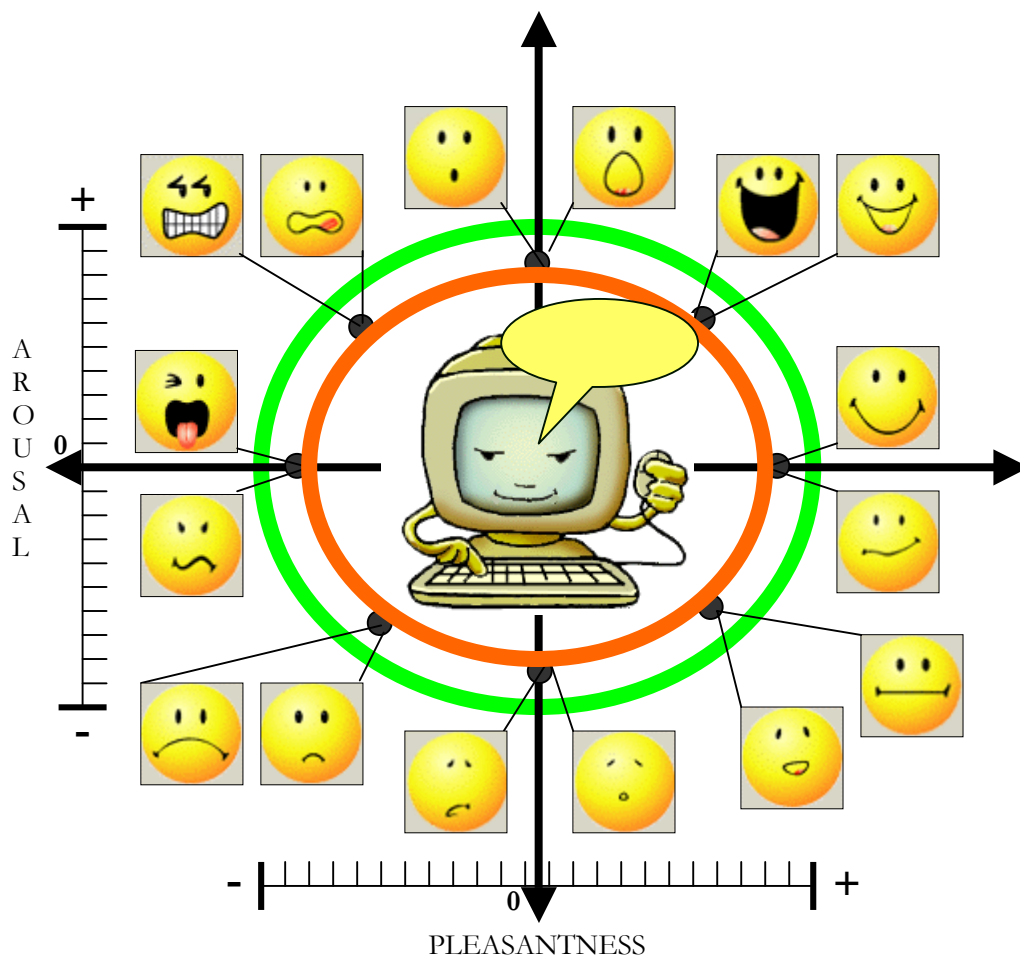


Thesis Report

# My\_Eliza

## A Multimodal Communication System

Siska Fitrianie





Graduation Committee:

1. Dr. Drs. L.J.M. Rothkrantz
2. Prof. Dr. Ir. E.J.H. Kerckhoffs
3. A. Wojdel MSc.
4. Ir. R.J. van Vark
5. Prof. Dr. H. Koppelaar



## Abstract

The field Natural Language Processing (NLP) is an active area of research. One of the possible applications of NLP systems is a *question answering system* – QA system. QA system is a program that attempts to simulate typed conversation in human natural language. In contrast with it, there are only a few researches involving research in Nonverbal Communication. Even though, it is proved that beyond the speech bodily activities give real substance to face-to-face interaction in real life conversation.

Motivated by this issue, in this report we investigate a design and implementation of a multimodal communication system, called my\_Eliza. The design of this system is based on Weizenbaum's Eliza [WEI66]. A user or client can communicate with the system using typed natural language. The system will reply by text-prompts and appropriate facial-expressions. In order to communicate using a nonverbal facial display, the developed system should be able to process natural language and emotional reasoning.

A first prototype as a proof of concept has been developed that consists of a dialog box, an emotional recognizer based on stimulus response, and a facial display generator. To implement the dialog box, the work of Richard Wallace, A.L.I.C.E system [WAL95], has been used as a starting point. My\_Eliza system has a rule engine that determines current system's affective state as reaction to the user's string input and conversation content. This report presents the result of a conversation between my\_Eliza and a human user.



## Table of Contents

ABSTRACT .....	III
TABLE OF CONTENTS.....	IV
FIGURES .....	VI
TABLES.....	IX
ACKNOWLEDGEMENTS .....	X
<b>CHAPTER 1 INTRODUCTION .....</b>	<b>1</b>
1.1 Who is Eliza?.....	1
1.2 Multimodal Communication System.....	4
1.3 The Aim of this Thesis: My_Eliza.....	5
1.4 Thesis Problem Definition.....	6
1.5 Thesis Overview .....	7
<b>CHAPTER 2 HUMAN-COMPUTER INTERACTION .....</b>	<b>8</b>
2.1 Chatterbot.....	8
2.2 Aspect of Natural Language Generation in a QA System.....	9
2.3 Aspects of Emotion Recognition and Facial Display Generation in a QA System .....	15
2.4 Nonverbal QA Systems.....	19
2.5 Related Work In Automatic Emotion Recognition and Facial Display Generation.....	21
2.6 A New Approach: My_Eliza .....	24
<b>CHAPTER 3 ARTIFICIAL INTELLIGENCE .....</b>	<b>27</b>
3.1 Natural Language Processing .....	27
3.2 Knowledge Based System .....	28
3.3 Intelligence Agent.....	29
3.4 AI Application Development .....	30
<b>CHAPTER 4 MY_ELIZA GLOBAL DESIGN .....</b>	<b>33</b>
4.1 System Architecture .....	33
4.2 Knowledge Extraction.....	38
4.2.1 <i>Databases in Natural Language Processing Layers</i> .....	38
4.2.1.1 List of Words/Phrases .....	39
4.2.1.2 List of Pattern Rules (System's Memory Structure).....	40
4.2.1.3 User Personal Database .....	45
4.2.1.4 System's GSP Database .....	45
4.2.2 <i>Databases and Knowledge Base in Facial Display Construction Layer</i> .....	46
4.2.2.1 Emotive Lexicon Dictionary.....	47
4.2.2.2 Affective Knowledge-Base.....	48
4.2.2.3 Facial Display Dictionary.....	49
4.3 Natural Language Processing Layers.....	49
4.4 Facial Display Construction Layers .....	53
<b>CHAPTER 5 MY_ELIZA DIALOG BOX: PREPROCESSING .....</b>	<b>60</b>
5.1 From Eliza to A.L.I.C.E.....	60
5.2 From A.L.I.C.E to My_Eliza.....	63
5.3 Implementation of My_Eliza's Dialog Box .....	69
5.3.1 <i>My_Eliza's Dialog Box Architecture</i> .....	71
5.3.2 <i>My_Eliza's Dialog Box Design</i> .....	72



