TS)

<s> CENTRAL PROCESSING UNIT </s> (12AM4TR)
<s> CENTRAL PROCESSING UNIT </s> (12BM4TR)
<s> CENTRAL PROCESSING UNIT </s> (12AM1TS)
<s> CENTRAL PROCESSING UNIT </s> (12BM1TS)

<s> FILE TRANSFER PROTOCOL </s> (13AF4TS)
<s> FILE TRANSFER PROTOCOL </s> (13BF4TS)
<s> FILE TRANSFER PROTOCOL </s> (13AF2TR)
<s> FILE TRANSFER PROTOCOL </s> (13BF2TR)
```

Figure 4.1: Example of transcripts and format naming files

`<S> CENTRAL PROCESSING UNIT <S>` tells the transcripts of the line while `(12AM4TR)` is the name of the wav files. Each file has different name. For example, let see the four naming files at figure 4.2.

```
<s> CENTRAL PROCESSING UNIT </s> (12AM4TR)
<s> CENTRAL PROCESSING UNIT </s> (12BM4TR)
<s> CENTRAL PROCESSING UNIT </s> (12CM4TR)
<s> CENTRAL PROCESSING UNIT </s> (12DM4TR)
```

Figure 4.2: Example of naming file

There are 4 transcript files with different name files. *12AM4TR* can be divided into 5 integrated sub-groups. 12 represent a number of technical terms that want to recognize. There are 15 words will be covered in this prototype. A, B, C, D is the letters that represent a number of repeated voices. Each speaker will say 10 times of technical terms. So, ‘A’ will represent number 1, ‘B’ will represent number 2, and so on until letter ‘J’. A letter ‘M’ represent a speaker gender. ‘M’ is for Male, while ‘F’ is for Female. Number 4 represent a number of speakers that recorded their voice. There were 50 speakers collected for this speech database. Both letter ‘TR’ represented the word TRAINING. So that, this wav file is for data training. There are two types of data that were collected, data for training and data for testing.

4.4 Discussion

This sub-topic will discuss achievement of designing process, implementation process, testing process and the result. General outcome from this sub-topic will be a guide for this prototype to be improved.

4.4.1 Design Process

This prototype execute in LAN environment which it has 1 server and multiple client working together. The design of this prototype was based on distributed application. Some criteria must be consider in design development, such as network traffic, agent communication, mobility of agent, separated speech recognition engine and the way to record a voice.

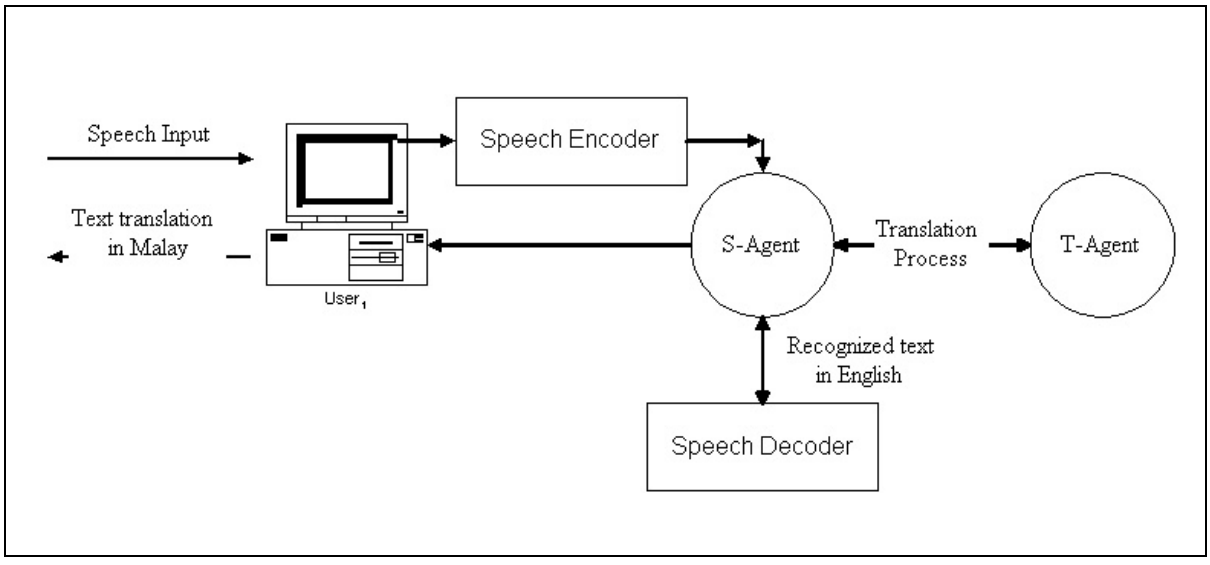

Figure 4.3 : Applying Intelligent agent for Speech Recognition and Translation Framework

Based on figure 4.3, the user will speak a word in English and record the voice into the interface layer. Interface layer is responsible to record and receive voice from user and transmit speech data to the speech encoder after voice is recorded. Speech encoder is responsible to extract the features of the voice at the client-side and the extract data will be transmits to the recognizer that need the information for decoding process at server-side. S-Agent is responsible to carry the speech information over internet and give the information to speech decoder. Speech decoder will recognize the speech and send back the result in English text to S-Agent. These are the general internet-based speech recognition process. After this, the system will go on with translation process. When S-Agent gets back the recognized speech, S-Agent will give the English in text to T-Agent, which responsible to translate the text from English to Malay. After T-Agent finish his job to translate the English word to Malay, T-Agent will give the result back to S-Agent and S-Agent will transmit the result to interface layer to show to the end user.

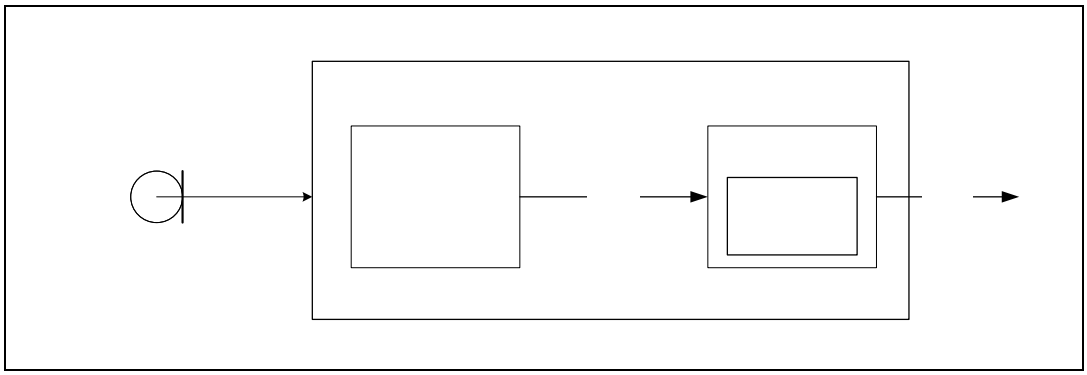

Figure 4.4 : Block diagram of the client-side for speech recognition system.

Figure 4.4 show block diagram of the client-side for speech recognition system. In the client-side, a microphone is needed to record speech. The speech recognition program is develop using Java language and will be embedded in browser. A local process is responsible for recording the speech voice using and transmitting the speech data to the sphinx program. Sphinx will process the wav file to feature extraction file, this feature file will be using for recognition process at server side and will be carried by S-Agent via network or internet.

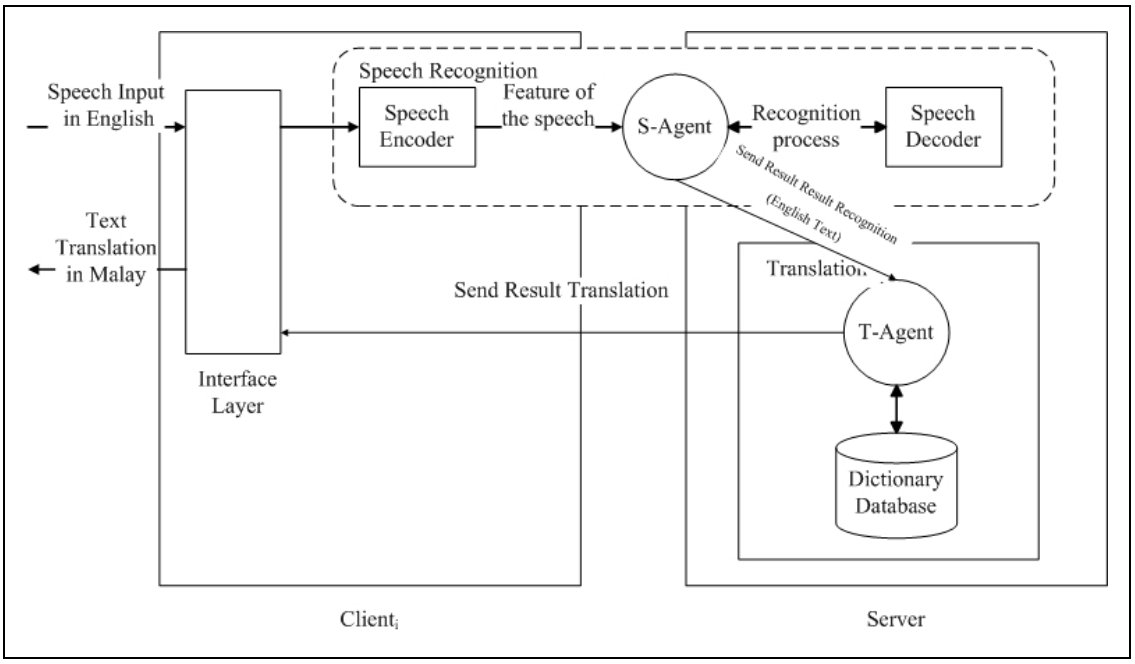

Figure 4.5: The architecture of an intelligent agent for speech recognition and translation prototype

The figure 4.5 shows the high - level of the Speech Recognition and Translation Agent Architecture Design prototype. The users will record theirs voice using the interface layer. The basic task for the interface layer is record user voice and sends to the speech decoder. After the users recorded theirs voice, they can submit their voice from the interface layer. The speech recognition will be separated into three parts, the speech encoder, speech decoder and the *S – Agent*. The responsibility of speech encoder is to extract data from voice recorded. The speech encoder contains feature extractions, and acoustic processing. The *S – Agent* is a agent that facilitate for transmitting the speech data via internet or network. The S – AGENT is acting as a data transmit control to ensure the transmitted speech data will secure delivered. The speech decoder contains a speech recognizer. The output from the speech decoder is a text in English word and will send to the *T – Agent*. The Translation has the *T – Agent*. *The T – Agent* will translate the word from English to Malay and save all the word on its memory for future purpose. The

dictionary database is a dictionary that contains all word in English and Malay for translating purpose.

Figure 4.6 show the Master-Slave Agent Architecture for Speech Recognition & Translation in Network. It describes the communication, action and responsibility for each components to perform the given tasks. Below are the descriptions of the components based on figure 4.6.

i. MasterAgent (Application)
   - Manage application request (Recognition, Training @ Translation).
   - Make a conversation with correspondent agents based on application requirement.
   - Controlling SlaveAgent (Application) life-cycle

ii. SlaveAgent (Application)
   - 3 kind of application, Speech Recognition, Speech Data Collection and Malay-English Word Translation.
   - Request service from MasterAgent (Application)

iii. MasterAgent (Recognizer)
   - Create SlaveAgent (Recognizer) based on application requirement.
   - Manage queuing for Recognition Service.
   - Choose appropriate acoustic model and grammar model based on application request.
