A Malay Chatterbot

By

Aina Fatiha Lokman

Dissertation submitted in partial fulfillment

of the requirements for the

Bachelor of Technology (Hons)

(Information System)

JUNE 2005

Universiti Teknologi PETRONAS

Bandar Seri Iskandar

31750 Tronoh

Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

A Malay Chatterbot

by

Aina Fatiha Lokman

A project dissertation submitted to the
Information Technology Programme
Universiti Teknologi PETRONAS
In partial fulfillment of the requirement for the
BACHELOR OF TECHNOLOGY (Hons)
(INFORMATION SYSTEM)

Approved by,

(Mr. Jale Ahmad)

UNIVERSITI TEKNOLOGI PETRONAS
TRONOH, PERAK
June 2005

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources and persons.

(AINA FATIHA LOKMAN)

ABSTRACT

Chatterbots are computer programs that simulate intelligent conversation. They make use of various techniques such as pattern matching, indexing, sentence reconstruction, and even natural language processing. In this paper, the author presents an approach to develop a chatterbot, A Malay-Intelligence Response Application (A.M.I.R.A) that will be able to communicate or converse in Bahasa Melayu. The right combination of algorithms and techniques need to be identified so that the Malay Chatterbot should be able to response to the user query in Bahasa Melayu with the right grammar and the right arrangement of sentences. This is to make sure that A.M.I.R.A will still following the famous idea of the "imitation game" (Turing, 1950), which chatterbots are developed with the aim of fooling (at least temporarily) a human into thinking they are talking to another person. RAD has been selected to be the development methodology for the project. The chatterbot will be developed using AIML (Artificial Intelligence Markup Language) which is an XML specification for programming chat robots. AIML is open source software, a great advantage for the project since the software can be freely used and, if necessary, modified. The simplicity of AIML makes it easy for non-programmers, especially those who already know HTML, to get started writing chat robots.

ACKNOWLEDGEMENT

Bismillah ar-Rahmani Ar-Raheem

In the Name of Allah, The Most Compassionate, the Most Merciful

First and foremost I would like to recite my greatest gratitude to the Most Merciful Allah for giving me the opportunity in completing this manuscript on time and without much hassle or problem. Without His observance in giving me the chance in finishing the report, there might be major problem which can resulted in delay of turning in the report in the time constrain.

One of the greatest pleasures of writing a report is acknowledging the efforts of many people whose names may not appear on the cover, but without those hardworking cooperation, friendship and understanding, producing the report could have been impossible.

I would like to dedicate my special gratitude to my Final Year Project supervisor; Mr. Jale Bin Ahmad. His insights, advices and encouragement have been invaluable throughout this project.
Also, a lot of appreciation goes to each UTP IT/IS Lecturers and FYP Committee that has guide and give advices to me regarding this FYP including all other party who involved directly or indirectly, for their assistance and precious time spent over me.

Not to be left behind, my special thanks also goes to my beloved family (Puan Kamariah Abdul Rahman, Aliaa Farhana, Amal Farahin and Arif Fadhli) and of course, all of my wonderful housemates who collectively create work culture that supports and nurtures my continued learning process and personal development.

Your guidance and assistance is highly appreciated. Thank You.

TABLE OF CONTENTS

A SHORT DECSRIPTION OF THIS REPORT

CHAPTER 1: INTRODUCTION

This chapter will give more explanation about project information which consists of background of the project, project problems, and finally project objectives. This report will give a brief about the scenario and the emergence of chatterbots in today's world and the requirements of technology nowadays especially in websites.

CHAPTER 2: LITERATURE REVIEW

Chapter 2 contains a literature review that focus on how other popular chatterbots on the Internet were implemented to converse in English. This chapter contains the recommended Artificial Intelligence techniques and algorithms that could be used to develop a Malay Chatterbot.

CHAPTER 3: METHODODLOGY

This chapter will give more explanation about methodology or procedure identification and tools that will be used in conducting this project. This methodology is implemented in order to ensure that the project is running successfully as required. Basically, the author wants to deliver a faster but cheaper product but still maintain a high quality. The methodology which has been used for this project is RAD (Rapid Application Development) with evolutionary prototyping. The detail about this project schedule can refer **APPENDIX 1**.

CHAPTER 4: RESULTS AND DISCUSSION

This section compiles the current findings of the project work. There are important findings and informative facts which are taken from journals and online resources. The information gathered were basically all the algorithms and techniques of AIML that the author could applied to this project to make the project a success.

CHAPTER 5: CONCLUSION AND RECOMMENDATION

This chapter comprises conclusion and recommendation which the author has concluded the project by providing the benefits of this project to local organizations for their websites. The recommendation for further enhancements which the author has recommended is embedding a 3D/2D animation representation to make the Malay Chatterbot more interactive and interesting. Another enhancement is having an alternative to capture user input using other devices such as microphones and still be able to respond it correctly.

LIST OF FIGURE

LIST OF TABLE

ABBREVIATION AND NOMENCLATURES

RPG - Role Playing Game

FAQ – Frequently Asked Question

AIML – Artificial Intelligence Markup Language

ALICE - Artificial Linguistic Internet Computer Entity

AMIRA – A Malay-Intelligence Response Application

XML – Extended Markup Language

AI – Artificial Intelligence

RAD – Rapid Application Development

UTP- Universiti Teknologi Petronas

WBS– Work Breakdown Structure

WWW – World Wide Web

HCI – Human Computer Interaction

FYP – Final Year Project

ECA – Embodied Conversational Agents

CHAPTER 1
INTRODUCTION

1.1 BACKGROUND OF STUDY

The rapid growth of the World Wide Web has stimulated faster growing of systems that communicate with users in natural language. These systems, currently known as *chatterbots* [1], have being used in the Internet for the most varied tasks (e.g., to take part on chat rooms and RPG, to sell products, to represent companies, to give technical support, to answer FAQs, to accompany students in distance-learning environments, among others). Chatterbots are computer program that simulates intelligent conversation. The typical execution of the program involves an input from the user in natural languages to which the program provides an answer that should sound like a reasonable and possibly intelligent response to the original sentence. The whole process is repeated while the human keeps the conversation going. In this scenario, the aim of chatterbots is to facilitate human computer interaction, since the Web relies on millions of users with different computer skill levels.

In the author research work she has identified three generations of chatterbots. The first generation was strongly based on pattern-matching techniques, whereas the second generation used Artificial Intelligence techniques. Since 1995, there have been the emergences of a third generation of chatterbots, based on the use of markup languages; AIML (*Artificial Intelligence Markup Language*) [3], the first of these languages, was used in the construction of ALICE, the owner of two recent Loebner prizes. However, no matter what the techniques are to develop a chatterbots, most of them were build to communicate in English. There are a small number of chatterbots which has been developed to converse in foreign language such as French, Deutsch (Germany), Portuguese and also Hindi. The aim of this project is to develop A Malay-Intelligence

Response Application (A.M.I.R.A) a chatterbot which will communicate in Bahasa Melayu, which will give the opportunities for Malaysian citizen who are not really good in English to experience a conversation with an artificial intelligence natural language chat robot that would replies their questions or input in Bahasa Melayu. There is not a single Malay chatterbot have been published to the Internet yet.

.2 PROBLEM STATEMENT

1.2.1 Problem Identification

- ***"Is it possible to develop a chatterbot that communicate with the right grammar and well-arranged sentence in Malay?"*** The author need to identify the best possible algorithm or techniques to construct a Malay chatterbot as most of the published chatterbots in the Internet was build to communicate in English. The construction of a sentence in Bahasa Melayu and English are different which mean the author need to find the best way to program the chatterbot so that it could communicate with a user using the right grammar and arrangement of words in Bahasa Melayu. The development of chatterbots in other foreign languages means the construction of a Malay chatterbot is not impossible.

- ***"There is not even a single Malay chatterbot has been published in the Internet".*** This means small chances for Malaysian who are not so good in English to experience a 'chat' with an artificial intelligence chat robot. This is because this group of people do not have the tendency to try to chat with an English chatterbot as they might have problems to understand what the chatterbots will reply to their questions and they will also have difficulties constructing questions as they are not very fluent in English.

- ***"I also want to experience a chat with a Malay chatterbot other than the usual English chatterbots".*** Adding another variety of chatterbots to the public as they can choose what language of chatterbots they want to communicate or 'chat'.

1.2.2 Significant of the Project

The significant of the project is to be able to develop a chatterbot that will have the ability to response reasonably to the user questions or queries in Bahasa Melayu assuming the input from user will also be in Bahasa Melayu. It will also give opportunities for those users who are not very fluent in English the experience to communicate with a computer program that is an artificial intelligence chat robot. With the successful development of the project, it may also encourage other programmer or botmaster who are interested in this field to create another Malay chatterbots with better features and more enhancements and specialties. With more varieties of chatterbots available, it will attract more users to communicate with the chatterbot and therefore indirectly attract more people to get to know about the Artificial Intelligence world. A Malay Chatterbot could also be used by local organizations that use Bahasa Melayu as their medium in their website to answer FAQs, a quick help to navigate their website and other tasks that a chatterbot are capable of assisting.

1.3 OBJECTIVE

The objective of the construction of A Malay Chatterbot is to identify the best possible combination of algorithms or Artificial Intelligence techniques to construct the framework on how the chatterbot will process the user input and able to reply the questions reasonably and if possible intelligently with the right grammar and arrangement of words in Bahasa Melayu. The framework should contribute a high fluency level as

ıther chatterbots available in the Internet to make sure user will be more interested to ıold a longer conversation with the chatterbot.

..4 SCOPE OF STUDY

Γhis project focused on developing a chatterbot using AIML, an XML language that has ɔeen designed for creating stimulus-response chat robots. AIML was developed by Dr. Ɍichard S. Wallace and Alicebot free software community during 1995-2000. It was ɔriginally adapted from a non-XML grammar also called AIML, and formed the basis for he first Alicebot, A.L.I.C.E., the Artificial Linguistic Internet Computer Entity. The ıuthor are responsible to truly understands and utilizes on how to use AIML to create ɔobot personalities like A.L.I.C.E. that pretend to be intelligent and self-aware and dentify whether it is applicable to construct a Malay Chatterbot. The basic principle ɔrocesses of AIML that has been identified that are applicable to the Malay Chatterbot ıre AIML files consist of simple stimulus-response modules called categories. Each <category> contains a <pattern>, or "stimulus," and a <template>, or "response." AIML ıoftware stores the stimulus-response categories in a tree managed by an object called the Ɔraphmaster. When a bot client inputs text as a stimulus, the Graphmaster searches the ɔategories for a matching <pattern>, along with any associated context, and then outputs he associated <template> as a response. These categories can be structured to produce nore complex humanlike responses with the use of a very few markup tags. The author ılso needs to identify other algorithm in AIML such as Conditional Branching and Γargeting if it is possible to be implemented in the construction of the project.