---

5.3.3	<i>Build the List of Pattern Rules</i> .....	77
5.3.4	<i>My_Eliza's Dialog Box Implementation, Test and Result</i> .....	81
<b>CHAPTER 6 MY_ELIZA PROTOTYPE-1: STIMULUS-RESPONSE BASED REASONING</b> .....		<b>85</b>
6.1	From My_Eliza's Dialog Box to My_Eliza prototype-1 .....	85
6.2	Design of My_Eliza's Prototype-1 .....	89
6.2.1	<i>Context Model</i> .....	89
6.2.2	<i>Use case</i> .....	89
6.2.3	<i>Object Model</i> .....	91
6.2.4	<i>Dynamic Model</i> .....	94
6.2.5	<i>Affective Knowledge Base Design</i> .....	97
6.3	My_Eliza's Prototype-1 Implementation.....	99
<b>CHAPTER 7 TESTING OF MY_ELIZA PROTOTYPE-1</b> .....		<b>104</b>
7.1	Initial State of the System.....	104
7.2	Conversing with my_Eliza Prototype-1.....	106
<b>CHAPTER 8 CONCLUSION</b> .....		<b>124</b>
8.1	Evaluation.....	124
8.2	Future Work .....	126
<b>REFERENCES</b> .....		<b>128</b>
<b>APPENDIX A A.L.I.C.E AND AIML</b> .....		<b>131</b>
A.1	Alicebot Architecture.....	131
A.2	AIML Structure.....	132
A.3	Knowledge Extraction in A.L.I.C.E.....	138
<b>APPENDIX B PROGRAM D A.L.I.C.E</b> .....		<b>141</b>
B.1	Design.....	141
B.1.1	<i>Context Model</i> .....	141
B.1.2	<i>Use case</i> .....	141
B.1.3	<i>Object Model</i> .....	142
B.1.4	<i>Dynamic Model</i> .....	144
B.2	Compilation and Execution.....	144
B.3	Important Files and Folders .....	146
<b>APPENDIX C AFFECTIVE KNOWLEDGE BASE</b> .....		<b>150</b>
C.1	Preference Rules for Concern of the other Knowledge Base .....	150
C.2	Preference Rules of Cognitive Reasoning Knowledge Base.....	152
<b>APPENDIX D SYMBOLS AND NOTATIONS</b> .....		<b>161</b>
D.1	Flowcharting.....	161
D.2	UML Diagram in My_Eliza Design.....	162
D.2.1	<i>Use Case Diagram</i> .....	162
D.2.2	<i>Class Diagram</i> .....	163
D.3	Event Transition Diagram .....	164
D.4	State Transition Diagram .....	165
<b>APPENDIX E SCIENTIFIC PAPER</b> .....		<b>167</b>



## Figures

Figure 1- 1 (a) Original Eliza's basic algorithm and (b) keywords, decomposition rules, reassemble rules association in the script .....	2
Figure 1- 2 Example of reply construction in Eliza .....	2
Figure 2- 1 Turing test.....	10
Figure 2- 2 Memory Structure of Eliza (a) and A.L.I.C.E (b).....	13
Figure 2- 3 Example of one unit memory structure of A.L.I.C.E .....	14
Figure 2- 4 Example of A.L.I.C.E's memory unit for storing information.....	14
Figure 2- 5 Rusell's emotion-related word two-dimensional scaling .....	17
Figure 2- 6 The 'emotion circle' of Schlosberg (reproduced from [HEN98]) .....	17
Figure 2- 7 Some facial displays corresponding to emotions (disgust, sadness, surprise, happiness, angry and smile) .....	19
Figure 2- 8 My_Eliza's design requirements .....	25
Figure 2- 9 Comparing the process flow schema between Eliza and my_Eliza .....	25
Figure 3- 1 Basic schema of a knowledge-based system .....	28
Figure 3- 2 Agent interacts with environment through sensors and effectors .....	30
Figure 4- 1 My_Eliza three major implementation layers .....	33
Figure 4- 2 My_Eliza blackboard system architecture .....	34
Figure 4- 3 Flowchart of My_Eliza's stimulus-response for a user's string input.....	37
Figure 4- 4 Flowchart of My_Eliza's cognitive processing for user's string input.....	38
Figure 4- 5 Example of list of words/phrases .....	39
Figure 4- 6 Two forms of the AIML <that> tag .....	41
Figure 4- 7 Example of two memory units with the same topic in my_Eliza's list of pattern rules .....	43
Figure 4- 8 Example of atomic category .....	43
Figure 4- 9 Example of default category .....	44
Figure 4- 10 Example of recursive category .....	44
Figure 4- 11 Example of category for storing information during conversation .....	45
Figure 4- 12 Example of user's preferences, which are stored by my_Eliza per user id.....	45
Figure 4- 13 Example of my_Eliza's preferences database.....	46
Figure 4- 14 Example of the use of system's GSP database in a dialog .....	46
Figure 4- 15 Emotive lexicons dictionary .....	48
Figure 4- 16 Natural Language Processing Layers .....	50
Figure 4- 17 Algorithm for the normalization of a string .....	50
Figure 4- 18 My_Eliza's pattern matching algorithm.....	51
Figure 4- 19 Repetition recognition algorithm.....	51
Figure 4- 20 Example of a User Personal Database unit and how to use it in a conversation .....	52
Figure 4- 21 Example of system's GSP database and how to use it in a conversation .....	53
Figure 4- 22 Facial display construction layers.....	54
Figure 4- 23 Emotive lexicon dictionary parser.....	54
Figure 4- 24 Algorithm to set system's reaction affective state thermometer .....	57
Figure 5- 1 Example of Eliza's memory unit and fragment between Eliza and the user.....	60
Figure 5- 2 Comparing Eliza's memory structure (a) and A.L.I.C.E's memory structure (b) .....	61
Figure 5- 3 Example of A.L.I.C.E's memory unit and fragment between A.L.I.C.E and the user... ..	62
Figure 5- 4 The algorithm of A.L.I.C.E pattern matching operation.....	63
Figure 5- 5 The process involved in a typical transaction with A.L.I.C.E.....	63
Figure 5- 6 The process involved in a typical transaction with my_Eliza .....	63
Figure 5- 7 My_Eliza blackboard system (the shadowed layers have been implemented in A.L.I.C.E).....	64
Figure 5- 8 Example of a transformation from Eliza's memory structure unit to my_Eliza's.....	65
Figure 5- 9 Step by step transformation from Eliza's memory structure to my_Eliza's.....	66