   - Controlling SlaveAgent (Recognition) life-cycle

iv. SlaveAgent (Recognizer)
   - Perform recognition to a given feature file and reply a result as a recognize string.

v. MasterAgent (Translator)
   - Create SlaveAgent (Translator) based on application requirement.
   - Make a conversation with correspondent agents based on application requirement.
   - Controlling SlaveAgent (Translator) life-cycle

vi. SlaveAgent (Translator)
   - Searching for the recognized word in database.
   - Create MobileAgent if no translated word available

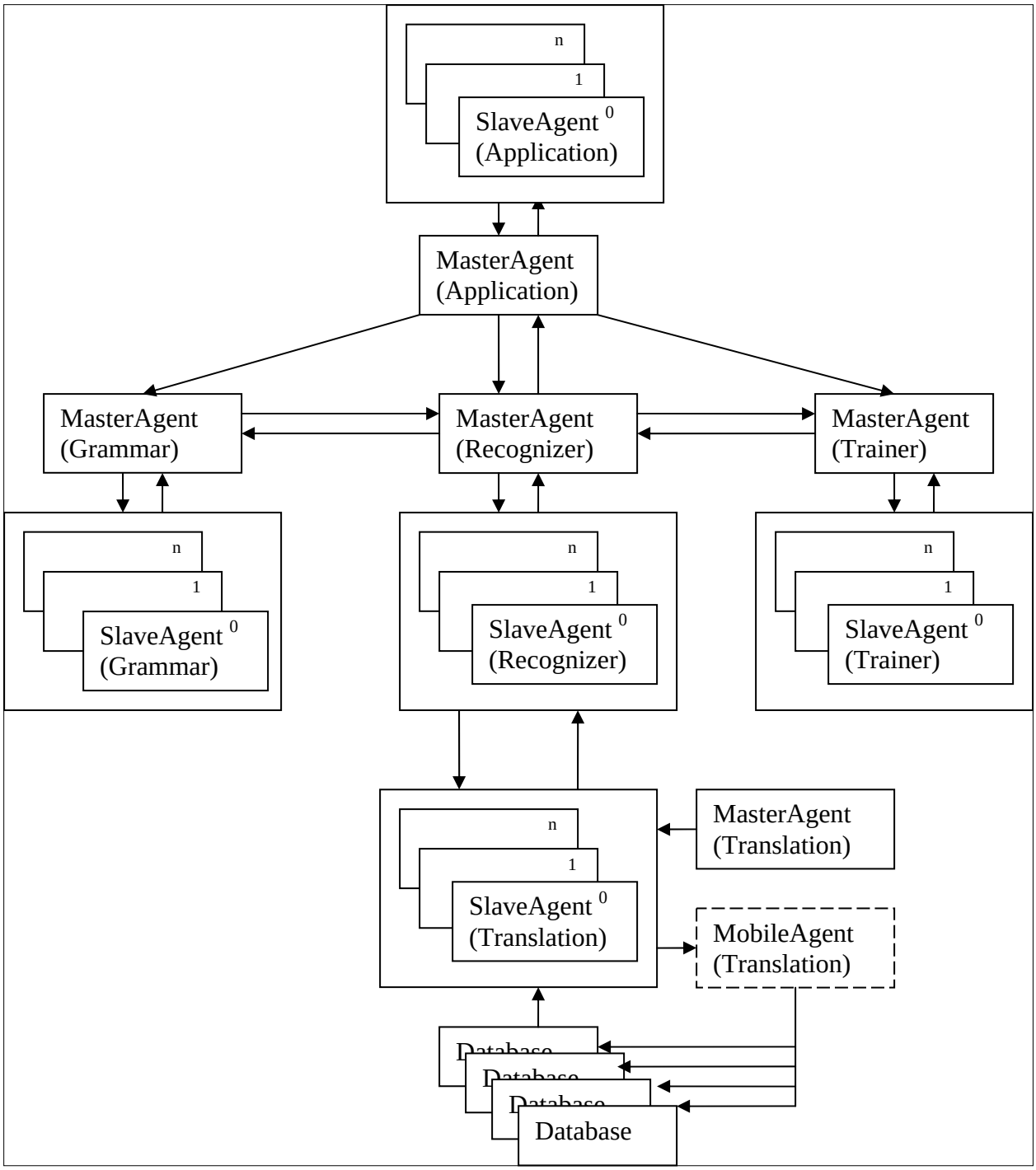

Figure 4.6: Master-Slave Agent Architecture for Speech Recognition & Translation in Network

vii. MobileAgent (Translator)
- Moving from one container to another and carry out the searching process.
- Use DataStore to storing temporary data for reference.

viii. MasterAgent (Trainer)
- Create SlaveAgent (Trainer) based on application requirement.
- Gather all the speech training data and the transcription
- Generate Transcript file, control file, dictionary and phonelist for training process.

ix. SlaveAgent (Trainer)
- Set configuration parameter for SphinxTrain based on model requirement.
- Produce continues acoustic model for recognition process.

x. MasterAgent (Grammar)
- Create SlaveAgent (Grammar) based on application requirement.
- Controlling SlaveAgent (Grammar) life-cycle
- Preparing transcription for N-gram language modeling.

xi. SlaveAgent (Grammar)
- produce language model in ARPA @ binary format.

Distributed Speech recognition processing will be divided into two parts, front-end (speech encoder) and Recognizer (Speech Decoder).

i. Decoder

Each time the search arrives at the next state in the graph, a token is created. A token points to the previous token, as well as the next state. The active list keeps track of all the current active paths through the search graph by storing the last token of each path. A token has the score of the path at that particular point in the search. To perform pruning, we simply prune the tokens in the active list.

When the application asks the recognizer to perform recognition, the search manager will ask the scorer to score each token in the active list against the

next feature vector obtained from the front end. This gives a new score for each of the active paths. The pruner will then prune the tokens (i.e., active paths) using

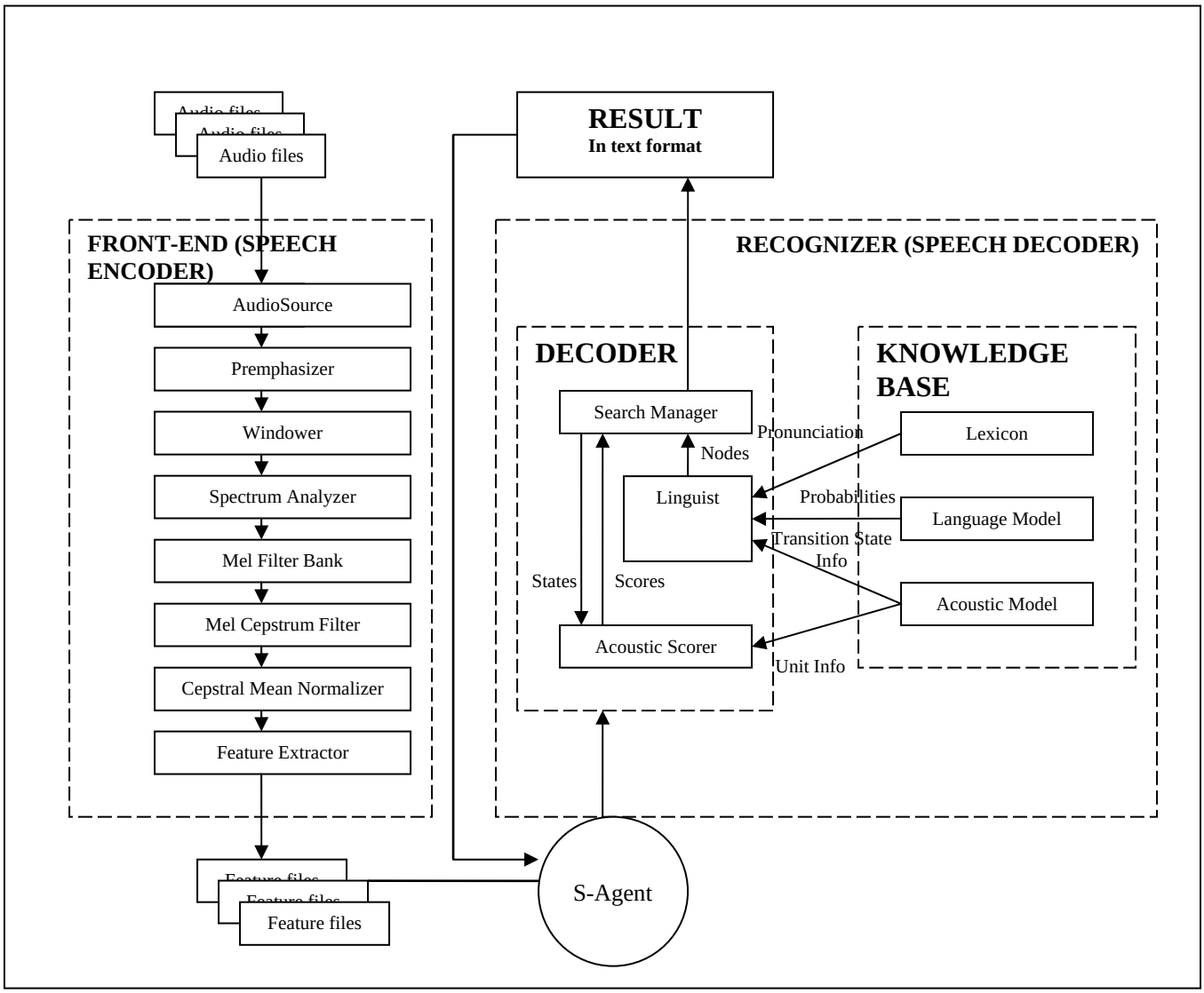

Figure 4.7 : Detailed architecture of distributed speech recognition system

certain heuristics. Each surviving paths will then be expanded to the next states, where a new token will be created for each next state. The process repeats itself until no more feature vectors can be obtained from the front end for scoring. This usually means that there is no more input speech data. At that point, we look at all

paths that have reached the final exit state, and return the highest scoring path as the result.

ii. **Front-end**

The components of front-end are shown below. Each component has task and responsibility that they must perform.

Preemphasizer

- Implements a high-pass filter that compensates for attenuation (a decrease in intensity of a signal) in the audio data

Windower

- Slices up a Data object into a number of overlapping windows (usually refer to as "frames" in the speech world).
- Minimize the signal discontinuities at the boundaries of each frame

Spectrum Analyzer

- Computes the Discrete Fourier Transform (FT) of an input sequence, using Fast Fourier Transform (FFT).
- Fourier Transform is the process of analyzing a signal into its frequency components

Mel Filter Bank

- Filters an input power spectrum through a bank of number of mel-filters

Mel Cepstrum Filter

- Applies a Discrete Cosine Transform (DCT) to the input data

Cepstral Mean Normalizer

- Applies cepstral mean normalization (CMN), sometimes called channel mean normalization, to incoming cepstral data
- reduce the distortion caused by the transmission channel

Feature Extractor

- Computes the delta and double delta of input cepstrum

4.4.2 Implementation Process

This prototype has been developed using several tools and platforms. For prototype development, there are a number of tools that provide an ability to develop this prototype, such as Net Bean 4.0 and JBuilder Enterprise. However, Net Bean 4.0 has been used in this prototype development. Net Bean provided an easiest ways to develop a system based on Java language programming. This tool will integrate those functions, speech recognition engine, agent platform and word translation engine together. Net Bean 4.0 also provided graphical user interface which ease communication between user and systems.

For agent development process, JADE framework has been used for that purposed. JADE is the platform to develop agent functions. JADE platform has been developed using Java language programming. It provides several types of agent abilities that can be applied in this prototype. Agent Management Service (AMS), Directory Facilatator (DF), agent communication, cloning, mobility, ontology and services are the example that JADE provided to develop an agent functions. JADE platform will integrated with Net Bean 4.0 tools to complete the prototype and those functions.

Differ from agent development process, CMU Sphinx4 Speech Recognition Engine has been used to develop English – Malay speech recognition engine. The process

of recording a speaker voice while system running also will used CMU Sphinx4 framework. Training process to develop English – Malay acoustic model also will be covered by CMU Sphinx4. Almost parts in speech recognition engine process will be done by CMU Sphinx4 speech recognition engine such as feature extraction part, training part, language model and recognition part.

Prototype development based on agents and several components that been defined at chapter 3 in research methodology. Agents like *SpeechAgent, Translation Agent, ClientAgent* and components like Register Agent, Receive/Transmit Data, Request/Response, Word Translation, Recognition, Feature Extraction have been developed using Jade and Sphinx4 speech recognition programming. Each function has it own responsibility and tasks.

SpeechAgent responsible to handling and performs a tasks that related with speech function while *TranslationAgent* working on word translation area. Figure 4.8 and figure 4.9 show the sample code from both of the agents.