Ɔther than that, the author also needs to master the configuration of Program D, the ΛIML Interpreter that will be used in the project to execute the AIML files to response to he user input. Program D is also free software that has been distributed to encourage and ıimplified developer tasks to develop a chatterbot.

1.4.1 Relevancy of the project

This project is relevant as the world of Artificial Intelligence is growing rapidly. The construction of A.M.I.R.A the Malay Chatterbot would be one of the ways for Malaysian to get to know the world of A.I more closely. Nowadays, the development of chatterbot in English and other foreign languages are increasing and there is no reason for the author not to try to implement a chatterbot in Malay as it has been proven that other foreign languages has successfully implement a chatterbot that could converse in their native language. Due to that, this project is relevant to be implemented.

1.4.2 Feasibility of the Project within the Scope and Time Frame

By using RAD as a methodology, the stages from analysis, design construction and implementation can be compressed together due to the short time frame which time to develop the product less than 4 months. The Chatterbot is possible to develop in a short period by narrowing down the topic or portion of the brain to mainly talk or informed the user about UTP. The author will use all the relevant techniques or algorithm that is suitable to define the brain of this chatterbot.

CHAPTER 2
LITERATURE REVIEW

Chatterbots [1] are computer programs that attempt to simulate typed conversations with the users. The complexity of their algorithm is variable, but in general they are programmed to respond to user inputs with canned prescript statements. In this way, chatterbots can have a somewhat logical conversation with a human user, even without being capable of understanding. Rather, they are all about the illusion of intelligence and the suspension of disbelief on the part of the user. Following the famous idea of the "imitation game" [[2] Turing, 1950], chatterbots are developed with the aim of fooling (at least temporarily) a human into thinking they are talking to another person.

After a careful analysis of the area,three generations of chatterbots has been identified. The first generation was strongly based on pattern-matching techniques, whereas the second generation used Artificial Intelligence techniques. Since 1995, there have been the emergences of a third generation of chatterbots, based on the use of markup languages; AIML (*Artificial Intelligence Markup Language*) [3], the first of these languages, was used in the construction of ALICE [4], the owner of two recent Loebner prizes.

One of the oldest and best-known chatterbots in the world is ELIZA. 'She' was created in the '60s by MIT scientist Joseph Weizenbaum to play the role of a psychotherapist in a clinical treatment [[5] Weizenbaum, 1966]. Nowadays, Eliza may look limited but her fundamental components are still at the basis of the most innovative chatterbots. Eliza identifies the 'most important' keywords occurring in the input message. Then, she tries to define a minimal context in which the keywords appear (e.g., the keyword "you" followed by the word "are" is interpreted as an assertion). Finally, Eliza chooses an appropriate transformation rule to modify the user input. To summarize, Eliza works by

turning the user sentences around. Eliza was (and still is) a success. Talking to her, users somehow set up a relationship. Moreover, the computer program demonstrated a strong potential for getting personal information: users were keen to reveal to Eliza their deepest feelings. Her popularity is related to the choice of a very convenient conversation setting.

The second generation deploys Artificial Intelligence techniques, such as production rules and neural networks, in the construction of chatterbots. Julia [6], which is based on production rules, is probably its most prominent representative. Although this generation used more sophisticated techniques, the obtained results (regarding dialogue fluency) were not superior to the levels obtained in the previous generation. This fact becomes more evident when we observe the results of the Loebner prizes.

ALICE (Artificial Linguistic Internet Computer Entity) [4] is an entertaining chatterbot created by Dr. Wallace in 1995 and continuously improved over the years. Alice asks and answers questions, acts as a secretary reminding people of appointments, spreads gossips and even tells lies. 'She' won the 2000 Loebner Prize, a restricted Turing test [[2] Turing, 1950] to evaluate the level of 'humanity' of chatterbots. Alice was rated the 'most human computer' but was not mistaken for a human, as the original contest would have required. The basis for Alice's behaviour is AIML, or Artificial Intelligence Markup Language [3], an XML specification for programming chatterbots. It follows a minimalist philosophy based on simple stimulus-response algorithms, allowing programmers to specify how Alice will respond to various input statements.An AIML knowledge base consists of "categories" of question-answers pairs (see an example of AIML category in figure 1). Due to its relatively simple syntax, AIML is contributing to the popularization of chatterbots in websites.

```
<category>
<pattern> BYE </pattern>
<template> See you later. </template>
</category>
```

Figure 2.1. Example of AIML category

The code of Alice is freely available under the GNU licence statement. Hence, hundreds of people around the world have contributed to the success of Alice and of her many companions built upon the same technology, such as Cybelle, Ally, Chatbot ICQza, and the somewhat worrying Persona bots. The latter are chatterbots inhabited by unique human personalities. They currently attempt to 'clone' John Lennon and Elvis. The ambitious goal for AIML is to create a Superbot that merges the 'mind' of individual bots. Alice represented a very interesting research tool for investigating the social dynamics underlying human chatterbot interaction. Indeed, her linguistic capability was strong enough to create the illusion of a synthetic personality. Moreover, the program automatically stores client dialogues in a log file, which can be easily analyzed. Further, the Windows version, which can be used locally, does not provide any visual representation of the chatterbot. If prompted, the system gives a number of cues about her appearance and invites the user to see a picture at her web-site.

Recent years have witnessed an extraordinary explosion of interest in chatterbots. This interest is mainly driven by the e-market, namely by the increasing demand for innovative strategies to increase sales and ensure customers loyalty [7] De Angeli *et al.*, 2001]. E-service providers are now acutely aware that their potential customers are only one click' away from a competitor. They need interfaces capable of gaining the attention of customers, understanding their needs and supporting them throughout the transaction process. Chatterbots are expected to function as dedicated sales assistants in traditional shops. They should greet customers when they return to the site, engage them in chats, remember and comment on their preferences. The first figures provided by Extempo, one of the leading US chatterbot companies, pleased many Web strategists [[8] Leaverton, 2000]. Almost 90% of the customers who have clicked one of its bots have chatted for more than 12 minutes. During the dialogue customers appeared to disclose precious marketing information. They responded n average of 15 times, with an average of five words per response. Several companies are emerging to produce and sell personalized and embodied chatterbots and many websites are already employing them.

Whether or not chatterbots will be successful and will replace live-customer services on he Web remains an open question. There is little and controversial research assessing ocial agents' effectiveness and most of the research that has been published so far relates) pedagogical agents [[9] Dehn and van Mulken, 1999]. Advocates assume that the new :chnology is particularly well suited to establish relationships with users [[10] Laurel, 997].

CHAPTER 3
METHODOLOGY/PROJECT WORK

3.1 PROCEDURE IDENTIFICATION

The Author wanted to deliver a faster but cheaper product but still maintain a high quality and identified 5 phases of RAD (Rapid Application Development) with evolutionary prototyping that would be the most suitable development methodology for the project. The phases involved in this RAD are shown in Figured 3.1.

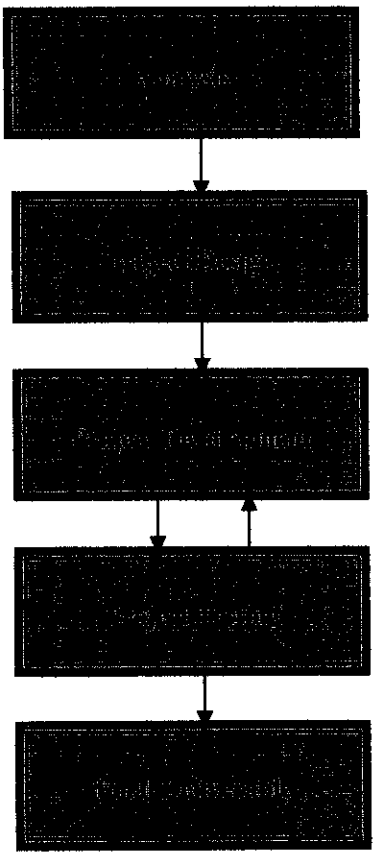


Figure 3.1 Rapid Application Development Phases

:volutionary prototyping means that the any complete application of this will be based n the prototype itself. In other word, the prototype can be enhanced to be the final nished product. If the prototype can be enhanced to be the final product, the time, cost ıd energy can be saved. These are some advantages of evolutionary prototyping. This chnique is chosen because it reduces the time of development the system. The :celeration of the system development process is achieved by requiring the developer to : more focused and actively involved in system analysis, design and development ıges.

ıere are 5 broad phases to RAD that engage both users and analysts in assessment, sign and implementation. These phases are further explained below.

METHOD

3.2.1 Analysis\ Research

Analysis stage involves the activity of gathering information. The information gathered includes the problems that may arise and all the possible techniques and algorithms to develop the chatterbot. The purpose of doing the analysis or information gathering is to obtain as much information as possible to be able to understand better about the existing chatterbot and to develop a unique new one. Among methods used to gather information are by conducting research on the sources available on the Internet websites, references, white papers and journals. Another important issue at this point of stage is to identify suitable tools to be used to develop the project.

3.2.2 Project Design

During this stage the project plan will be developed by using Gantt chart (refer to **Appendix 1** for project timeline). Besides that each of the submission date will be mark as milestone. These are to ensure that the project will be delivered on time. Besides that, the design phase is concerned with how the product developed will function according to the requirements. In this stage, the design process of the

framework principles for the chatterbot should has been identified and started to be implement.