Figure 5- 10 Comparing the memory structure of Eliza (a), A.L.I.C.E (b) and my_Eliza (c).....	66
Figure 5- 11 Example of a my_Eliza's memory unit with different topic.....	67
Figure 5- 12 Example of a negative and positive situation type in my_Eliza's memory structure....	68
Figure 5- 13 Algorithm of my_Eliza's pattern matching operation .....	69
Figure 5- 14 My_Eliza's blackboard system (the shadowed layers are required for the dialog box). 70	
Figure 5- 15 My_Eliza's dialog box implementation scope (white areas).....	71
Figure 5- 16 New A.L.I.C.E = my_Eliza's dialog box .....	72
Figure 5- 17 Context flow diagram .....	73
Figure 5- 18 Use case diagram .....	73
Figure 5- 19 Object Diagram of my_Eliza's dialog box.....	75
Figure 5- 20 Event trace diagram of my_Eliza dialog box: loading and conversing .....	76
Figure 5- 21 Blackboard transition diagram .....	76
Figure 5- 22 Emotive Labeled Extractor transition diagram.....	77
Figure 5- 23 List of pattern rules construction using dialog analysis .....	77
Figure 5- 24 Constructing a category from a dialog.....	78
Figure 5- 25 Index page of my_Eliza's prototype-1.....	81
Figure 5- 26 My_Eliza's dialog box pages: (a) register a new user and (b) login.....	81
Figure 5- 27 My_Eliza's dialog box main page chat.html .....	82
Figure 6- 1 My_Eliza's blackboard system (the shadowed layers are required for prototype-1) .....	86
Figure 6- 2 My_Eliza's prototype-1 implementation scope (white areas) .....	87
Figure 6- 3 My_Eliza's prototype-1 flowchart to construct the first reaction facial display.....	87
Figure 6- 4 My_Eliza's prototype-1 flowchart to construct a corresponding facial display to a reply .....	88
Figure 6- 5 My_Eliza prototype-1's context flow diagram .....	89
Figure 6- 6 My_Eliza prototype-1's use case diagram .....	90
Figure 6- 7 Object Diagram of my_Eliza prototype-1 .....	92
Figure 6- 8 Event flow diagram of my_Eliza prototype-1: loading and conversing.....	95
Figure 6- 9 Concern of the other analyzer, cognitive reasoning and affective attributing analyzer transition diagram .....	95
Figure 6- 10 Emotive lexicon dictionary parser and user's affective state analyzer transition diagram .....	96
Figure 6- 11 Thermometer and intensity analyzer transition diagram.....	96
Figure 6- 12 Facial display selector transition diagram.....	96
Figure 6- 13 Nonverbal property transition diagram.....	96
Figure 6- 14 Emotion transformation transition diagram.....	97
Figure 6- 15 Starting my_Eliza server script .....	100
Figure 6- 16 Example category for loading the list of pattern rules files.....	101
Figure 6- 17 My_Eliza prototype-1's login page.....	101
Figure 6- 18 The main page of my_Eliza prototype-1: chat.html.....	102
Figure 6- 19 My_Eliza's View.html.....	103
Figure 7- 1 Templates of default category with <pattern>*<pattern>.....	104
Figure 7- 2 The first connect .....	106
Figure 7- 3 My_Eliza stores user name .....	107
Figure 7- 4 My_Eliza jokes with the user.....	108
Figure 7- 5 The user is sad.....	109
Figure 7- 6 My_Eliza makes a mistake.....	111
Figure 7- 7 The user is confused .....	112
Figure 7- 8 The user hates my_Eliza (part 1) .....	113
Figure 7- 9 The user hates my_Eliza (part 2) .....	114
Figure 7- 10 The user gets angry .....	116
Figure 7- 11 My_Eliza versus the user .....	117
Figure 7- 12 My_Eliza feels sorry for the user.....	119
Figure 7- 13 My_Eliza in a topic about computer.....	120



---

Figure 7- 14 My_Eliza feels happy helping the user .....	122
Figure A- 1 The process involved in a typical transaction with A.L.I.C.E.....	131
Figure A- 2 Example of an AIML category in a topic about dogs .....	133
Figure A- 3 Example of the use of <that> tag in AIML categories .....	134
Figure A- 4 (a) Response construction process and (b) Alicebot's Graphmaster.....	138
Figure A- 5 Searching algorithm of Alicebot's pattern matching operation .....	140
Figure B- 1 Alicebot context diagram .....	141
Figure B- 2 Alicebot's use case diagram.....	142
Figure B- 3 Alicebot's class diagram.....	143
Figure B- 4 Alicebot's event trace diagram.....	144
Figure B- 5 Server script when the server runs.....	145
Figure B- 6 Alicebot's main page .....	148