```
protected void setup() {
    getContentManager().registerLanguage(new SLCodec());
      getContentManager().registerOntology(MobilityOntology.getInstance());
    registerAgent();
    try{
       int i = 0;

 while(true){

       ACLMessage msj = blockingReceive();
       if ("English".equals(msj.getLanguage())) {
          i++;
          AgentName an = (AgentName)msj.getContentObject();
          //AID client = an.getAID();
          String agentName = getLocalName()+i;
           System.out.println("Create SphinxAgent (SpeechRecognizer)...");
          createAgent(agentName,"SphinxAgent");
          sendMessage(new AID(agentName,AID.ISLOCALNAME),an,"client");
        }
       else if ("Malay".equals(msj.getLanguage())){
          i++;
          AgentName an = (AgentName)msj.getContentObject();
          //AID client = an.getAID();
          String agentName = getLocalName()+i;
          createAgent(agentName,"SphinxAgent2");
          sendMessage(new AID(agentName,AID.ISLOCALNAME),an,"client");
          break;
       }
     }
  }
  catch(Exception e){
     e.printStackTrace();
   }
 }
```

Figure 4.8: Sample code for *SpeechAgent*

This sample code described a request from other agents that need a service from *SpeechAgent*. Then *SpeechAgent* created new agent based on agent requested specifications. SpeechAgent get the request from message method that provided in Jade framework.

```
protected void setup(){
    registerAgent();
    getContentManager().registerLanguage(new SLCodec());
    getContentManager().registerOntology(MobilityOntology.getInstance());
    int i=0;
     while (true) {

        ACLMessage msg = blockingReceive();

    if ("result".equals(msg.getLanguage())) {
       try{
        Kptsn k = (Kptsn)msg.getContentObject();
        capaian = k.fAmblKptsn();
        String result = "";
        klien = k.fAmblPnghmtr();
        ACLMessage mesej = new ACLMessage(ACLMessage.INFORM);
        oPd = new DatabaseHandler("root","greenjade");
        result = oPd.fAmkPrktn(capaian);


        if(result == "takde"){

           ACLMessage mesej2 = new ACLMessage(ACLMessage.INFORM);
           mesej2.setContent("Perkataan GAGAL!! diterjemahkan...");
           mesej2.setLanguage("result2");
           mesej2.addReceiver(klien);
           send(mesej2);
           DataStore ds = new DataStore();
           //ds.put(,k);
          }else{
           mesej.setContent(result);
           mesej.setLanguage("result2");
           mesej.addReceiver(klien);
           send(mesej);
           }
        }catch(UnreadableException e){
           e.printStackTrace();
        }
    } else{
       System.out.println(getLocalName()+ " read Java String " + msg.getContent());
       break;
    }
}
```

Figure 4.9: Sample code for *TranslationAgent*

This is a sample code for *TranslationAgent*. It will received a English word from *SphinxAgentSlave* (recognition agent), then send the English word to *TranslationAgent* slave to perform this task. Finally, it will get the result from it agent slave and send back the final result to the *ClientAgent*.

Figure 4.10 show sample code for *ClientAgent*. *ClientAgent* responsible to handle and monitor any tasks that been assigned at the client side. It also provided GUI for user purpose, such to record voice and view result.

```
protected void setup() {
   DFAgentDescription oPgnln = new DFAgentDescription();
   ServiceDescription oServis = new ServiceDescription();
   oPgnln.setName(getAID());
   try {
      DFService.register(this,oPgnln);    //daftar servis
   } catch (FIPAException e) {
      System.err.println(getLocalName()+" registration with DF unsucceeded. Reason:
"+e.getMessage());
      doDelete();
   }

   try{
      URL url = new File(configFile).toURI().toURL();
      ConfigurationManager cm = new ConfigurationManager(url);
      dumper = new Client(cm, frontEndName);
   }catch(MalformedURLException e){
      e.printStackTrace();
   }catch(Exception e){
      e.printStackTrace();
   }

   gui = new ClientGUI(this);
   gui.setVisible(true);

 }
```

Figure 4.10: Sample code for *ClientAgent*

However, before they can perform the assigned tasks, several actions must be done. First, agents need to register their services and existence, so other agent can use the service that they provided and communicate with them easily. Figure 4.11 show the sample code for register the service. By each service, it can be applied by other agents suitable for their purposed.