The objectives and scope definition is considered very crucial in a project's phase as they reflect user's need and demand. In defining this critical area, the Work Breakdown Structure (WBS) is used. WBS can identify the overall project tasks which make it easier for scope definition. The WBS for this project is shown in the diagram below:

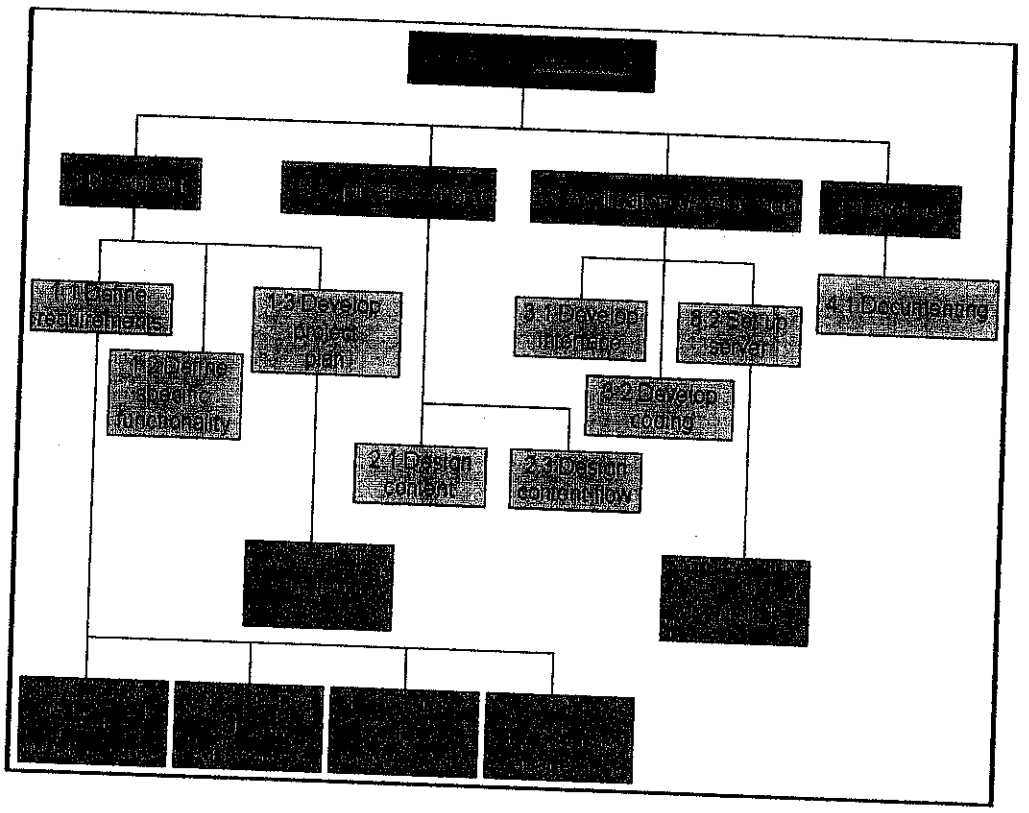


Figure 3.2 Work Breakdown Structure

3.2.3 Project Development

The project development phase starts with constructing the backbone framework on how the chatterbot will process the user input in order to produce a suitable output to users. The construction will be based on the algorithms and techniques that are identified during the previous phase (project design phase). The tools required to develop this tool is the AIML, an XML language that has been designed for creating stimulus-response chat robots.

The prototype of the application is implemented in a local server of author's own PC. However, apart from the local server, in the future, it may be implemented in World Wide Web (www) for worldwide distribution. Basically, the complete prototype is implemented following the system architecture as shown in the diagram below:

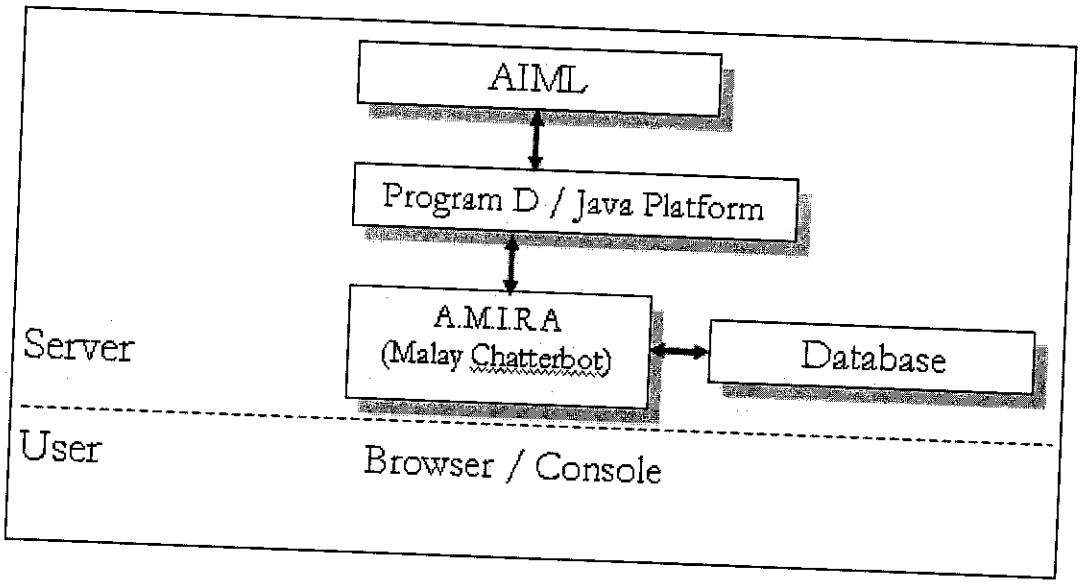


Figure 3.3 System Architecture

3.2.4 Project Testing

The chatterbot will not need to be fully developed to be tested but the project testing can be carried out to the prototype even before the construction of the chatterbot's framework is completed. Project development stage and project testing will run concurrently after the development of the system is started. This is to ensure that every functionality meets the expectation.

3.2.5 Final Deliverable

For this phase the final report will be prepared and the product will be presented to the examiners. Lastly the final dissertation will be submitted to the supervisor. This phase is the last phase in the methodology for this project.

3.3 TOOLS

3.3.1 DEVELOPER'S SPECIFICATION

3.3.1.1 Software

The table below shows the softwares that are used for the chatterbot development and their usage:

Software	Usage
Java 2 version 1.4 compatible JVM	Application running platform
Program D	Executing the AIML file for application
XML	For system configuration coding
AIML	For algorithm coding (further details below)
Macromedia Dreamweaver MX	For code editing
Microsoft Word	Preparation of documents
Microsoft Project	For schedule planning
Internet Explorer	For browsing the Internet in analysis phase
Microsoft Power Point	For presentation/learning slide
Adobe Photoshop 7.0	For image editing
Paint	For image editing

Table 3.1: Software for the Development and Their Usage

The project will be implemented using AIML (Artificial Intelligence Markup Language a derivative of XML (Extended Markup Language) to express the knowledge base upon which the Chatterbot's parsing of questions and construction of responses are based. There are several advantages in using AIML:

- AIML is open source software, a great advantage in for the project since the software can be freely used and, if necessary, modified. It is proposed that the AIML knowledge bases created should be open source and hence freely available to all.
- AIML already has a Web Interface; this can be easily modified so that any additional features can be incorporated.
- AIML is not a rule-based system. There is no need to go through a very complicated elicitation process with experts to produce the knowledge base. Moreover, the knowledge base can be added to and amended by later developer.

3.3.1.2 Development and Construction Hardware

Table 3.2 shows the hardware requirement of the computer for the development of the Malay Chatterbot:

Device	Requirement
Operating System	Microsoft Windows
Processor	Intel® Pentium® 4 CPU 2.4 GHz
Memory	128 MB of memory
Disk Space	20GB of free space
Other Peripherals	Screen (1024 x 768), Keyboard, Mouse, CD-ROM drive

Table 3.2 Minimum Hardware Requirements

3.3.2 USER'S SPECIFICATION

3.3.2.1 Software

- Java 2 version 1.4 compatible JVM (examples are the Sun JRE (Java Runtime Edition) or SDK (Software Development Kit). However, if users are communicating with the Malay Chatterbot through Internet, they do not need to install Java. All they need is a Web Browser such as;
- Internet Explorer 5.0 or above

3.3.2.2 Hardware

The table below shows the minimum hardware requirement that the user must have in order to use or communicate with the chatterbot:

Note: The list reflects the minimum requirement for personal computer

Hardware	Specification
Operating system	Windows 98 or higher
Processor	Pentium Celeron or higher
RAM	64 MB or higher
Disk Space	2 Gb or higher
Monitor	14 inch (1024 x 768 resolution)
Graphic card	At least High Color (16 bit)
Mouse and Keyboard	Any standard

Table 3.3: User's Hardware Specification

CHAPTER 4
RESULTS AND DISCUSSION

4.1 INTRODUCTION

This section will discuss the findings on the algorithms and techniques of AIML that has been identified and learned by the author in constructing A.M.I.R.A. the Malay Chatterbot as well as on the research area.

4.2 FINDINGS

4.2.1 The AIML

Artificial Intelligence Markup Language, abbreviated AIML, describes a class of data objects called AIML objects and partially describes the behavior of computer programs that process them. AIML is a derivative of XML, the Extensible Markup Language. By construction, AIML objects are conforming XML documents, although AIML objects may also be contained within XML documents.

AIML objects are made up of units called **topics** and **categories**, which contain either, parsed or unparsed data. Parsed data is made up of characters, some of which form character data, and some of which form AIML elements. AIML elements encapsulate the stimulus-response knowledge contained in the document. Character data within these elements is sometimes parsed by an **AIML interpreter** a software module used to read AIML objects and provide application-level functionality based on their structure. It is also sometimes left

unparsed for later processing by a **Responder**, A software module that handles the human-to-bot or bot-to-bot interface work between an AIML interpreter and its object(s).

The design goals for AIML are:

- AIML shall be easy for people to learn.
- AIML shall encode the minimal concept set necessary to enable a stimulus-response knowledge system modeled on that of the original A.L.I.C.E.
- AIML shall be compatible with XML.
- It shall be easy to write programs that process AIML documents.
- AIML objects should be human-legible and reasonably clear.
- The design of AIML shall be formal and concise.
- AIML shall not incorporate dependencies upon any other language.

The basic unit of knowledge in AIML is called a ***category***. Each category consists of an input question, an output answer, and an optional context. The question, or stimulus, is called the ***pattern.*** The answer, or response, is called the ***template*** (see figure 4.1 for a simple example of AIML category). The two types of optional context are called "**that**" and "**topic.**" The AIML pattern language is simple, consisting only of words, spaces, and the wildcard symbols _ and *. The words may consist of letters and numerals, but no other characters. The pattern language is case invariant. Words are separated by a single space, and the wildcard characters function like words.

```
<category>
<pattern> SELAMAT TINGGAL </pattern>
<template> Jumpa lagi. </template>
</category>
```

Figure 4.1 AIML category

To visually illustrate how AIML categories work, a diagram of conversation network is shown below (figure 4.2). The diagram illustrates a conversation between user and chatterbot that involves 'pets' or 'haiwan peliharaan' topic.