**Error! No table of figures entries found.**





## Tables

Table 1- 1 Example of conversation between Eliza and user .....	3
Table 2- 1 List of skills a chatterbot should have .....	11
Table 2- 2 OCC's twenty-four emotion types .....	16
Table 2- 3 Properties of an ideal nonverbal QA system .....	20
Table 2- 4 Properties of recently proposed approaches to a nonverbal QA system .....	21
Table 2- 5 Distances between six emotional expressions .....	23
Table 2- 6 My_Eliza's properties .....	26
Table 3- 1 Four possible goals to pursue in Artificial Intelligence .....	27
Table 4- 1 My_Eliza's twenty-four emotion type to classify system's reaction affective state .....	36
Table 4- 2 Corresponding twenty-four OCC's theory emotion types into six clusters emotion types .....	47
Table 4- 3 Distances dataset between six universal emotions .....	55
Table 4- 4 Corresponding between six universal emotion types and OCC's theory emotion types .....	58
Table 5- 1 Example of fragment in positive and negative situation type .....	69
Table 5- 2 Example of categories construction based on a dialog analysis .....	78
Table 5- 3 Tests and results of my_Eliza's dialog box .....	82
Table 5- 4 Example of a conversation between user and my_Eliza in a topic about computer .....	84
Table 6- 1 Example of stimulus response preference rules .....	97
Table 6- 2 Example of cognitive reasoning preference rules .....	98
Table 7- 1 My_Eliza prototype-1's facial display dictionary .....	105
Table 7- 2 New categories in a topic about computer .....	118
Table C- 1 Concern of user happiness preference rules .....	150
Table C- 2 Concern of user sadness preference rules .....	151
Table C- 3 Concern of user fear preference rules .....	151
Table C- 4 Concern of user surprise preference rules .....	151
Table C- 5 Concern of user disgust preference rules .....	151
Table C- 6 Concern of user anger preference rules .....	152
Table C- 7 Concern of user neutrality preference rules .....	152
Table C- 8 Cognitive reasoning preference rules for user happiness .....	152
Table C- 9 Cognitive reasoning preference rules for user sadness .....	154
Table C- 10 Cognitive reasoning preference rules for user fear .....	155
Table C- 11 Cognitive reasoning preference rules for user surprise .....	156
Table C- 12 Cognitive reasoning preference rules for user disgust .....	157
Table C- 13 Cognitive reasoning preference rules for user anger .....	158
Table C- 14 Cognitive reasoning preference rules for user neutrality .....	159
Table D- 1 Process flowchart basic symbols .....	161
Table D- 2 Use case diagram notations .....	163
Table D- 3 Class diagram notations .....	164
Table D- 4 Event transition diagram notations .....	165
Table D- 5 State transition diagram notations .....	165



## Acknowledgements

This report describes the master's thesis that I have done in the past nine months. A master's thesis is the final stage in the Master of Science program to get a master degree. The project has been done in the group Knowledge Based System, headed by prof. dr. H. Koppelaar. A lot of people have helped me in the process.

Specifically I would like to thank Drs. Dr. L.J.M. Rothkrantz for his guidance during the work. Although he is a very busy man, he can always be the one whom I count on to be available. Much of this work would not have been possible without his dedicated support. I would like to thank him for understanding my personality and abilities, his willingness to listen, his patient to read my Indonesian-English translated thesis report and, most of all, for his abiding encouragement. I am looking forward to continuing our cooperation next future. Also for Ir. Frank van der Vlugt for his support as MSc Coordinator.

My infinite thanks go to Archi Delphinanto for his big love, support, and helping me a lot with the work. My admiration is sent for his effort creating the cover logo of this thesis report. My special thanks also I dedicate to Ibu, Ayah, Fabi and all my family members who always pray for me and support me. Furthermore, I would like to thank my colleague Sylvia "Cepi" G. Yuvenna for being a great companion, A. Wojdel MSc. for sharing knowledge about *automated character animation*, Dr. Clark Elliott for sharing his expert knowledge about *emotive lexicon dictionary*, Ir. Edwin de Jongh for sharing his Facial Expression Dictionary, and Ir. Caspar Treijtel for sharing his expert knowledge about Jess. Finally I would like to thank the following people who also were very supportive. These are Wiyono Hoo, Daniel Oranova Siahaan, Dian Indrawaty and M.T.A.P Kresnowati for listening to my stories, Lien Ngan and Bix Xi for being a great friend, Ir. Boi Sletterink for helping me with all kinds of technical problems and offering solutions, and the last but not least all my friends (whom I cannot write on one by one) for providing me with all the support I need.

*To ayah & ibu*



## Chapter 1 Introduction

NLP (Natural Language Processing) is a field in Artificial Intelligence involving anything that processes natural language, such as automated speech recognition, language translation, and etc. In particular, NLP research also deals with *question-answering systems* - QA systems. Such system has capability to interpret natural language in order to conduct semi-human conversations with users. We can find QA systems spreading in the Internet, but they were not invented on the Internet. They are generally believed to have been created in the form of *Eliza*, one of the oldest programs that can engage a human in conversation. Actually Eliza is not the first QA system, but this program has given so many effects (namely called Eliza-effect [COW95]) because it was so convincing that there are many people emotionally caught up in dealing with Eliza. All this was due to Artificial Intelligence (AI), which is an advanced form of computer science that aims to develop software capable of processing information automatically, without the need for human support. Eliza has also shocked AI community because it gave the impression of deep semantic linguistic processing but it was in fact based on shallow language processing.

This chapter is organized as follows. First, linguistic processing in Eliza is described in section 1.1. Eliza was only a dialog box that communicates with human users by exchanging text prompts. However, when people get involved in communication, they use a lot of nonverbal signals to help to regulate the conversation and complement their verbal utterances. In many situations, nonverbal communication is not used alone but jointly to verbal communication as a mean to strengthen what they say or to explicit it. The issue about face-to-face communication system as an ideal model of a multimodal communication system is generally discussed in section 1.2. Based on Eliza in section 1.3, we present the aim of the thesis, that is: to develop a multimodal communication system namely my\_Eliza. The problem definition of the thesis work is presented in section 1.4. Finally, the thesis report overview is introduced in section 1.5.

### 1.1 Who is Eliza?

Eliza is a program which simulated a Rogerian psychoanalyst by rephrasing many of the patient's statements as questions and posing them to the patient. The original Eliza was described by Joseph Weizenbaum in *Communication of the ACM* in January 1966 [WEI66]. Eliza was one of the first programs that attempted to communicate in Natural Language, which is written in MAD-SLIP for IBM 7094 operated within the MAC time-sharing system at MIT, USA.

Eliza worked by simple pattern recognition and substitution of keywords. The basic algorithm of Eliza is the following (see figure 1-1 (a), below):

1. Identify the "most important" keyword occurring in the input message and some minimal context within which the chosen keyword appears.
2. Choose an appropriate transformation rule and its mechanism. There are two transformation rules that are associated with certain keywords (see figure 1-1 (b), below). *Decomposition rule* serves to decompose a data string according to certain criteria (pattern). *Reassemble rule* serves to reassemble a decomposed string according to certain assembly specifications (reply sentence). If Eliza finds a keyword, she will pattern-match the string input against each decomposition rule for that keyword. If it matches, she randomly selects one of the reassemble rules (for that decomposition rule).
3. Use a selected reassemble rule to construct the reply.

The pre and post substitution lists, the keyword lists, and the list of decomposition rules and reassembly rules are constructed in a script. This script file controls all the behavior of Eliza.