```
void registerAgent(){
      DFAgentDescription df = new DFAgentDescription();
      ServiceDescription sd = new ServiceDescription();
      sd.setType("speechRecognizer1");
      sd.setName(getName());
      df.addServices(sd);
      df.setName(getAID());
   try {
      DFService.register(this,df);
      }
 catch (FIPAException e) {
      System.err.println(getLocalName()+" registration with DF unsucceeded. Reason:
     "+e.getMessage());
      doDelete();
   }
}//end of void registerAgent()
```

Figure 4.11: Sample code for register the service

To execute this system and enabled user to communicate with this system, several Graphic User Interface (GUI) are provided. There are two main interface, user interface and server interface. Figure 4.12 show the user interface at the client side while figure 4.13 shows the interface at the server side.

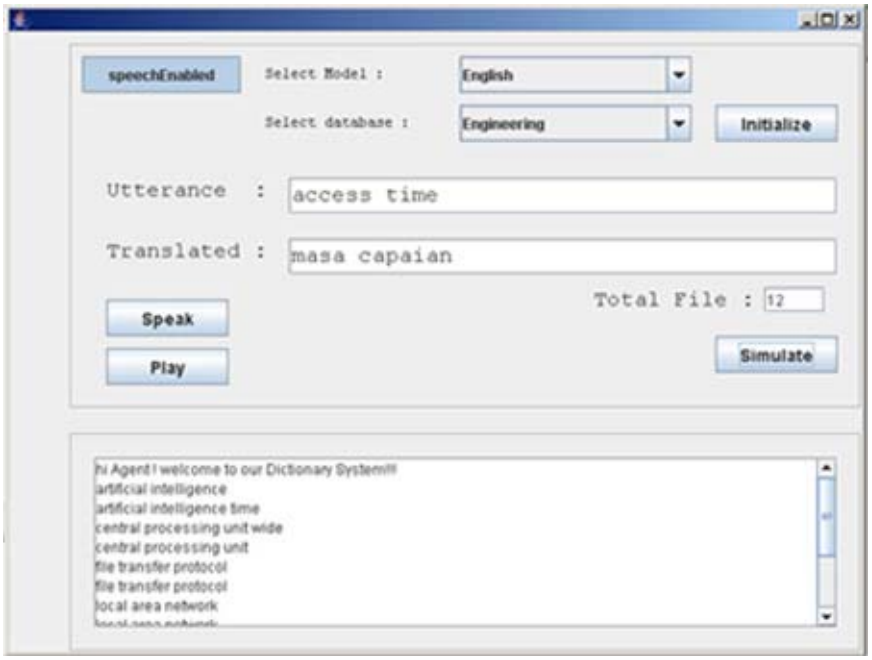

Figure 4.12: User interface at the client side

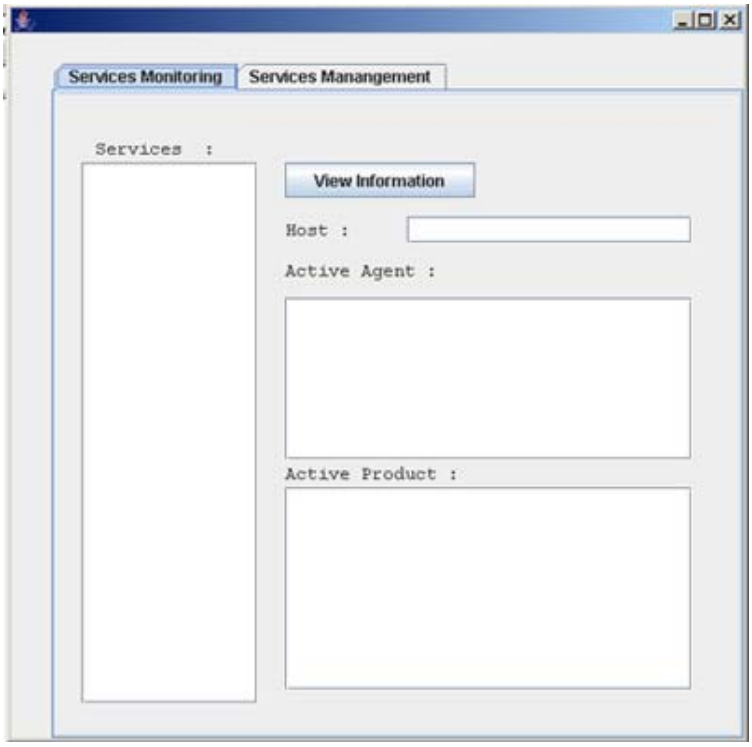

Figure 4.13: Server side interface

4.4.3 Testing Process and Result

This prototype executed in LAN environment where there is one server and multiple computer clients connected with the server. The server can be connected either with it IP address or a special name that JADE can recognize when computer client want to make a connection. For every computer clients that want to make a connection with the server, they should have JDK 1.4.2 above to run Java language and JADE platform. The prototype has been tested based on certain performances metrics, which include CPU efficiency of engine server, number of jobs (threads), response time of each request (user), amount of processed data, communication throughput and memory requirement.

There are two ways to execute and testing this prototype in LAN environment, live mode and batch mode. Live mode is a way that used human voice to represent the

raw data. The human voice will be collected while the system is running. While, batch mode is a way that system accepted input from the raw data that already inside the system at the client side. By batch mode, the human voices are collected before the system running. The human voice data save in wav file and keep in data storage system. All these files will be identified by using listing file and this file save as text file. Those files that been used for this test can be view at APPENDIX A.

The server has been tested using one server and a number of computer clients. A number of computer clients have been divided into several groups. Table 4.1 shows the number of each group and how this testing will be done. This testing also based on batch mode testing.

Table 4.1: Grouping of computer clients

Groups	Server	Number of Clients
1	1	2
2	1	4
3	1	6

The purpose of different value of computer clients of each group is to test efficiency of this prototype for each given tasks (threads) and response time of each request which based on agent platform. With this different number of clients, the efficiency of the prototype will be known, decrease, increase or static. The outcome of the testing shows that the prototype gave an interesting result. Table 4.2 shows the result for each given task to be completed by agent. The number of clients that connected with the server has been defined. The result will be taken in time measurement and in second unit.

Table 4.2, 4.3 and 4.4 shows the result that been figure out from the testing process, while table 4.4 and 4.5 shows the average time taken for each tasks to be completed and the number of tasks will be different. Those tasks will be assigned to 5 to

80 tasks for each session. Those testing have been measure in time unit (second). Generally, those testing were based on time processing for each given task to be performed from beginning of the process until the result send back to the clients.

Table 4.2 : Speech recognition time processing for 2 clients.

Server	1	
Tasks	PC 01	PC 02
	Time (second)	
05	3.76	3.94
10	6.79	6.39
15	9.30	9.28
20	12.05	10.97
25	14.40	14.02
30	16.76	15.47
35	19.24	18.89
40	21.81	20.67
45	25.34	23.97
50	28.64	28.30
55	30.67	29.45
60	33.11	32.98
65	35.85	35.19
70	37.98	37.66
75	40.67	40.14
80	43.12	41.91

Table 4.3 : Speech recognition time processing for 4 clients.

Server	1			
Tasks	PC 01	PC 02	PC 03	PC 04
	Time (second)			
05	4.21	4.41	4.25	4.44
10	9.63	10.20	9.94	8.31
15	16.23	15.61	16.34	16.09
20	19.86	19.92	20.11	19.19
25	25.08	24.80	24.44	24.78
30	29.23	29.23	29.74	28.38
35	33.98	34.08	33.58	34.27
40	38.01	35.56	37.19	38.22
45	44.47	44.53	43.72	42.99
50	47.14	44.00	48.17	48.09
55	50.84	51.10	51.52	51.49
60	57.97	56.44	57.78	57.27
65	62.28	61.84	62.16	61.55
70	67.99	65.88	65.77	63.94
75	68.87	70.06	69.63	65.72
80	74.32	70.00	71.77	73.67

Table 4.4 : Speech recognition time processing for 6 clients.

Server	1					
Tasks	PC 01	PC 02	PC 03	PC 04	PC 05	PC 06
	Time (second)					
05	8.86	9.33	8.77	8.86	9.33	9.39
10	14.13	15.94	15.20	15.94	14.95	15.86
15	21.72	20.91	21.63	20.17	19.81	22.11
20	28.04	28.55	27.39	26.50	27.08	28.09
25	35.28	34.16	33.08	33.34	34.14	34.56
30	42.40	43.55	43.53	43.30	40.49	42.56
35	45.80	46.16	45.98	45.47	46.20	46.16
40	52.44	52.89	53..48	49.61	53.27	53.09
45	60.15	57.13	59.36	58.80	57.89	59.91
50	65.73	62.45	64.13	62.73	65.22	64.39
55	71.11	71.59	71.55	72.81	69.84	73.86
60	76.49	76.67	75.02	76.80	76.17	76.45
65	80.87	82.19	81.21	81.50	82.14	81.34
70	86.10	86.47	86.92	85.30	86.72	85.61
75	91.49	90.59	90.70	91.10	90.2	91.72
80	106.60	105.06	106.38	100.94	104.39	104.72