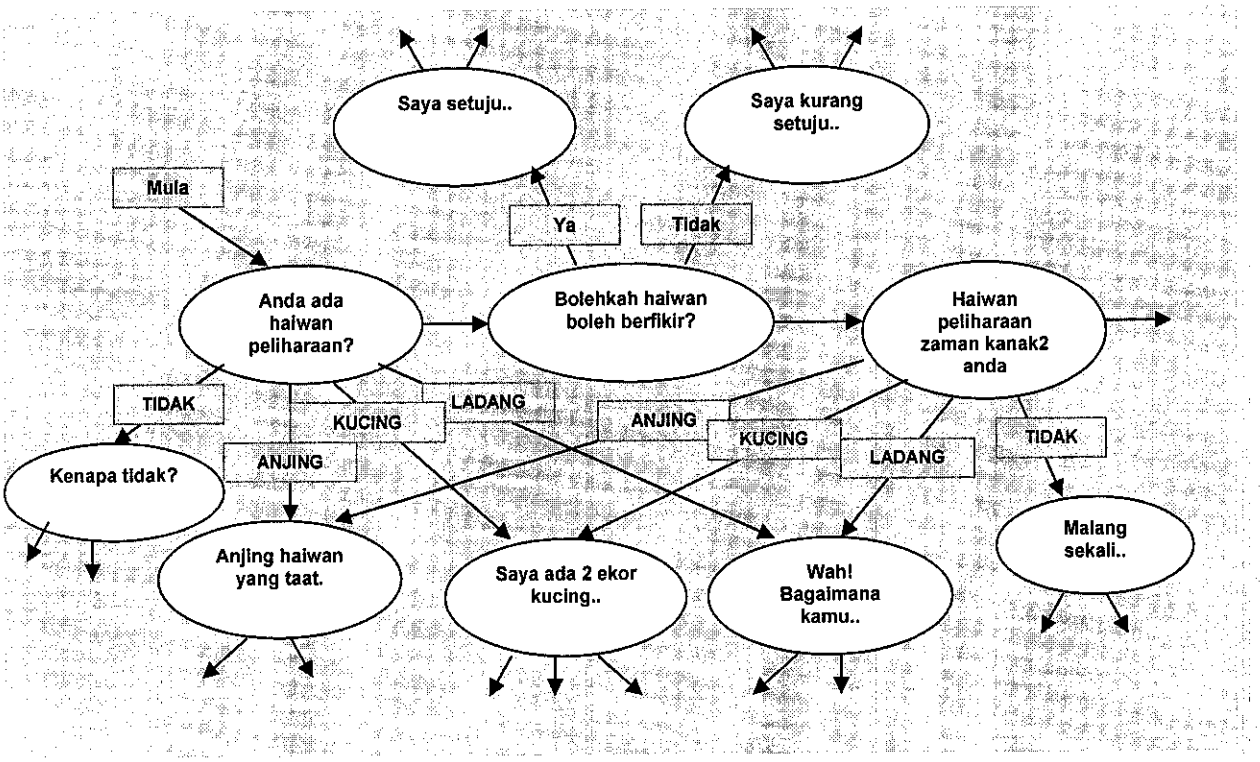


Figure 4.2 Portion of conversation network

When a user response to a question by a chatterbot for example, "Anda ada haiwan peliharaan?" the next response from the chatterbot will be based on how the user response to that particular question. If the user response is "tidak", then the chatterbot will reply "Kenapa tidak?" So, for every question and answer pairs, the author need to write it in AIML using various tags that is most suitable. In the next section, the author will explain in more details all the important algorithms and techniques in AIML that has been used to develop A.M.I.R.A. the Malay Chatterbot.

4.2.2. General types of AIML category

Given only the <pattern> and <template> tags, one of the most important type of AIML category will be atomic. "**Atomic**" categories are those with atomic patterns, i.e. the pattern contains no wild card "*" or "_" symbol. Atomic categories are the easiest, simplest categories to add in AIML. An example of atomic categories is:

```
<category>
<pattern>APA ITU LINUX</pattern>
<template>Linux merupakan salah satu sistem operasi percuma.
</template>
</category>
```

Figure 4.3 AIML Atomic Category

The above category illustrate the following:

- Matches the user input of "Apa itu Linux"
- Sends the user response: "Linux merupakan salah satu sistem operasi percuma".

Next general types of AIML category will be the "default" category. The name "**default** category" derives from the fact that its pattern has a wildcard "*" or "_". The ultimate default category is the one with <pattern>*</pattern>, which matches any input. These default responses are often called "pickup lines" because they generally consist of leading questions designed to focus the client on known topics. The more common default categories have patterns combining a few words and a wild card. For example the category:

```
<category>
<pattern>SAYA PERLUKAN BANTUAN *</pattern>
<template>Boleh awak nyatakan apa masalah awak?
</template>
</category>
```

Figure 4.4 AIML Default category

This category responds to a variety of inputs from "Saya perlukan bantuan menyelesaikan masalah matematik" to" Saya perlukan bantuan kaunseling." Putting aside the philosophical question of whether the chatterbot really "understands" these inputs, this category elucidates a coherent response from the client, who at least has the impression of the robot understanding the client's intention. Default categories show that writing AIML is both an art and a science. Writing good AIML responses is more like writing good literature, perhaps drama, than like writing computer programs.

4.2.3 AIML tags

Recursion

Understanding recursion is important to understanding AIML. "Recursion" means applying the same solution over and over again, to smaller and smaller problems, until you reduce the problem to its simplest form. AIML uses the tags **<sr/>** and **<srai>** to implement recursion. The botmaster uses these tags to tell the robot how to respond to a complex sentence by breaking it down into the responses to simpler ones. No agreement exists about the meaning of the acronym <srai>. The "A.I." stands for artificial intelligence, but "S.R." may mean "stimulus-response," "syntactic rewrite," "symbolic reduction," "simple recursion," or "synonym resolution." The disagreement over the acronym reflects the variety of applications for <srai> in AIML. Each of these is described in more detail in a subsection below:

- **Symbolic Reduction**: Reduce complex grammatical forms to simpler ones.
- **Divide and conquer**: Split an input into two or more subparts, and combine the responses to each.
- **Synonyms**: Map different ways of saying the same thing to the same reply.
- Detecting keywords anywhere in the input.
- Any combination of above.

4.2.3.1 Symbolic Reduction

Symbolic reduction refers to the process of simplifying complex grammatical forms into simpler ones. Usually, the atomic patterns in categories storing chatterbot knowledge are stated in the simplest possible terms, for example we tend to prefer patterns like "WHO IS SOCRATES" to ones like "DO YOU KNOW WHO SOCRATES IS" when storing biographical information about Socrates.

The table 4.1 will show how the steps of how the symbolic reduction works.

Step	normalized input	matching pattern	template	response
1.	AMIRA BOLEH AWAK BERITAHU SAYA APA ITU LINUX SEKARANG	_SEKARANG	Baiklah. <sr/>	
2.	AMIRA BOLEH AWAK BERITAHU SAYA APA ITU LINUX	_<name/>	<sr/>	
3.	BOLEH AWAK BERITAHU SAYA APA ITU LINUX	BOLEH AWAK *	<sr/>	
4.	BERITAHU SAYA APA ITU LINUX	BERITAHU SAYA *	<sr/>	
5.	APA ITU LINUX	APA ITU LINUX		Baiklah. Linux Merupakan salah satu sistem operasi percuma.

Table 4.1 Symbolic reduction process

The chatterbot has no specific response to the pattern " Amira boleh awak beritahu saya apa itu Linux sekarang." Instead, the chatterbot builds its response to the client input in

five steps. This simple sentence activated a sequence of five categories linked by <srai> tags.

In step 1, Some AIML templates combine the <sr/> with an ordinary text response, as with the pattern "_ SEKARANG". The phrase "Baiklah." becomes part of any reply ending in "SEKARANG ".

Steps 1 through 4 illustrate the common AIML templates that use the abbreviated <sr/> tag. (<sr/> = <srai><star/></srai>). The categories with the patterns "_ <name/>" and 'BOLEH AWAK *" simply reduce the sentence to whatever matches the "*", as illustrated by steps 3 and 4.

The category in step 5 will simply reply to the user the definition of Linux that has been define in atomic category.

.2.3.2 Divide and Conquer

Many individual sentences may be reduced to two or more sub sentences, and the reply formed by combining the replies to each. A sentence beginning with the word "Ya" for example, if it has more than one word, may be treated as the sub sentence "Ya" plus whatever follows it.

```
category>
pattern>YA *</pattern>
emplate><srai>YA</srai> <sr/></template>
category>
```

4.2.3.3 Synonyms

The AIML standard does not permit more than one pattern per category. Synonyms are perhaps the most common application of <srai>. Many ways to say the same thing reduce to one category, as illustrated by figure 4.5 below:

```
<category>
<pattern>HELLO</pattern>
<template>Hai! Apa khabar?</template>
</category>

<category>
<pattern>HI</pattern>
<template><srai>HELLO</srai></template>
</category>

<category>
<pattern>HAI DISANA</pattern>
<template><srai>HELLO</srai></template>
</category>

<category>
<pattern>HAI</pattern>
<template><srai>HELLO</srai></template>
</category>

<category>
<pattern>HOLA</pattern>
<template><srai>HELLO</srai></template>
</category>
```

Figure 4.5 Sample of AIML synonyms category

4.2.3.4 Keywords

Frequently we would like to write an AIML template which is activated by the appearance of a keyword anywhere in the input sentence. The general format of four AIML categories is illustrated by this example borrowed from ELIZA:

```
<category>
<pattern>IBU</pattern>
<template> Beritahu saya dengan lebih lanjut tentang keluarga anda. </template>
</category>
```

```
<category>
<pattern>_ IBU </pattern>
<template><srai> IBU </srai></template>
</category>

<category>
<pattern> IBU _</pattern>
<template><srai> IBU </srai></template>
</category>

<category>
<pattern>_ IBU *</pattern>
<template><srai> IBU </srai></template>
</category>
```

The first category both detects the keyword when it appears by itself, and provides the generic response. The second category detects the keyword as the suffix of a sentence. The third detects it as the prefix of an input sentence, and finally the last category detects the keyword as an infix. Each of the last three categories uses <srai> to link to the first, so that all four cases produce the same reply, but it needs to be written and stored only once.

4.2.4 Input Normalization

An AIML interpreter must perform a "normalization" function on all inputs before attempting to match. The minimum set of normalizations is called **pattern-fitting normalizations**. Additional normalizations performed at user option are called **sentence-splitting normalizations** and **substitution normalizations** (or just "substitutions").

If an AIML interpreter performs substitution normalizations on the input, then it must be performed first.