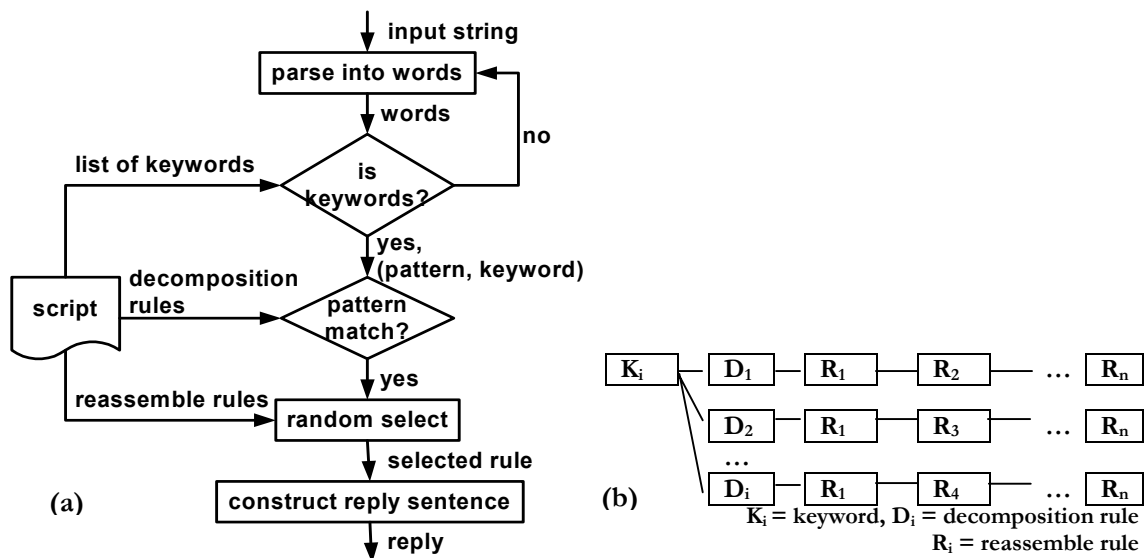


Figure 1- 1 (a) Original Eliza's basic algorithm and (b) keywords, decomposition rules, reassemble rules association in the script

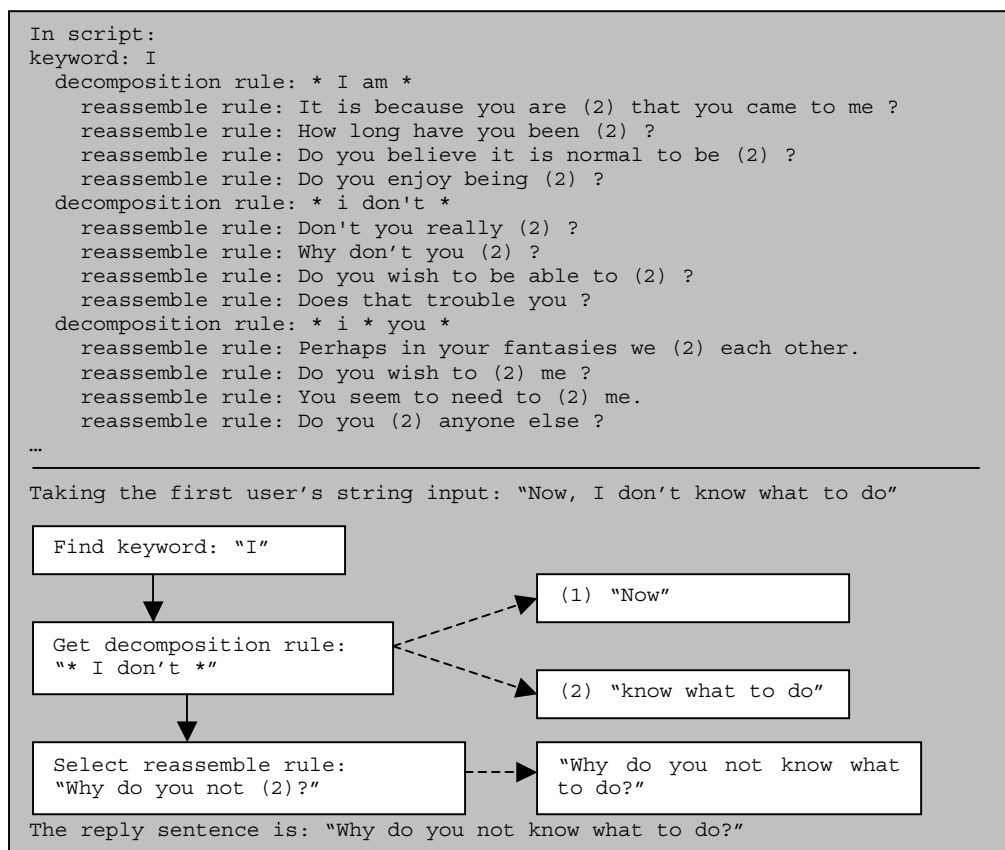


Figure 1- 2 Example of reply construction in Eliza

Figure 1-2 above shows an example when Eliza reads the string input "Now, I don't know what to do" (taken from the example fragment on table 1-1 below, dialog state number 7 to 8) and finds the first keyword "I". The system searches through its entire list of keywords, "I" is the only one that matches. The keywords are ranked; therefore if more than one keywords match, the system picks



the most salient. Next it applies all the decomposition rules for the keyword “I”, (“\* I am \*”, “\* I don’t \*”, and “\* I \* you \*”) to see if any matches. For the string input above, one rule does match: “\* I don’t \*”. Using this rule, it isolates parts of the string input around “I don’t”: the two phrases (1) “now” and (2) “know what to do”. Next she randomly selects a reassemble rule: “Why do you not (2)?”. The number is a sign to tell the system which decomposition phrase will be substituted into the sentence. The system uses this rule to construct the reply. It replaces (2) with the text that matches with the second asterisk in the decomposition rule – in the example above: “know what to do”. Finally Eliza replies with, “Why don’t you know what to do?”. The Eliza algorithm has pre- and post-processing steps as well, these handle the transformation of words like “I” becomes “you” or “my” becomes “your” on the reply sentence to the utterance. This can be done using a synonym matching facility.