Table 4.2, 4.3 and 4.4 shows the result from Intelligent Agent for Speech Recognition and Translation prototype executed in LAN environments. This prototype testing has been tested in three different situations. First, the server connected with 2 PC clients and performed the given tasks. Second, the server connected with 4 PC clients and performed the given tasks and lastly, the server connected with 6 PC clients and performed the given tasks. For each number of clients, they were assigned a number of tasks (feature file) to be performed by the speech recognition server and the number of tasks was 05, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 0, 75 and 80 tasks for each session. Figure 4.14, 4.15, and 4.16 shows the performance of intelligent agent for speech

recognition and word translation prototype. Axis-x refers to number of assigned tasks while axis-y refers to time processing (second).

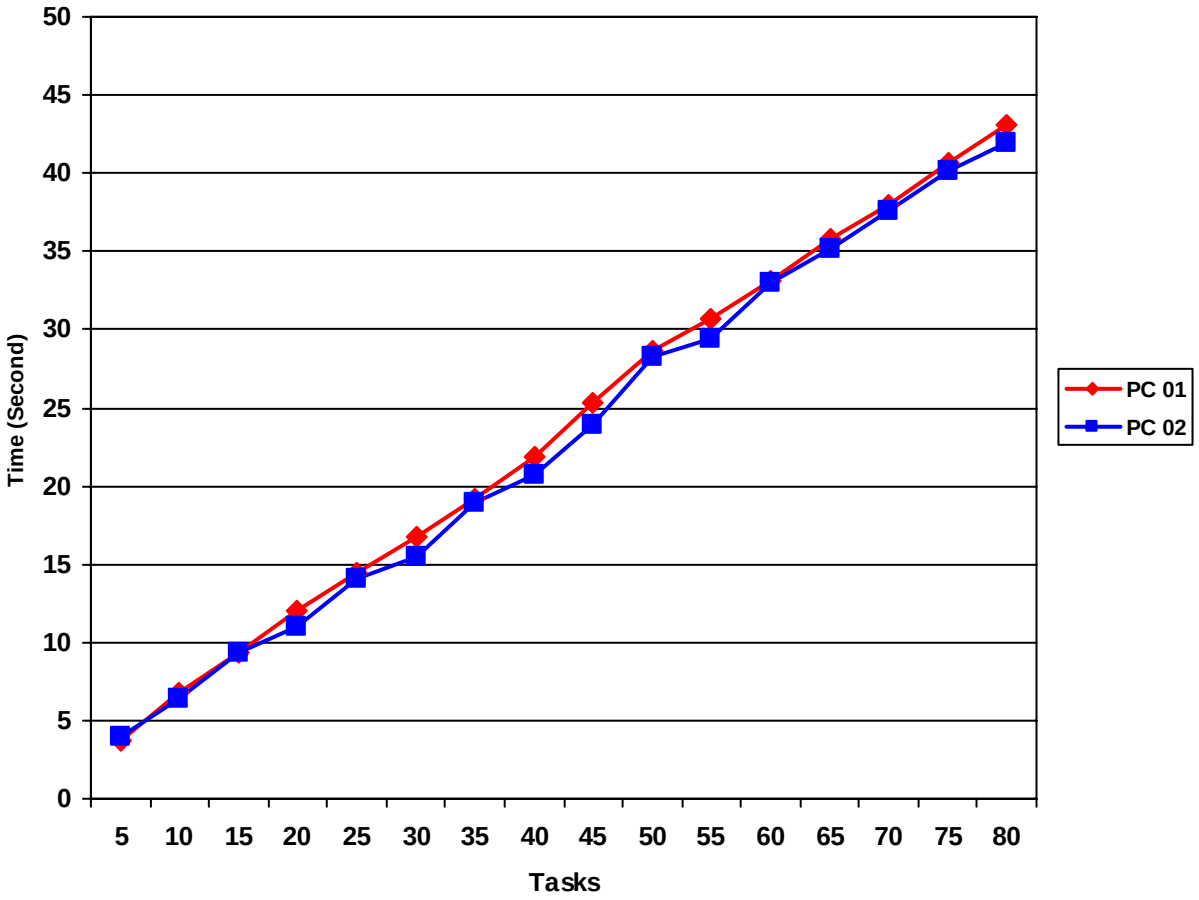

Figure 4.14: Time Processing of Speech Recognition Engine Server and Word Translation for 2 clients

This prototype, Intelligent Agent for Speech Recognition and Translation has been tested in LAN environment where there is a server and a number of computer clients connected to the server to request a recognition tasks. Generally, the result is based on certain performance metrics, such as time processes. Time processes can be grouping into several group like speech recognition server processing, client time request processing, and also result time processing.

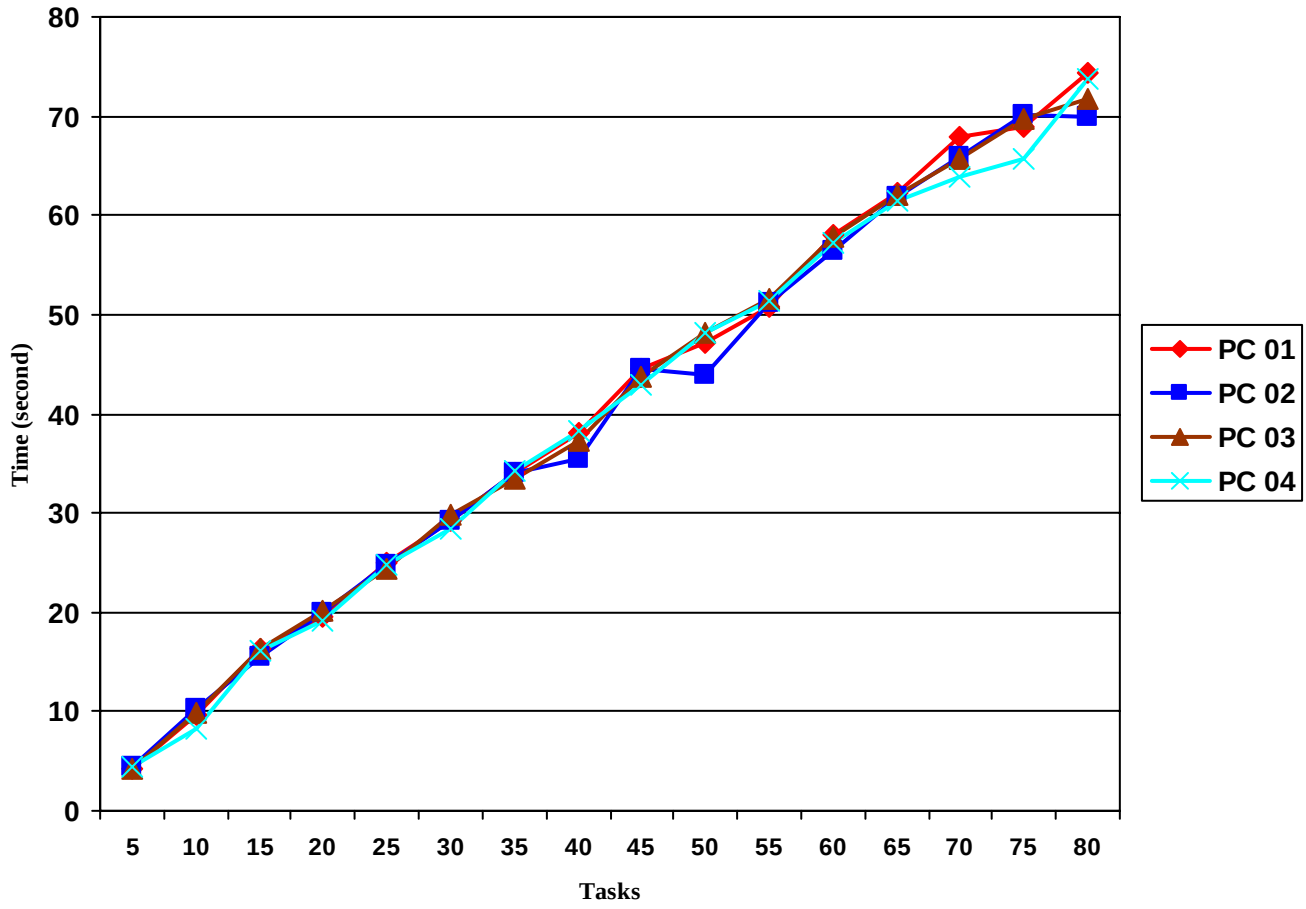

Figure 4.15: Time Processing of Speech Recognition Engine Server and Word Translation for 4 clients

Based on table 4.2, 4.3, 4.4, 4.5 and 4.6 shows the result from testing process. From those tables, it described the performance of the prototype based on time processing. Time processing has been measured in time unit. Each testing will performs a number of tasks (wav file) to be recognized and the result must translate from English word to Malay word and the full detailed of the testing process was included at APPENDIX B and APPENDIX C.

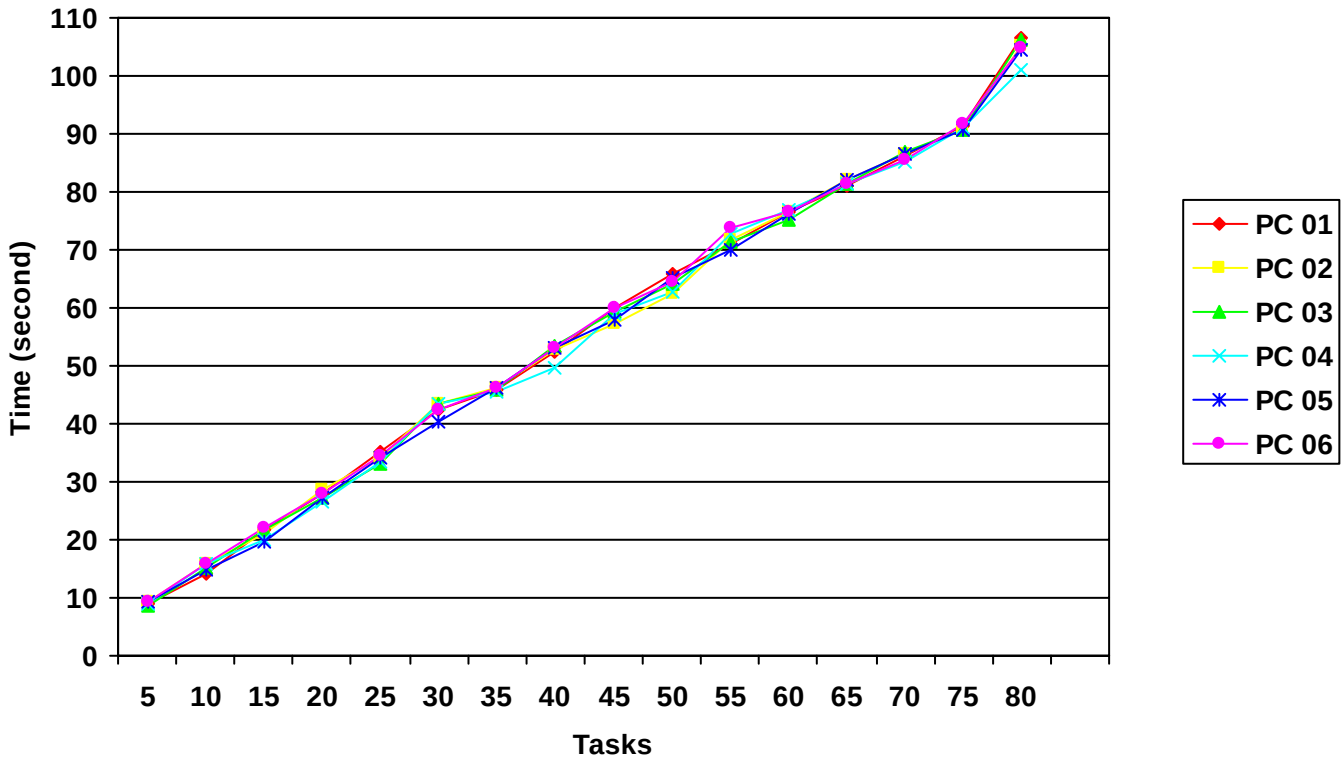

Figure 4.16: Time Processing of Speech Recognition Engine Server and Word Translation for 6 clients

Table 4.2 show time taken for each given task or wav file that must be recognize and translate. There are 2 clients connected to a server and each client assigned 05 to 80 tasks (feature files) to the server to perform, while table 4.3 shows 4 clients connected with the server and table 4.4 shows 6 clients connected with the server. The processing has been measured in time unit. For each tasks, the time were take for whole each time processing. Table 4.5 shows the average time processing for each number of tasks. Table 4.6 shows the average speech recognition engine and word translation time processing for each assigned tasks per total tasks.

Table 4.5 : Average speech recognition engine time processing for each tasks per total assigned tasks.

	2 Clients	4 Clients	6 Client
Tasks	Time Average (Second)		
05	3.85	4.33	9.09
10	6.59	9.52	15.34
15	9.29	16.07	21.06
20	11.51	19.77	27.61
25	14.21	24.77	34.09
30	16.11	29.14	42.64
35	19.07	33.97	45.96
40	21.24	37.24	52.46
45	24.65	43.93	58.87
50	28.47	46.85	64.11
55	30.06	51.24	71.79
60	33.05	57.36	76.27
65	35.52	61.96	81.54
70	37.82	65.89	86.19
75	40.41	68.60	90.98
80	42.52	72.44	104.68

From table 4.6, it shows the performance of the intelligent agent for speech recognition and translation in different number of clients connected with 1 server. For group 1, a server connected with 2 clients, it shows the time processing is still static or ±0.05 sec for each number of assigned tasks. The result shows that for group 1, the number of tasks did not effect too much to the time processing. It shows this prototype can handle many assigned tasks but still in the same time processing.

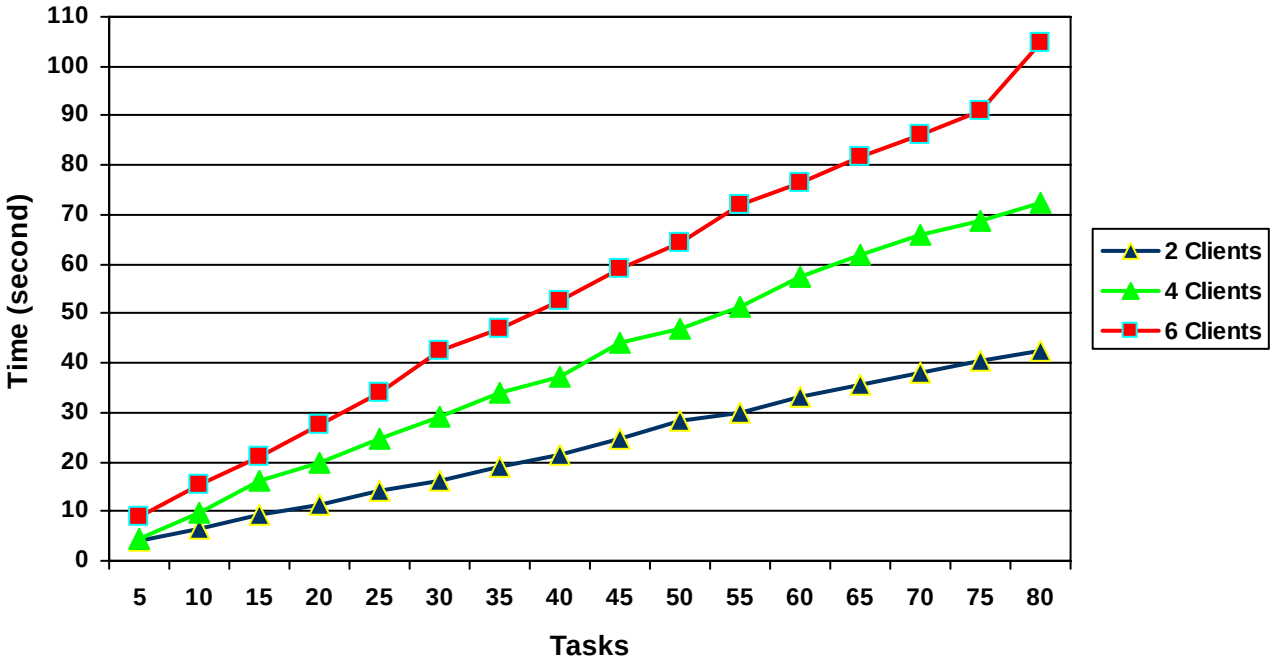

Figure 4.17: Average Task Time Processing of Speech Recognition Engine time performance

Figure 4.17 and 4.18 shows linear and static graphs produced from the prototype experiments that been done in LAN environments. They are a number of task has been assigned to the speech recognition engine server and word translation engine that responsible to perform the assigned feature files. Those feature files were been transmitted from the PC clients. Figure 4.17 show the average result from 3 set of experiments, a server connected with 2, 4 and 6 computer clients in the network. Figure 4.17 show, for each experiments show a different graph but still in a linear graph. It show that this prototype can support many computer client with many assigned tasks but overall time processing still depend on how many tasks that computer client assigned to the prototype server to perform.

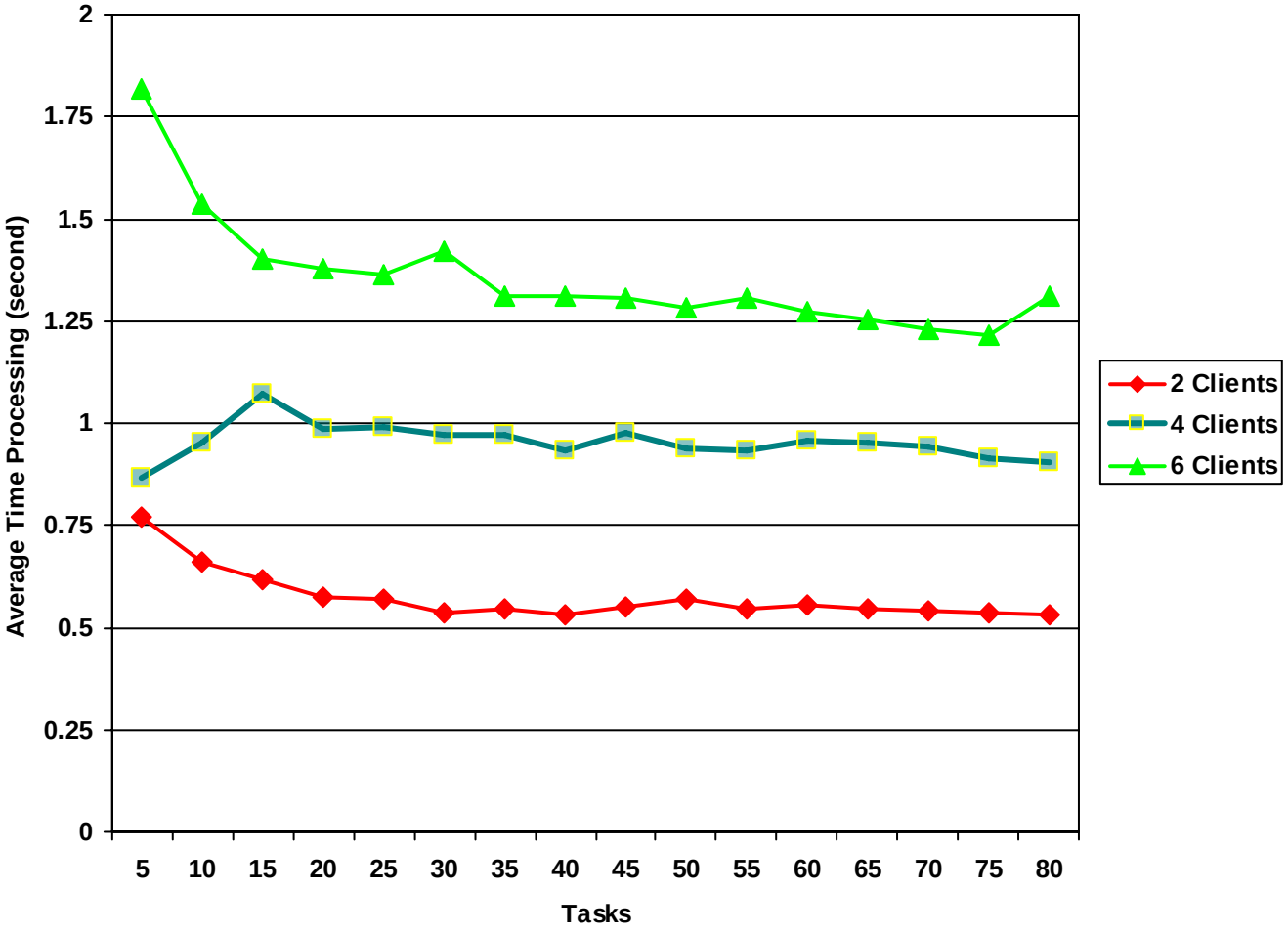

Figure 4.18: Average Speech Recognition Time Processing for every one task

From the data that been produced from the experiments, an average time processing for each assigned task has been calculate. Table 4.6 show the average data for prototype time processing for each assigned tasks per total tasks.

$$\text{Average one task processing} = \frac{\text{Average time processing}}{\text{Total Tasks}}$$

From the data that been calculated, figure 4.18 has been produced. From that graph, it show that for each experiments, a server connected with 2, 4 and 6 clients, the prototype

time processing still in static graph with ± 0.05 second increasing and decreasing. However, based on three types of experiments, the graph show that the number of assigned tasks didn't effect the prototype performance because the prototype still can performed each tasks simultaneously and comparatively. However, in the other hand, some situation, the prototype can performed the number of assigned tasks efficiently from one task to another and so on. It's learned how to perform the assigned task more faster than before as show from the graph for experiment 3 while a server connected with 6 computer clients.