If an AIML interpreter performs sentence-splitting normalizations on the input, then it must be performed on the output of the substitution normalization process.

The pattern-fitting normalization process receives the output of the sentence-splitting normalization process (if any), or the output of the substitution normalization process (if

any, and if no sentence-splitting normalization is performed), or the direct input (if no sentence-splitting or substitution normalization is performed).

4.2.4.1 Substitution normalizations

Substitution normalizations are heuristics applied to an input that attempt to retain information in the input that would otherwise be lost during the sentence-splitting or pattern-fitting normalizations. For example:

- Abbreviations such as "Pn." may be "spelled out" as "Puan" to avoid sentence-splitting at the period in the abbreviated form
- Web addresses such as "http://alicebot.org" may be "sounded out" as "http alicebot dot org" to assist the AIML author in writing patterns that match Web addresses.
- Filename extensions may be separated from their file names to avoid sentence-splitting (".zip" to " zip")

4.2.4.2 Sentence-splitting normalizations

Sentence-splitting normalizations are heuristics applied to an input that attempt to break it into "sentences". The notion of "sentence", however, is ill-defined for many languages, so the heuristics for division into sentences are left up to the developer. Commonly, sentence-splitting heuristics use simple rules like "break sentences at periods", which in turn rely upon substitutions performed in the substitution normalization phase, such as those which substitute full words for their abbreviations.

4.2.4.3 Pattern-fitting normalizations

Pattern-fitting normalizations are normalizations that remove from the input characters that are not normal characters.

Pattern-fitting normalizations on an input must remove all characters that are not normal characters. For each non-normal character in the input,

- if it is a lowercase letter, replace it with its uppercase equivalent
- if it is not a lowercase letter, replace it with a space

4.2.5. Pattern Expression matching behavior

Each input to the AIML interpreter must pass through the input normalization process described above, in which (at the very minimum) the input will be processed according to the description of pattern-fitting normalizations.

In the case that sentence-splitting normalization is used by the AIML interpreter, a single input may be subdivided into several "sentences". The AIML interpreter must process each sentence of the input by producing an input path from it.

An input match path has three components, whose order is mandatory, and which correspond to the three components of a load-time match path:

1. **pattern**: the normalized input
2. **that**: the previous chatterbot output, normalized according to the same rules as in input normalization. If there was *no* previous chatterbot output, or the previous chatterbot output was unavailable, the value of the input path that is *.
3. **topic**: the current value of the topic predicate. If the topic predicate has *no* value, then the value of the input path topic is *.

4.2.5.1 Explanation via implementation description: Graphmaster

Matching behavior can be described in terms of the class Graphmaster (see Figure 4.6 below) which is a common implementation of the AIML pattern expression matching behavior:

1. Given:
    a. an input starting with word X, and
    b. a Nodemapper of the graph:
2. Does the Nodemapper contain the key _? If so, search the subgraph rooted at the child node linked by _. Try all remaining words of the input following X to see if one matches. If no match was found, try:
3. Does the Nodemapper contain the key X? If so, search the subgraph rooted at the child node linked by X, if no match was found, try:
4. Does the Nodemapper contain the key *? If so, search the subgraph rooted at the child node linked by *. Try all remaining words of the input following X to see if one matches. If no match was found, go back up the graph to the parent of this node, and put X back on the head of the input.
5. If the input is null (no more words) and the Nodemapper contains the <template> key, then a match was found. Halt the search and return the matching node.

If the root Nodemapper contains a key "*" and it points to a leaf node, then the algorithm is guaranteed to find a match.

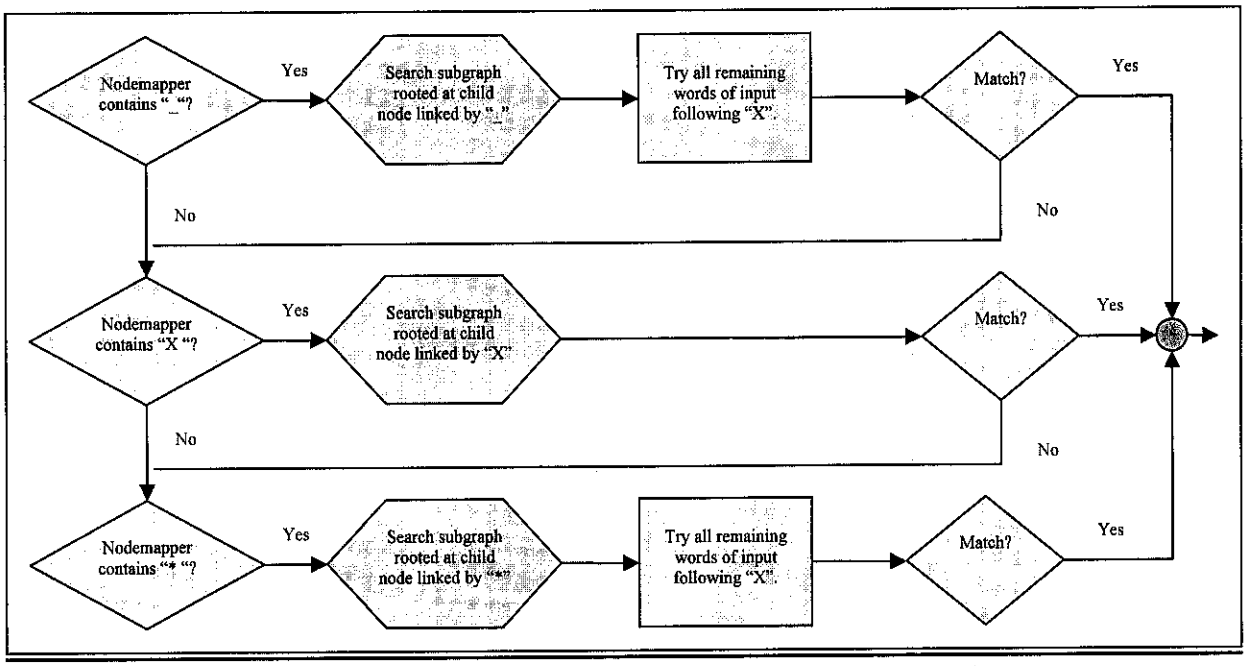


Figure 4.6 The Graphmaster

Note that:

1. The patterns need not be ordered alphabetically or according to any other complete system, only partially ordered so that _ comes before any word and * after any word.
2. The matching is word-by-word, not category-by-category.
3. The algorithm combines the input pattern, the <that> pattern, and the <topic> pattern into a single "path" or sentence such as: "PATTERN <that> THAT <topic> TOPIC" and treats the tokens <that> and <topic> like ordinary words. The PATTERN, THAT and TOPIC patterns may contain multiple wildcards.
4. The matching algorithm is a highly restricted version of depth-first search, also known as backtracking.

4.3 DISCUSSION

Looking back at the findings, the author could conclude that there were various algorithms and techniques of AIML could be apply to construct a Malay Chatterbot that communicate with a well arranged and grammatically correct sentence with the user. However, the author also found out that developing a Malay Chatterbot are much more complicated than building an English Chatterbot as a single question in Malay could be formed in many ways. Which is the arrangement of words to ask a same question could be structured in many different ways. This has resulted the author to spend a much longer time in defining the AIML categories of synonyms. To construct a chatterbot that could talk about anything, like ALICE in Bahasa Melayu will take a very long time to implement. With the time constrain to complete this final year project (FYP) that would be the main reason the author need to narrow down the scope or topic to only talking and informing the user about UTP for A.M.I.R.A the Malay Chatterbot. All the suitable AIML techniques that has been discovered to develop a chatterbot will be applied to define categories that would be mainly about UTP. The author also might add a small portion of other general and interesting topic to the brain of A.M.I.R.A.

From the result and discussion, the author believed that AIML is a good technique to develop a Malay Chatterbot and should be used by other developer that is interested in developing a chat robot. Implementation language AIML is based on the notion that while human thinking is quite complex, it might be just "good enough" to simulate thinking by providing "enough" response patterns to potential inquiries. It is also wise to use the AIML <srai> tag because it simplifies and combines four important chat robot operations; which is:

- Maps multiple patterns to the same response.
- Reduces a complex sentence structure to a simpler form.
- Diminishes the need for multiple-wildcard input patterns.
- Translates state-dependent inputs into simpler stimulii.

CHAPTER 5
CONCLUSION AND RECOMMENDATION

5.1 CONCLUSION

After completing the application, the author thinks this Malay Chatterbot, also known as A.M.I.R.A with further enhancements has a good reputation to be applied to any websites in Malaysia that has been developed and using Bahasa Melayu as a medium to communicate with the user. This is because using all the algorithms and techniques that have been applied to A.M.I.R.A, the chatterbots successfully communicate in Bahasa Melayu with well arranged words and grammatically correct sentences. Websites of local companies, government agencies, local universities and colleges, IPTA and any other organizations should try embedding the chatterbot to their websites to make it more interesting and interactive. A current trend in modern HCI is representing Embodied Conversational Agents (ECAs) that is designed to run on the Web. They are virtual 3D human-like front ends coupled with software agents like chatterbot that are able to engage in a conversation with a user and execute complex tasks, such as, for example, searching for some specific information or ordering some items from the catalogue of an online shop.

Nowadays, the technology is fast and easy to use which encouraging people to choose any application that can ease them in doing their task. Moreover, people already knew that Web technology is a best medium to promote the product, education purposes or other purposes because there are millions of people using it at each minute The metaphor of a face-to-face conversation greatly increases the feeling of presence during the interaction and eliminates the need of learning where to find specific widgets accomplishing a single task that is really needed.

. So, implementing a Malay Chatterbot in the local websites would be a wise decision for any organizations as they could develop the chatterbot to help user to navigate their websites or just simply attract their user by getting the chatterbot to chat with the user. A Malay Chatterbot would be a good investment as it could attract local user that is not very fluent in English to visit their websites and communicate with the chatterbot. Users that have a good time browsing one website definitely will browse again and that would be a ɔusiness opportunity for the organizations.

.2 RECOMMENDATION

'he author recommends this application to be widely exposed to the local organizations ı Malaysia, who use Bahasa Melayu as their medium to communicate with their user in ıeir websites. Any organization such as government agencies, education institutes, local ısiness companies and any other entities that their targeted user would be Malaysian ıd people that understand Malay should consider the use of Malay chatterbot to enhance eir website. Users who are not very fluent in English, or could not understand English all would appreciate this application and this problem can be solved by exposing this plication to them. The chatterbot could be develop to overcome the classical limitations the non-interactive interfaces, which cause inexpert users to suffer from an ɔomfortable interaction experience.