Table 1- 1 Example of conversation between Eliza and user

Dialog#	Conversation	
1.	Eliza:	Hello. I am ELIZA. How can I help you?
	User:	Hello.
2.	Eliza:	How do you do. Please state your problem.
	User:	My name is Siska, we have met each other before.
3.	Eliza:	I have told you before, I do not care about names.
	User:	That's too bad.
4.	Eliza:	Please go on.
	User:	I have a nightmare and I rarely sleep at night.
5.	Eliza:	Can you elaborate on that?
	User:	I always woke up in the night because I got a nightmare then I couldn't sleep any longer.
6.	Eliza:	When?
	User:	Almost for the last two months, I guess.
7.	Eliza:	Tell me more about it.
	User:	I don't know what to do.
8.	Eliza:	Why don't you know what to do?
	User:	I think there is something wrong with my head.
9.	Eliza:	Why do you say about your head?
	User:	I always have nightmares if I have a lot of problems in mind.
10.	Eliza:	Do you wish that you have a lot of problems in mind?
	User:	Of course not, but what should I do. I am only human.
11.	Eliza:	Does that question interest you?
	User:	That is why I come to you because if I do not have enough sleep I just add my problem.
12.	Eliza:	Do you wish that you did not have enough sleep you just add your problem?
	User:	What????? I think you did not get my point.
13.	Eliza:	Why do you say about your point?
	User:	Do you understand what I have been talking to you about?
14.	Eliza:	Oh, I understand what you have been talking to I about.
	User:	Yeah, it began when my mother died.
15.	Eliza:	Why do you say your mother died?
	User:	Because that's what happens.
16.	Eliza:	Is there any other reason?
	User:	My mother and I are really close to each other.
17.	Eliza:	Who else in your family and you are really close to each other?
	User:	Nobody can understand me as much as my mother, even my boyfriend.
18.	Eliza:	Surely not everyone?
	User:	Absolutely, she really knows me.
19.	Eliza:	Can you elaborate on that?
	User:	...
...	...	...

Eliza is the classic template of many current Internet chat programs. In [HOL94] e-journal, Norman N. Holland deliberated that one of the reasons the original Eliza was so successful was Weizenbaum has created clever rules with limited domain. Referring to table 1-1 as an example of conversation between Eliza and one of her users, one of these limitations is Eliza does not store any knowledge about the users or any storage for users’ personal data. Therefore it does not care



about user's name (see dialog state number 3). The role of a Rogerian kind of psychotherapist doesn't require much intelligence, and a relatively simple algorithm can pull it off. That way, demands on its seeming ability to answer to the user would be limited. The responses can simply be canned sentences such as "That's quite interesting" or "Tell me more" or only substituting a word for the keyword. For example in dialog state number 16 to 17, if the string input contains the word "mother" Eliza will ask about "family". At most sophisticated level, Eliza can reverse or incorporate a type-in sentence so as to make a reply apparently based on an understanding of what has been said by the user. If the string input had no keywords at all (see dialog state number 4, 5, and 19), she would respond: "Please go on" or "Can you elaborate on that?". Eliza's interaction with human users was often comical and sometimes unearthly naturalistic, for example in dialog state number 11 to 13. This begins when Eliza finds the keyword "if" with decomposition rule "\* if \*" in user dialog state number 11 and the system selects randomly reassemble rule: "Do you wish that if you did not (2)". The Eliza system does not accommodate the facility to guarantee a correlation between the reply sentence and the entire conversation content. The user will find that Eliza's reply is inappropriate and at this point it may raise misunderstanding.

In spite of those limitations, in the book *Computer Power and Human Reason* [WEI76], Weizenbaum found even though the Eliza algorithm a relatively simple program, she can fool the naive users into believing that Eliza understood what they were saying. Some evidences showed some people actually felt more comfortable talking to Eliza than to a real psychiatrist.

Next generation of Eliza have grown to a more sophisticated program and is more specialized over the past thirty years [LAV96], for example Julia created by Mauldin [MAU94], A.L.I.C.E developed by Richard Wallace [WAL95] and so on. However, their primary reason for existence, which is to perform tasks at the command of humans, reminds the same.

## **1.2 Multimodal Communication System**

One of the other limitations of Eliza system is that users can only communicate with Eliza by exchanging text prompts. However beyond speech, human people can express their feelings or thoughts through the use of their body, facial expressions, and tone of voice. These bodily activities give real substance to face-to-face interaction in real life conversation. As indicated by Mehrabian (quoted by Donnel A. King in [KIN97]), it is proved that about 7 percent of the emotional meaning of a message is communicated through explicit verbal channels, about 38 percent is communicated by paralanguage, which is basically the use of the voice, and about 55 percent comes through the nonverbal channel, which includes such things as gesture, posture, and facial signals. Nonverbal communication is behavior other than spoken or written communication that creates or represents meaning. By adding this channel, face-to-face communication can approach the social interaction in real world fully and give its participants the possibility to achieve a high qualitative and realistic communication. Beside that, facial signals provide information about the observed person's affective state, attractiveness, cognitive activity and personality. In recent advances of QA systems, facial expression recognition and adapting life-like agents open up the possibility of automatic emotion recognition from user interaction in conversation between human and computer. Automated emotion recognition could facilitate machine perception of human behavior in order to reflect users' affect by replying their interaction with appropriate behavior, in verbal and/or nonverbal channel. This facility also could give the possibility to bring facial signals into human-machine interaction using QA systems as new modality making the conversation more natural and more efficient and make classification and quantification of users' affective state widely accessible to research in cognitive-behavioral science and psychology. This thesis addresses various problems concerning the modeling, recognition and classification of the encountered emotion state from written conversation between user and QA systems.





Takeuchi and Nagao [TAK93], and Schiano et.al. [SCH00] pointed out that human face-to-face conversation has provided an ideal model for designing a multimodal human-computer interaction (HCI). Characteristics of face-to-face conversation are the multiplicity and multi modality of the communication channel. A channel is a communication medium, while a modality is the sense used to perceive signals from the outside world. Examples of human communication channels are: the auditory channel that carries speech and vocal intonation, the visual channel that carries facial expressions and body movements. The sense of sight, hearing and touch are all examples of modalities. Multimodal user interfaces are interfaces with multiple channels that act on multiple modalities. Conversation is supported by multiple coordinated activities of various cognitive levels. As a result communication becomes highly flexible and robust, so that failure of one channel is recovered by another channel and a message in one channel can be explained by another channel. This is the basic idea how a multimodal HCI should be developed to facilitate realistic human-machine interaction. However, it cannot be neglected the possibility that a multimedia communication system sends ambiguous asynchronous signal via different channels. For example if the machine sends output string “It’s really the happiest day in my life” through verbal channel but it displays ☹ (sadness facial expression) through nonverbal channel, which opposites to the output string. This problem rises since the machine did not have the facility to recognize emotion in the text prompt and/or cannot coordinate it with appropriate facial expression.