Table 4.6: Average time processing for 2, 4 and 6 clients.

	2 Clients	4 Clients	6 Client
Tasks	Time Average (Second)		
05	0.77	0.87	1.82
10	0.66	0.95	1.53
15	0.66	1.07	1.40
20	0.58	0.99	1.38
25	0.57	0.99	1.36
30	0.54	0.97	1.42
35	0.55	0.97	1.31
40	0.53	0.93	1.31
45	0.55	0.98	1.31
50	0.57	0.94	1.28
55	0.55	0.93	1.31
60	0.56	0.96	1.27
65	0.55	0.95	1.25
70	0.54	0.94	1.23
75	0.53	0.91	1.21
80	0.53	0.91	1.31

4.5 Summary

Generally, to develop Intelligent Agent for Speech Recognition and Translation prototype, it needs an amount of data either for pre-development or current development.

There are two types of data, data training and data testing. Data training used for developed sample feature for speech recognition engine while data testing was used to measure the accuracy of speech recognition engine. Both of the data were collected as the same way and format but different speakers. Speakers for data training must be different from speakers for data testing. Testing phase has been done to measure the prototype efficiency based on several type of measurement such as time processes. Testing process will defined time processing in speech recognition server processing, client time request processing and result time processing. From the experiments that been done and graph that been produced, those experiments and graph shows that the speech recognition and word translation prototype can perform the assigned task simultaneously. This prototype performed parallel processing to support speech recognition engine server. So, this server can served and performed the assigned tasks precisely.

CHAPTER V

CONCLUSION AND SUGGESTION

5.1 Introduction

Agent-based speech recognition and translation is a prototype that integrated speech process and translation process in one system. This prototype is able to get an input from user's voice in English and send back the result in Malay word in text. In other word it is a word translation application. However, this prototype differs from other word translation application because the input is a voice not a text like other translation application. This prototype also involved mobile agent technology in prototype development. Agent is responsible to control and manage the process between speech function and word translation function.

Therefore, to ensure this prototype can give an excellent result to the user, several testing has been made. The purpose is to get system performance based on it accuracy on both speech recognition rate and translation rate. This is important to ensure the system can complete the user requirement and to make sure the system is comfortable and robust.

5.2 Advantages

This agent-based speech recognition and translation is a system that recognized a human voice and then translates the recognized text from English to Malay word but it's more than recognition and translation system. This system prototype has several advantages either in system performance or education values.

This prototype describes several advantages using this agent-based speech recognition and translation application. The advantages are lists below:

i. With this prototype of an intelligent agent for speech recognition and translation, a user has an opportunity to translate a language from English to Malay. The input is in spoken English and the result is in Malay word in text.
ii. This prototype is able to support multiple users at the same time. Using mobile agent paradigm, the host architecture exactly had been design to achieve that purpose.
iii. There is an interesting part where the input for this prototype is a voice not an input from a keyboard as usual. User just record their voice through an application (interface) that been developed for that purposed.
iv. In translation part, this prototype used an intelligent agent to translate a source word to a target word. We have applied an intelligent agent method to achieve that purpose not just a matching technique.

5.3 Contribution

Development of an intelligent agent for speech recognition and translation, it involved three types of area, mobile agent paradigm, speech recognition and word translation. Each area has it own way to produce a result. The architecture design that

been used in this prototype development is a new approach to integrating intelligent agent with both speech engine and word translation area. The prototype executed in LAN environment so that the speech functions will be separated into two parts, front-end processing and back-end processing. This prototype comes out with several contributions in speech and intelligent agent area.

i. This prototype involved intelligent agent technology in prototype development. The used of this technology is useful in managing and controlling processes in both speech functions and word translation functions. This technology processed the tasks in parallel and executed in real-time. So many requests from the clients can be handled by server which support by agent.
ii. This prototype proposed the new architecture and framework for speech recognition engine and word translation with mobile agent technology.
iii. This is a distributed application and executed in LAN environment which many users can use this applications since their computer connected with the server.
iv. This prototype also produced a language model for English – Malay vocabulary. This vocabulary suitable for Sphinx4 speech recognition engine and another developer who wants to develop a system that involve English – Malay vocabulary can used the vocabulary that been produced in this