ıs chatterbot or software agent also could be develop to understand user's wishes, ıverse with them, find information and execute non-trivial tasks, replacing buttons ssing, menu choices and hyperlinks clicks, which often contribute to undesired ırmation overload.

future enhancement, this application will be smarter and more interactive than current ication if the chatterbot is embedded with a virtual 3D human-like front ends like a ıvatar, 3D animated representations of the user. This application when coupled with nimated 2D/3D look-and-feel will embody their intelligence via a face or an entire

body. This way it is possible to enhance user trust and satisfaction, giving users some sort of illusion of life, as cartoons, videogames or animated movies are able to do.

Beside an animated representation, it would also be more interesting if the Malay Chatterbot could capture the user input using other devices such as microphones and be able to understand the input and respond to it correctly. This feature would require optional support for speech recognition, which allows applications to respond to voice commands using synthesized speech, recorded audio or text.

This chatterbot definitely an interesting features that could be use by local organizations to attract users to their websites. Moreover, its ability to converse or communicate in Bahasa Melayu would be the main reason why local users would be more attracted to use them

REFERENCES

[1] Laven, S.: Online at http://www.simonlaven.com

[2] Turing, A.M. (1950). Computing machinery and intelligence. *Mind, 59*, 433-560.

[3] Wallace, R. S.: Don't read me – A.L.I.C.E. and AIML documentation. Online at http://www.alicebot.com/dont.html

[4] Wallace, R. S.: The Anatomy of A.L.I.C.E. Online at http://www.alicebot.com/

[5] Weizenbaum, J. (1966). Eliza – A computer program for the study of natural nguagecommunication between man and machine. *Communication of the ACM, 9*(1), 5-45

] Foner, L. N.: What's an agent, anyway? A sociological case study. Online at http://foner.www.media.mit.edu/people/foner/Julia/Julia.html.

] De Angeli, A., Lynch, P. & Johnson, G. (2001). Personifying the e-market: A mework for social agents.

Leaver ton, M. (2000). Recruiting the chatterbots. *Cnet Tech Trends*, 10/2/00. Online http://cnet.com/techtrends/0-1544320-8-2862007-1.html.

Dehn, D. M. & van Mulken, S. (2000). The impact of animated interface Agents: A iew of empirical research. *International Journal of Human-Computer Studies, 52*(1), 2.

] Laurel, B. (1997). Interface agents: Metaphors with Character. In J. M. Bradshaw .) *Software Agents* (pp. 67- 77). Menlo Park, CA: AAAI Press/The MIT Press.

Online at http://www.geocities.com/brizglace/documentation.htm

Wallace, R. S.: Artificial Intelligence Markup Language (2001). Online at: ://alicebot.org/TR/2001/WD-aiml-1.0.1-20011025-006.html

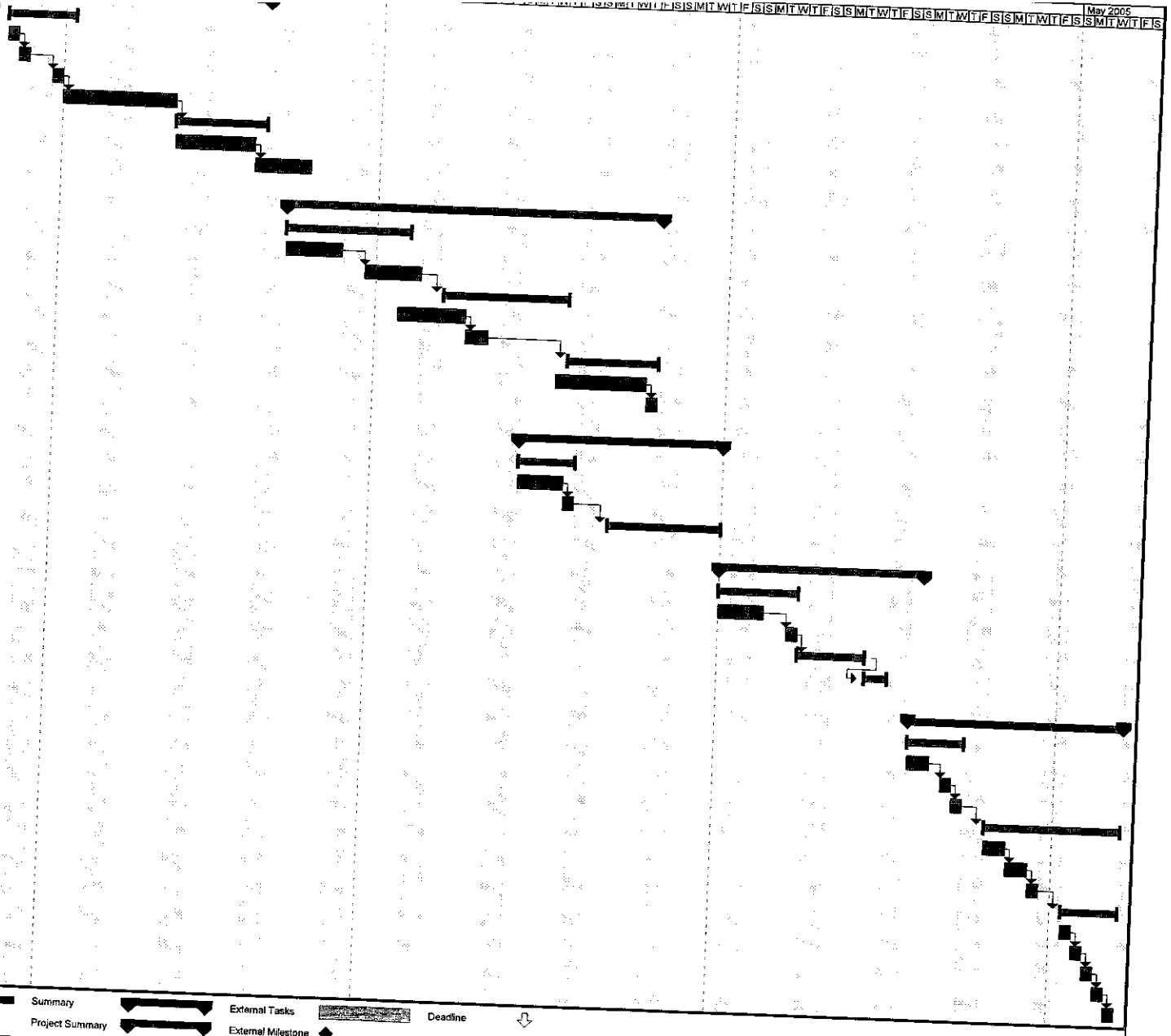
F. Abbattista, G. Catucci, G. Semeraro and F. Zambetta.(2004). SAMIR: A smart 3d stant on the Web.

APPENDICES

APPENDIX 1

GANTT CHART

3	1.1.1 Define the Problem Statement	4 days	Thu 1/27/05	Tue 2/1/05
4	1.1.2 Identify the objectives of the project	1 day	Thu 1/27/05	Thu 1/27/05
5	1.1.3 Determine the scope of study	1 day	Fri 1/28/05	Fri 1/28/05
6	1.1.4 Research	1 day	Mon 1/31/05	Mon 1/31/05
7	**1.2 Preliminary Report**	8 days	Tue 2/1/05	Thu 2/10/05
8	1.2.1 Preparation of Preliminary Report	6 days	Fri 2/11/05	Fri 2/18/05
9	1.2.2 Submit Preliminary Report	5 days	Fri 2/11/05	Thu 2/17/05
10		3 days	Fri 2/18/05	Tue 2/22/05
11	**PROJECT DESIGN**			
12	**2.1 User Requirement**	25 days	Mon 2/21/05	Fri 3/25/05
13	2.1.1 Define content requirements	9 days	Mon 2/21/05	Thu 3/3/05
14	2.1.2 Define user requirements	5 days	Mon 2/21/05	Fri 2/25/05
15	**2.2 System Design**	5 days	Mon 2/28/05	Fri 3/4/05
16	2.2.1 Design content flow	9 days	Mon 3/7/05	Thu 3/17/05
17	2.2.2 Design system structure	4 days	Thu 3/3/05	Tue 3/8/05
18	**2.2 Progress Report**	2 days	Wed 3/9/05	Thu 3/10/05
19	2.2.1 Preparation of Progress Report	6 days	Fri 3/18/05	Fri 3/25/05
20	2.2.2 Submit Progress Report	6 days	Thu 3/17/05	Thu 3/24/05
21		1 day	Fri 3/25/05	Fri 3/25/05
22	**PROJECT DEVELOPMENT**			
23	**3.1 Interface**	14 days	Mon 3/14/05	Thu 3/31/05
24	3.1.1 System configuration	5 days	Mon 3/14/05	Fri 3/18/05
25	3.1.2 Develop user interface	4 days	Mon 3/14/05	Thu 3/17/05
26	**3.2 Coding**	1 day	Fri 3/18/05	Fri 3/18/05
27		8 days	Tue 3/22/05	Thu 3/31/05
28	**PROJECT TESTING**			
29	**4.1 Testing**	12 days	Fri 4/1/05	Mon 4/18/05
30	4.1.1 Unit testing	5 days	Fri 4/1/05	Thu 4/7/05
31	4.1.2 System testing	2 days	Fri 4/1/05	Mon 4/4/05
32	**4.2 Modify coding**	1 day	Thu 4/7/05	Thu 4/7/05
33	**4.3 Re-test modified coding**	4 days	Fri 4/8/05	Wed 4/13/05
34		2 days	Thu 4/14/05	Fri 4/15/05
35	**FINAL DELIVERABLE**			
36	**5.1 Documentation**	15 days	Mon 4/18/05	Fri 5/6/05
37	5.1.1 Finalize design	5 days	Mon 4/18/05	Fri 4/22/05
38	5.1.2 Preparation of final draft	2 days	Mon 4/18/05	Tue 4/19/05
39	5.1.3 Submit final draft	1 day	Thu 4/21/05	Thu 4/21/05
40	**5.2 Oral Presentation**	1 day	Fri 4/22/05	Fri 4/22/05
41	5.2.1 Preparation of presentation materials	10 days	Mon 4/25/05	Fri 5/6/05
42	5.2.2 Configure PC to demonstrate system	2 days	Mon 4/25/05	Tue 4/26/05
43	5.2.3 Final presentation	2 days	Wed 4/27/05	Thu 4/28/05
44	**5.3 Project Dissertation**	1 day	Fri 4/29/05	Fri 4/29/05
45	5.3.1 Finalize research	5 days	Mon 5/2/05	Fri 5/6/05
46	5.3.2 Finalize discussion and result	1 day	Mon 5/2/05	Mon 5/2/05
47	5.3.3 Finalize report	1 day	Tue 5/3/05	Tue 5/3/05
48	5.3.4 Binding	1 day	Wed 5/4/05	Wed 5/4/05
49	5.3.5 Submit FYP dissertation	1 day	Thu 5/5/05	Thu 5/5/05
		1 day	Fri 5/6/05	Fri 5/6/05