Current majority research works in separate studies is based on various human communicative signals [NAK99]. Although the non-verbal aspects of communications, such as emotion-based communications, play very important roles in our daily lives, most of the researches so far have concentrated on the verbal aspects of communications and have neglected the nonverbal aspects. Therefore since 1992, the Automated System for Nonverbal Communication is an ongoing project at Knowledge Based System Group of Delft University of Technology [VAN95]. The goal of this project is the development of an intelligent automated system for the analysis of human non-verbal communicative signals. The system has to provide qualitative and quantitative information on different levels about various nonverbal signals sensed while monitoring a human subject. The system should detect a nonverbal signal given by the observed person. Next, the system should categorize the detected signals as a specific facial action, a specific body action, a specific vocal reaction, or specific physiological reaction. After that, the system should give an appropriate interpretation of the recognized communicative signals (e.g. emotional interpretation). At last, the system should reason about the intentions of the user and (optionally) respond in a similar, anthropomorphic manner.

### **1.3 The Aim of this Thesis: My\_Eliza**

The aimed intelligent system to analyze human behavior should be able to process, interpret nonverbal signals and respond using animation of human nonverbal signals. The multimodal analyzing system consists of multimodal input and multisensor data. The system should analyze those inputs automatically and use the results to form a hypothesis about the intention of the current observed object. The hypothesis is used to perform a proper reaction of the system. As a first step in achieving automatic analysis of human behavior and face-to-face communication, automated emotion recognition in human conversation between the users and a QA system has been investigated. This thesis discusses the results of the research, which ensued in the development of the **my\_Eliza** – an advance version of the original Weizenbaum’s Eliza. My\_Eliza was aimed at the design and establishment of a QA system of a *semi* automated emotion recognition from human user written conversation. A user or client can communicate with the system using **typed natural language**. The system will reply by text-prompts and appropriate facial-expressions. In order to communicate using nonverbal facial signals, which are appropriate to the conversation, the developed system should be able to process natural language and emotional reasoning.





The problem of automating emotion recognition and generating appropriate nonverbal facial displays on a QA system as defined in this thesis research comprises into three sub-discussions, such as:

1. Automatic generation of system's reply text prompts with ability of anaphoric analysis and ability to respond the conversation based on its topic.
2. Semi automatic emotion recognition of user's affective state and its intensity.
3. Automatic facial display selection from a facial expressions database based on emotion analysis.

## **1.4 Thesis Problem Definition**

The phases of this thesis is as follows:

1. **Study literature.** In this phase, we collect information, data, journals, papers and experimental reports about QA systems, emotion recognition and facial display generation.
2. **Seek existing system.** In this phase, we compare and study some of the existing QA systems that are developed after Eliza's era in the Internet. We find A.L.I.C.E, a QA system developed using AIML (Artificial Intelligence Markup Language - an extended Extensible Markup Language - XML script specification for programming a QA system) as the most up to date QA system in the Internet. We will use A.L.I.C.E program as model of dialog box layer of our system. Using AIML, we can solve some limitations of the pattern matching operation (see section 2.2). As the matter of the fact A.L.I.C.E has richer pattern rules than Eliza, however its pattern matching approach is similar to Eliza.
3. **Identify possible emotion types and emotion-eliciting factors.** In this phase, we collect some methods to classify human emotions and factors that elicit those emotions.
4. **Define and design new model.** In this phase, we design the global architecture of my\_Eliza system by adapting the models that we have chosen in the previous phases. In order to find the way that the system can reply user's prompt, recognize emotion during conversation and generate appropriate facial displays, we refer to human abilities. This phase is divided into three stages:
  - **Build specification with AIML.** Since we develop my\_Eliza based on A.L.I.C.E program as our model of dialog box, we also use AIML to program our QA system to construct a reply sentence. In this phase, we also define the personality and role of my\_Eliza as a new version of Weizenbaum's Eliza.
  - **Combine two modalities.** We design my\_Eliza to accept user's prompt and reply it with a text prompt from the verbal channel and appropriate facial displays from the nonverbal channel. For this goal, two next stages are obligatory.
  - **Add emotion reasoning.** We adapt some methods we have found in phase 1 and phase 3 in my\_Eliza (see section 2.3).
  - **Add facial display.** We adapt the classification methods that we have found in phase 3 in my\_Eliza (see section 2.3). The facial display is selected based on emotion analyses that have been done in the previous stage.
5. **Implementation.** In this phase, we use an incremental development approach (see section 3.4) to implement my\_Eliza. Currently, we have implemented my\_Eliza prototype-1 (see chapter 6 for detail explanation). This system has its own reply generation (a dialog box) and is able to recognize emotion from a single user's prompt. This prototype recognizes emotion and generates facial displays based on stimulus response.
6. **Analysis and testing the first prototype.** My\_Eliza prototype-1 has not fully tested for multiuser yet. This report presents a result of conversation between my\_Eliza and a human user (see chapter 7). However, the development of this prototype-1 has already dedicated to serve multiuser and multi browser in the Internet. The fully testing of the prototype for this aim has been left out as the recommendation for future work of this system.



## **1.5 Thesis Overview**

This study involves two research fields: nonverbal QA systems (natural language processing and psychological processing based on emotion reasoning as well as computer vision of facial expression analysis) and AI. This thesis report starts with two introductory chapters and the other chapters explain the general design and implementation of my\_Eliza prototype. The structure of this thesis is as follows:

- **Chapter 2** introduces QA systems development in historical perspective and emotion recognition and facial display generation, – it provides the basic domain and surveys the current research so far to solve these problems in an automatic way.
- **Chapter 3** gives an introduction to the field of AI, NLP, and knowledge-based systems that provides an assessment of the problem of simulating human conversation, automated emotion recognition analysis and generating facial displays according to AI paradigm and presents an overview of the AI techniques deployed in my\_Eliza prototype.
- **Chapter 4** discusses the global design of the system as an ultimate design.
- **Chapter 5** discusses the first implementation layer, my\_Eliza's dialog box.
- **Chapter 6** discusses the design and implementation of the first prototype of the system.
- **Chapter 7** gives an overview of the overall testing of my\_Eliza prototype-1 to communicate with a human user in typed natural language.
- **Chapter 8** concludes the work on automating emotion recognition and facial display generation, and points out some recommendations for future works.