However, in this research project focus on mobile agent paradigm to develop a system prototype. On the other hand, both accuracy for speech engine and word translation engine has been improved. Agent-based methodology has been applied in system prototype development. However, for speech recognition engine development, CMU Sphinx4 methodology has been chosen because it is the most suitable to achieve the best recognition result. Even though in speech recognition engine development, different methodologies are selected, but to integrate all of the subsystem, agent-based methodology is the best.

5.4 Suggestion & Future Work

After system development process has been completed, the prototype still can be improved. This prototype covered several processes between agent, speech recognition function and word translation function. Here are several suggestions or ideas to make the system prototype more flexible, useful, robust and more commercialize. These suggestions can be divided into two groups of suggestions: academic values and commercial values.

5.4.1 Academic Values

Intelligent agent for speech recognition and word translation prototype has been developed to integrate both functions speech and word translation with mobile agent technology. The method used in this prototype focus on communication between agent and speech functions. Beside that, this prototype also focus on perform the processing tasks as parallel, especially for speech recognition server at the server side. This prototype also just provided English – Malay word translation. Acoustic and language model are based on English word for recognition.

Therefore, there is still some spaces this prototype can be improve. So, for this prototype future works, there are some improvement can be done. The lists of future works are shown below:

i. This prototype should provide more than two languages (English and Malay). Language like Chinese, Japanese, Philippines, Thai's should be included in this prototype. Therefore, the contribution of this prototype can be wider and more useful. However, that needed more works and efforts to build language and acoustic model for those languages. Data must be collected again and use local speakers for each language. Speech data should be train again to

produce language model. Finally, this prototype will have their own language model that consists of several languages.

ii. Intelligent agent for speech recognition and word translation developed to execute well in LAN environment but not in internet environment. Even this prototype also can be executed in internet environment but this platform of agent not integrated with web pages yet. Therefore, this system can be improved to execute the system in internet environment. First, mobile agent platform should be communicate well with the web server. Then, the agent must integrate well with web pages.

5.4.2 Commercial Values

Intelligent Agent for Speech Recognition and Translation prototype can give a number of contributions on several areas, such as education and tourism. For education contribution, this prototype can be applied for primary school students especially for those students who want to learn quickly and fast. For example, by using this prototype, they can speak a word such as ‘duck’, then the picture of duck will appear at the screen. So, this is the way they learn. Beside that, they also can speak a word or letter, so this prototype will process and send the result to the screen.

5.5 Summary

Intelligent Agent for Speech Recognition and Translation prototype has been developed where, human voice was an input and the final result is Malay word in text format. Two main area integrated in this prototype, speech recognition area and word translation area. Both areas were integrated using mobile agent technology as an approach. For agent development, Java Agent Development (Jade) framework was used

to ease the process, while CMU Sphinx4 speech recognition engine was a framework for speech recognition process. Beside that, by using mobile agent technology, this prototype was applied master/slave agent technique to support parallel processing for both side client side and server side (speech recognition engine server and translation server). Finally, this prototype executed in LAN environment which one server connected with a number of clients through network where many users can use this prototype to achieve on what kind their purposed.

REFERENCES

Abd. Manan Bin Ahmad, Mohamad Ashari Alias, Ag.Noorajis bin Ag.Nordin, Emrul Hamide Md. Saaim, Den Fairol bin Samaon, Mohd Danial Bin Ibrahim, (2005), " An Architecture Design of the Intelligent Agent for Speech Recognition and Translation". *IEEE Region 10 Technical Conference (TENCON), 21-24 Nov 2004.*

Aldebaro Klautau, (2000). Server-assisted speech recognition over the internet. *The 2000 IEEE International Conference on Acoustics, Speech and Signal Processing.*

A. Fuggeta, G.P. Picco, and G. Vigna. "Understanding code mobility". In IEEE Transactions on Software Engineering, Volume 24, May 1998.

Antonio Cardenal-Lopez, Laura Docia-Fernandez and Carmen Garcia-Mateo. (2004). Soft decoding strategies for distributed speech recognition over IP networks. (IEEE).

Brian Brewington and Robert Gray and Katsuhiro Moizumi and David Kotz and George Cybenko and Daniela Rus, "Mobile Agents for Distributed Information Retrieval," *In Matthias Klusch, editor, Intelligent Information Agents*, chapter 15, Springer-Verlag, 1999.

Claudio Becchetti and Lucio Prina Ricottu. (1999). Speech Recognition: Theory and C++ Implementation. John Wiley and Sons Ltd, Baffins Lane, Chichester, West Sussex PO19 1UD, England.

Danny B. Lange and Mitsuru Oshima. (1998). Programming and Deploying Java™ Mobile Agent with Aglets. *Corporate, Government and Special Sales Group, Addison Wesley Longman, Inc.*