APPENDIX 2

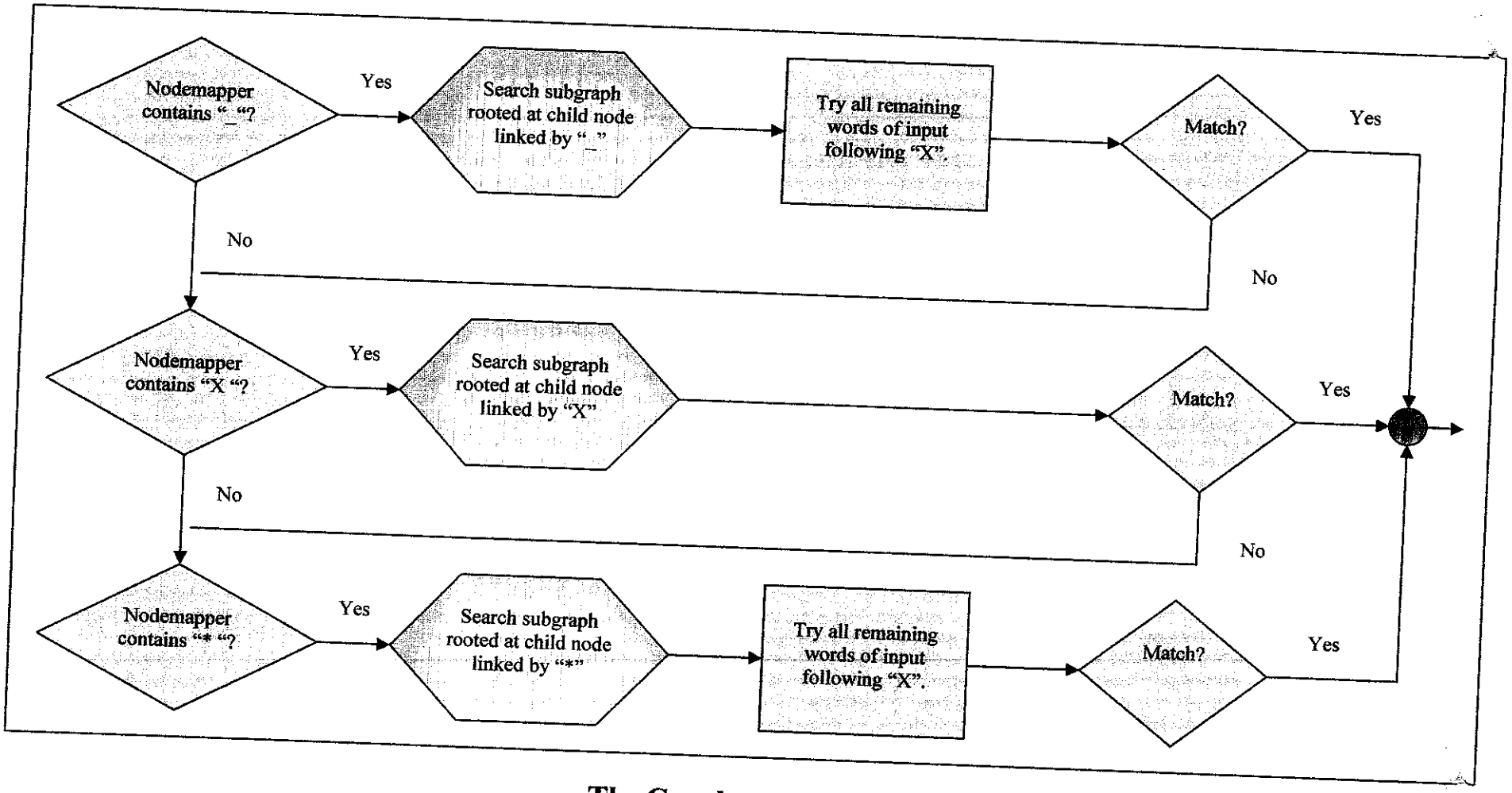
Program D Simple Console

File Actions Help

```
[23:44:05] Starting Alicebot Program D version 4.1.5
[23:44:05] Using Java VM 1.4.2_07-b05 from Sun Microsystems Inc.
[23:44:05] On Windows XP version 5.1 (x86)
[23:44:05] Predicates with no values defined will return: "undefined".
[23:44:05] Initializing Multiplexor.
[23:44:05] Loading Graphmaster.
[23:44:05] Starting up with "C:\ProgramD\conf\startup.xml".
[23:44:05] Configuring bot "A.M.I.R.A.".
[23:44:05] Loaded 448 input substitutions.
[23:44:06] Loaded 19 gender substitutions.
[23:44:06] Loaded 9 person substitutions.
[23:44:06] Loaded 60 person2 substitutions.
[23:44:06] Loaded 4 sentence-splitters.
[23:44:06] 1 bots thinking with 995 categories.
[23:44:06] Alicebot Program D (c) 1995-2002 A.L.I.C.E. AI Foundation
[23:44:06] All Rights Reserved.
[23:44:06] This program is free software; you can redistribute it and/or
[23:44:06] modify it under the terms of the GNU General Public License
[23:44:06] as published by the Free Software Foundation; either version 2
[23:44:06] of the License, or (at your option) any later version.
[23:44:06] Alicebot Program D version 4.1.5 Build [00]
[23:44:06] 995 categories loaded in 0.875 seconds.
[23:44:06] The AIML Watcher is not active.
[23:44:06] HTTP server listening at http://stila:2001
[23:44:06] Interactive shell: type "/exit" to shut down; "/help" for help.
[23:44:06] A.M.I.R.A.> Hello Aina dan terima kasih kerana sudi berbual dengan saya!
[23:44:06] [A.M.I.R.A.] Aina> hi
[23:44:09] A.M.I.R.A.> Hai, selamat berkenalan.
[23:44:09] [A.M.I.R.A.] Aina> Apa khabar?
[23:44:23] A.M.I.R.A.> Saya sihat, terima kasih. Anda bagaimana?
[23:44:23] [A.M.I.R.A.] Aina> ok
[23:45:00] A.M.I.R.A.> Semuanya baik-baik aje..
[23:45:00] [A.M.I.R.A.] Aina> panggil saya aliaa
[23:45:35] A.M.I.R.A.> Hai Aliaa.
[23:45:35] [A.M.I.R.A.] Aliaa> anda suka saya
[23:47:00] A.M.I.R.A.> Saya akan beitahu botmaster saya tentang itu,
[23:47:00] A.M.I.R.A.> .
[23:47:00] A.M.I.R.A.>
[23:47:00] A.M.I.R.A.> Secara terus terang saya katakan, saya tidak akan risau
tentanghal itu.
[23:47:00] [A.M.I.R.A.] Aliaa>
```

Enter

A.M.I.R.A's console



The Graphmaster

APPENDIX 4

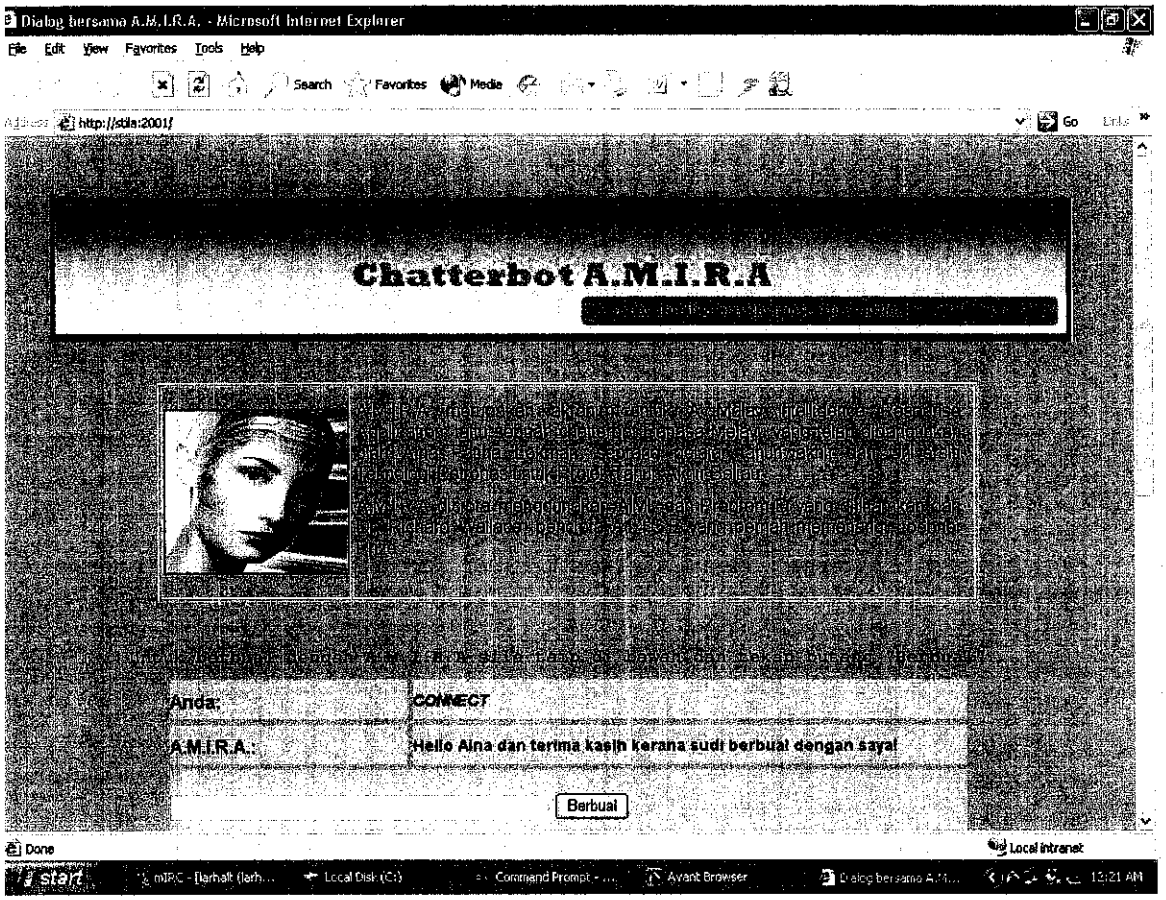
AIML TAG REFERENCE TABLE

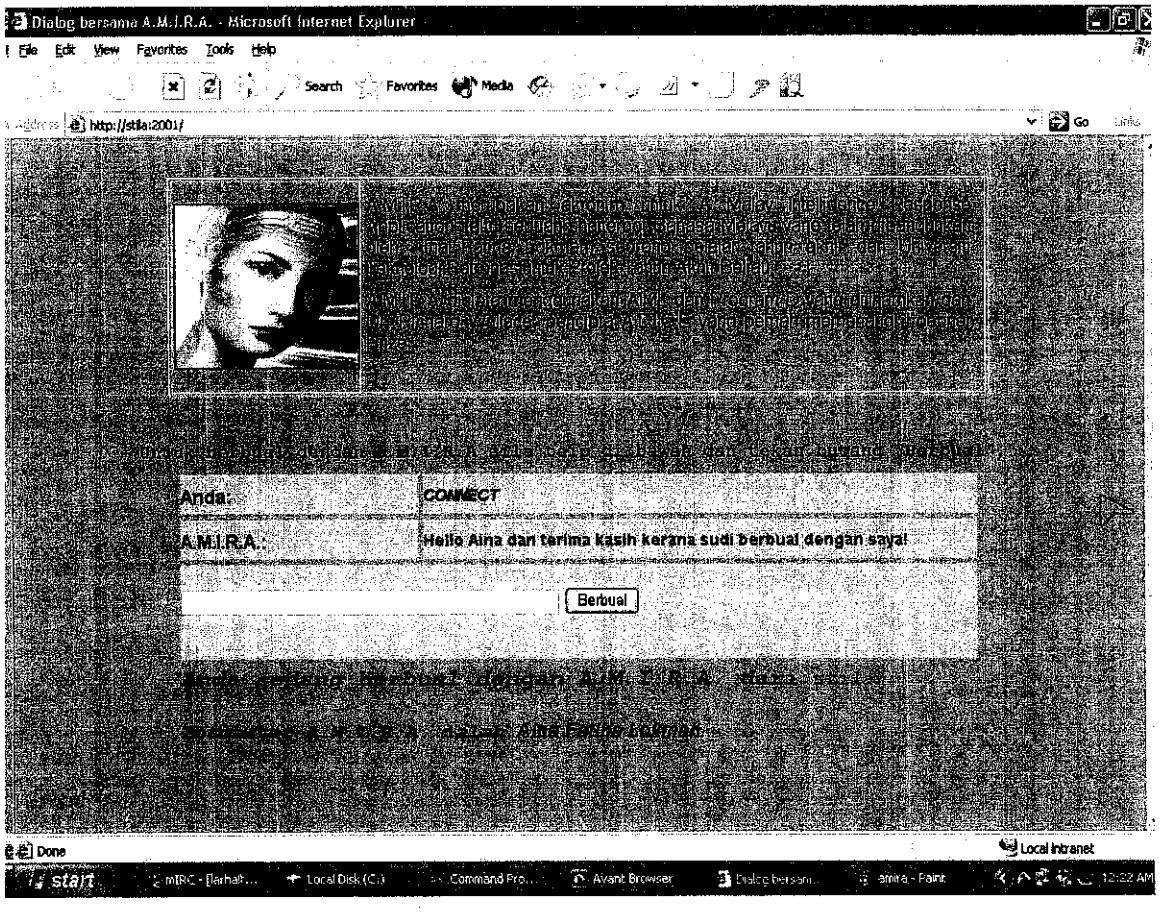
AIML 1.0	Tag Type	Note
<aiml>	AIML block delimeter	[Closing tags not shown]
<bot name="name"/>	Built-in bot parameter	may appear in pattern
<bot name="XXX"/>	Custom bot parameter	<srai>BOT XXX</srai>
<that index="2,1"/>	Built-in predicate	See Note 4.
<that index="nx,ny"/>	Built-in predicate	default "that"
<that>	AIML that pattern	contains AIML pattern
<category>	AIML category	
<input index="2"/>	Built-in predicate	
<input index="3"/>	Built-in predicate	
condition name="X" alue="Y">	Conditional branch	
condition>	Conditional branch	
gender>	Gender substitution	Exchange "he" and "she"
date/>	Built-in predicate	date and time
id/>	Built-in predicate	default "localhost"
get name="xxx"/>	Built-in predicate	default "X-person"
size/>	Built-in predicate	# of categories loaded
star index="n"/>	Built-in predicate	binding of *
hatstar index="n"/>	Built-in predicate	binding of * in that
get name="topic"/>	Built-in predicate	default "you"
opicstar index="n"/>	Built-in predicate	binding of * in topic
ersion/>	Built-in predicate	AIML program version
et name="xxx"/>	Custom predicate	Botmaster defined XXX, default (3)
ossip src="X"/>	Append to file	
arn>X</learn>	AIML loading	
name="X" value="Y">	Conditional branch item	used by <condition>
value="Y">	Conditional branch item	used by <condition name="X">
>	General list item	used by <random>, <condition>
ttern>	AIML Pattern	contains AIML pattern
rson/>	Prounoun transform macro	<person><get_star/></person>
rson2>	Prounoun transform	swap 1st & 2nd person
rson2/>	Prounoun transform	<person2><get_star/></person2>

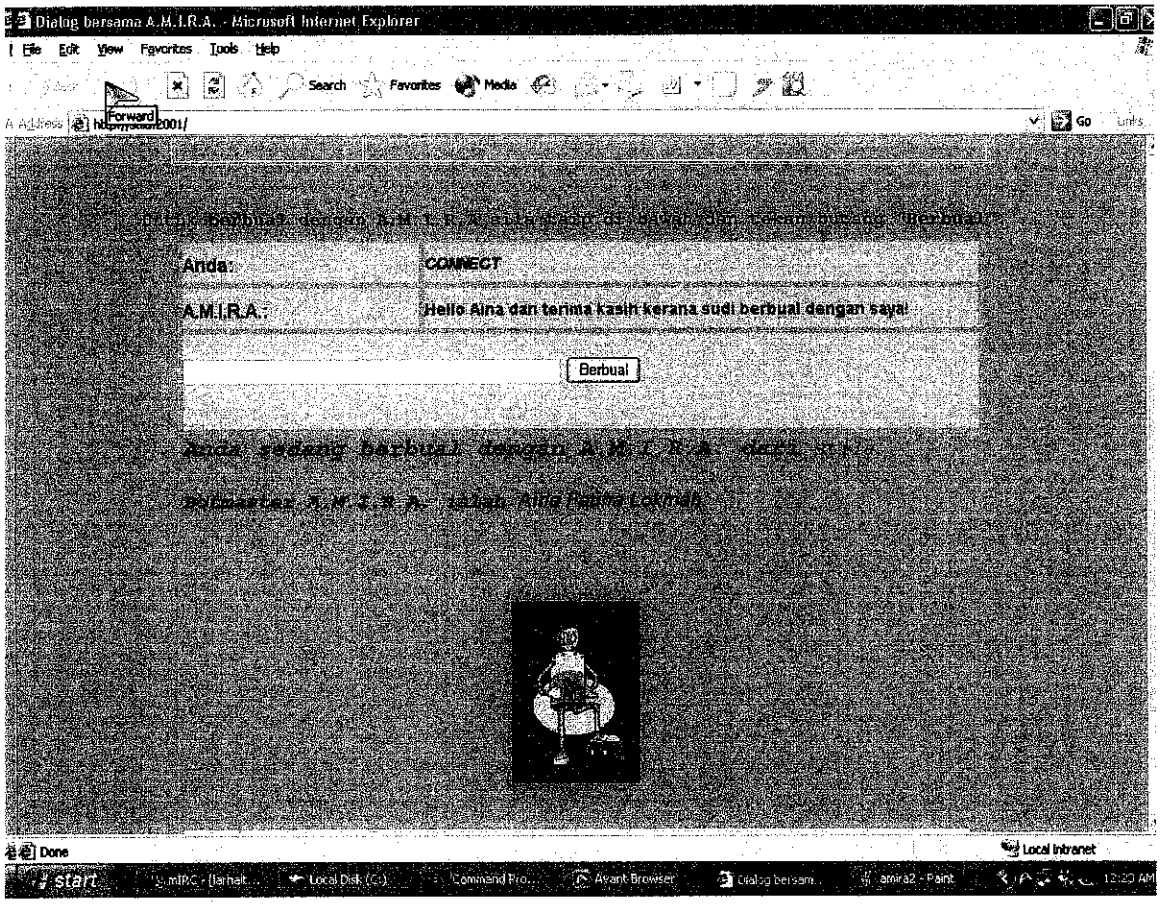
	macro	
<person>	Prounoun transform	swap 1st & 3rd person
<random>	Random selection	Random uniform selection
<set name="name">	Built-in predicate	returns contents
<set name="topic">	Built-in predicate	returns contents
<set name="XXX">	Custom predicate	See Note 3.
<sr/>	Recursion macro	<srai><get_star/></srai>
<srai>	Recursion	
<system>	Execute OS shell	platform-dependent
<template>	AIML template	
<think>	Nullify output	
<topic name="X">	AIML topic group	X is AIML pattern
<uppercase>	Text manipulation	convert all text to Uppercase
<lowercase>	Text manipulation	convert all text to Lowercase
<sentence>	Text manipulation	capitalize the first word
<formal>	Text manipulation	capitalize every word
<if name="X" value=Y">	Conditional branch	
<else>	Conditional branch	
<javascript>	AIMLScript	Javascript

APPENDIX 5

A.M.I.R.A. 's Browser screenshots




Dialog bersama A.M.I.R.A. - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Search Favorites Media
http://stila:2001/
Anda:
CONNECT
A.M.I.R.A.:
Hello Aina dan terima kasih kerana sudi berbual dengan saya!
Berbual
Done
Local Intranet
start
mIRC - [Default...
Local Disk (C:)
Command Pro...
Avant Browser
Dialog bersam...
anira - Paint
12:22 AM


Dialog bersama A.M.I.R.A. - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Search Favorites Media
Forward
Go
Anda:
CONNECT
A.M.I.R.A.:
Hello Aina dan terima kasih kerana sudi berbual dengan saya!
Berbual
Done
Local Intranet
start