Danny B. Lange and Mitsuru Oshima. (1999). Seven Good Reason using Mobile Agent. *Communication of the ACM, Vol. 42, No. 3.*

Danny B. Lange and Mitsuru Oshima, "Mobile Agent with Java: The Aglet API," World Wide Web Journal, 1998.

David Kotz and Robert S. Gray. "Mobile Code: The Future of the Internet." In Proceedings of the Workshop ``Mobile Agents in the Context of Competition and Cooperation (MAC3)" at Autonomous Agents '99, pages 6-12, May, 1999.

David Pearce. (2000). Enabling New Speech Driven Services for Mobile Devices: An overview of the ETSI standards activities for Distributed Speech Recognition Front-ends. *AVIOS 2000: The Speech Applications Conference, San Jose, CA, USA*

David Reilly and Michael Reilly, (2002). Java Network Programming and Distributed Computing. Pearson Education Corporate Sales Division, 201 W. 103rd Street, Indianapolis, IN 46290: Addison-Wesley. 2002.

D.B Lange and M. Oshima. "Programming and Deploying Java Mobile Agents". Communications of the ACM, March 1999.

Delbert Hart, Mihail Tudoreanu, and Eileen Kraemer. “Mobile agent fo monitoring distributed systems”. Proceeding on AGENTS’01, May 28 – June 1, 2001

Domonic Vaufreydaz et. al. (1999). A network architecture for building applications that use speech recognition and/or synthesis. *European Conference an speech Communication and Technology, pages 2159-62.*

Emerson Ferreira de Araújo Lima, Patrícia Duarte de Lima Machado, Jorge César Abrantes de Figueiredo, Flávio Ronison Sampaio. (2003). Implementing Mobile Agent Design Pattern in the Jade Framework. *TILAB "EXP in search of innovation" Journal.*

Emerson F. A. Lima, Patrica D. L. Machado, Flavio R. Sampaio and Jorge C. A. Figueiredo, “An Approach to Modelling and Applying Mobile Agent Design Patterns”, ACM SIGSOFT Software Engineering Notes, Volume 29 , Issue 3, Pages: 1 - 8, (May 2004).

Emrul Hamide Md Saaim, Abdul Manan Ahmad, Mohamad Ashari Alias, Jamal Nasir Ahmad. (2005), “Applying mobile agent technology in distributed speech Recognition”. *Postgraduate Annual Research Seminar PARS'05, 17-18 Mei 2005.*

Emrul Hamide Md Saaim, Mohamad Ashari Alias, Abdul Manan Ahmad, Jamal Nasir Ahmad. (2005) “Applying Mobile Agent for Internet-based Distributed Speech Recognition”. *ICCAS 2005. International Conference on Control, Automation and System, June 2-5 in Kintex, Gyeong Gi, Korea.*

Herve Paulino. (2002). A Mobile Agent System Overview. *Departamento de Informatica, Faculdade de Ciencios e Tecnologia, Universidade Nova de Lisboa.*

Imre Kiss, Ari Lakaniemi, Cao Yang and Olli Viikki. (2003). Review of AMR Speech Codec and Distributed Speech Recognition Based Speech Enabled Services.

Automatic Speech Recognition and Understanding, 2003. ASRU '03. 2003 IEEE Workshop.30 Nov.-3 Dec. 2003 Page(s):613 – 618

John Kirriemuir. (2003). Speech Recognition Technologies.

L. Ismail and D. Hagimont. “A performance evaluation of the mobile agent paradigm”.

Michael D. Coen. “SodaBot: A software agent environment and construction system”. In Yannis Labrou and Tim Finin, editors, Proceeding of the CIKM Workshop on intelligent Information Agents, Third International Conference in Information and Knowledge Management, Gaitherburg, Maryland, December 1994.

Michael Wooldridge. “An Introduction to MultiAgent Systems”. John Wiley & Sons, LTD, 2001

Sonera Plaza Ltd MediaLab. (2001). *Voice Portals White Paper.*

Ravi Jain, Farooq Anjum and Amjad Umar, (2000), “A comparison of mobile agent and client-server paradigms for information retrieval tasks in virtual enterprises”. *Applied Research, Telcordia Technologies, Inc.*

Robert S. Gray, David Kotz, Ronald A. Peterson, Joyce Barton, Daria Chacon, Peter Gerken, Martin Hofmann, Jeffrey Bradshaw, Maggie Breedy, Renia Jeffers and Niranjan Suri. (2001). Mobile-Agent versus client/server performance: Scalability in an information-retrieval task. *In proceeding of Mobile Agent 2001. Copyright Springer-Verlag.*

Stefan Fünfrocken and Friedemann Mattern. (1999). “Mobile Agents as an Architectural Concept for Internet-based Distributed Applications - *The WASP Project Approach,” In: Steinmetz (Ed.): Proc. KiVS'99, pp. 32-43, Springer-Verlag.*

V.Gyurjyan, D. Abbott, G. Heyes, E. Jastrzembski, C. Timmer, and E. Wolin. (2003). "FIFA agent based network distributed control system". Computing in High Energy and Nuclear Physics, 24 – 28 March 2003.

Wei Qi Zhang, Liang He, Yen-Lu Chow, Rong Zhen Yang and Ye Ping Su. (2000). The Study on Distributed Speech Recognition Systems. *Acoustic, Speech, and Signal Processing. ICASSP'00. Proceedings. 2000 IEEE International Conference, Volume 3, Page: 1431 – 1434 vol.3*

Willie Walker et al, (2004). Sphinx-4: A Flexible Open Source Framework for Speech Recognition. *Sun Microsystem Inc.*

Yariv Aridor and Danny B. Lange. (1998). Agent Design Patterns: Elements of Agent Application Design. *In Proceedings of Autonomous Agents '98,* ACM Press, USA.

Zhemin Tu, and Philipos C. Loizou. (1999). Speech Recognition over the internet using JAVA. *Acoustic, Speech and Signal Processing, 1999. ICASSP'99. Proceedings, 1999 IEEE International Conference. Volume 4, Pages: 2367 – 2370 vol.4.*

APPENDIX A: List of feature files transmit to speech recognition server.

11AF2TR.MFC
11AM2TS.MFC
12AF2TR.MFC
12AF4TS.MFC
13AF2TR.MFC
13BM2TS.MFC
14AF2TR.MFC
14AM2TS.MFC
15AF2TR.MFC
15AF3TS.MFC
1AF2TR.MFC
1BF3TS.MFC
2AF2TR.MFC
2AF5TS.MFC
3AF2TR.MFC
3BF4TS.MFC
4AF2TR.MFC
4AF5TS.MFC
5AF2TR.MFC
5AM1TS.MFC
11AF2TR.MFC
11AM2TS.MFC
12AF2TR.MFC
12AF4TS.MFC
13AF2TR.MFC
13BM2TS.MFC
14AF2TR.MFC
14AM2TS.MFC
15AF2TR.MFC
15AF3TS.MFC
1AF2TR.MFC
1BF3TS.MFC
2AF2TR.MFC
2AF5TS.MFC
3AF2TR.MFC
3BF4TS.MFC
4AF2TR.MFC
4AF5TS.MFC
5AF2TR.MFC
5AM1TS.MFC