## Chapter 2 Human-Computer Interaction

Human computer interaction - HCI is a discipline whose goal is to bring the power of computers and communications systems to people in ways and forms that are both accessible and useful in our working, learning, communicating, and recreational lives. As described in section 1.2, the way human communicate to each other provides an ideal model for designing a multimodal HCI-system. Therefore the development of a QA system to be a multimodal communication system should also take this approach as reference point. After all, human face-to-face communication has the characteristics of multiplicity and multi modality of the communication channel.

There are several related issues to be discussed in this chapter: the development of QA systems after *Eliza's* era is presented first (in section 2.1). The degree of intelligence of a QA system, natural language generation of a QA system and conversation flow between the user and the QA system are discussed in section 2.2. The next issues are: the classification of emotion type, the recognition process of emotion from the text prompt and facial display classification. These issues as well as the capability of the QA system to generate facial display by using the result of the recognition process are discussed in section 2.3. This chapter further surveys the past work on solving this problem. How human interprets and reacts to emotion is meant to serve as an ultimate goal and a guide for determining a set of recommendations for developing automatic emotion recognition and facial display generation (section 2.4). Section 2.5 provides a short overview of early works on emotion recognition and facial display generation. Finally, section 2.6 discusses the existing solution and lists the main contribution of the work presented in this thesis to the research field of automatic emotion recognition from human-computer conversation and facial display generation.

### 2.1 Chatterbot

Recent years have witnessed a proliferation in computer-based models of natural language structure and human language behavior; QA system is one of them. Nowadays, a QA system is also called a *chatterbot*. Using Simon Laven's term [LAV96], chatterbot is a short for "chatter" and "bot". Bot is short for "robot". The Merriam-Webster's Collegiate Dictionary defines the term robot as "a device that automatically performs complicated, often repetitive tasks." The bots discussed in this case are intelligent systems, machine-based that attempts to simulate human intelligence in a particular domain. The goal of these machines and the computer programs that drive them is to produce "behavior" indistinguishable from that of a human actor in the domain. Using bot definition, a chatterbot has the aim to at least temporarily fool a human into thinking they were having a dialog with another person. Weizenbaum's *Eliza* was an example of this class of these programs. Though for robots and bots cognitive models play an important role how they react to user request, original *Eliza* never used them because this system never really necessary understood everything to the detail what humans have conversed with her.

According to Nautilus' chatterbot FAQ [COW95], chatterbots play a role as human personally companion who can be entrusted all the worries and trouble. They are exciting and humorous interlocutors too; know very well how to get the users comfortable to talk with them. Each QA-system represents a certain character with certain gender and personality. They can be a man or a woman. They could represent a young paranoid such as Parry created by Colby [COL73], or an insane character such as Racter by Etter and Chamberlain [ETT84]. They could be very smart, do always know the answer and in order to support a user's work they could have ability too, such as: organize the date and addresses, check the Internet, or moreover do shopping cart. They also could be game partners as well and ready for having fun, such as bots within *Starship Titanic*



created D. Adams [ADA80]. While in the case of *Eliza*, Weizenbaum gave the program the role as a doctor that has the aim to keep the conversation going.

Humans not only communicate by oral or written language but they also communicate nonverbally. A politician uses gestures to make his arguments more convincing or a girl smiles in order to tell her friend that she is willing to keep on talking although the topic might generate conflicts. The nonverbal part of human communication plays an important role in defining, maintaining or avoiding a relationship. Since many people become emotionally involved with the QA system, automating the recognition of users' emotion would therefore be highly beneficial in order to give a proper user reply, both in the verbal channel (in this case typed text prompt) and in the nonverbal channel (in this case a facial display). Velasquez [VEL97] and Schiano et.al. [SCH00] pointed out that emotions are an essential part of human lives; they influence how human think and behave and how human communicate with others, and facial displays are human primary means of communicating emotion. Face to face communication is inherently natural and social for human-human interaction and substantial evidences suggest this may also be true for human-computer interaction. This implies to Elliot and Melchior [ELL94], and Nakatsu et.al. [NAK99] who pointed out that since nowadays as computer acts as electronic secretaries or communication mediators, they become common entities in human society. The capability of communicating with humans using both verbal and nonverbal communication channels would be essential. This will surely make interaction between computers and humans more intimate and human-like (Lee [LEE99], Predinger and Ishizuka [PRE01], and Cassel et.al. [CAS94]). Using human-like faces as means to communicate have been found to provide natural and compelling computer interfaces.

However, there are only a few researches involving research on human emotion recognition, because it is difficult to collect a large amount of utterances that contain emotion [NAK99]. Only a few of them work in recognizing emotion from text. Moreover, the interpretation of emotion eliciting factors is strongly situation and culture dependent [WIE99]. Therefore, the development of automated system that can accomplish this task is not trivial.

## **2.2 Aspect of Natural Language Generation in a QA System**

The main goal here is to explore the issues of the design and implementation of a QA system that could be the basic layer to implement an automatic user's emotion recognition system and a system for facial display generation – a nonverbal QA system. In general, three steps can be distinguished in tackling the problem. First, the QA system should have enough degrees of intelligence to simulate human conversation. The next step is regarding how a QA system talks with a human user. A QA System accepts natural language statements and questions as input, uses syntactic and semantic analysis that provides deductive or inductive procedures for such analysis as answering questions, and generates an output string as answer. The way QA systems represent and retrieve information is transparent by their **memory structure**. The memory structure functions as the systems' "brain and is the foundation of the ability level of the system to "speak" in human natural language. The final step is to maintain the continuity and conversation flow between the user and the system.

Before a QA system can be built, one should decide what functionality the system should have. A good reference point is the functionality of the human brain. After all, it is the best-known memory storage and information processor. This section discusses three basic problems related to the process how QA systems generate natural language as well as the capability of human brain with respect to these.

### **Degree of intelligence**

Developing a QA system that is not easily identified as being artificial and makes fun to talk with seems a good approach, especially since its role meant to be user's personal companion. It



is more important for the users to have the illusion that a QA system is human and not artificial with a high degree of believable. Believable systems are a combination of autonomous behavior systems and believable character representations in order to be more engaging user enjoys. In order to simulate a perfect believable QA system needs a certain degree of intelligence to perceive the conversation. The problem is that nobody has shown to be able to understand neither every aspect of human intelligence nor every aspect of a QA system.

An influential definition of “intelligence” was put forward by A.M. Turing [TUR50]. He formulated the idea of an universal Turing machine to define intelligence in a way that applied to anything that is intelligent. He wanted to formulate the issue of whether machines could be intelligent in terms of whether they could pass the following test: a judge in one room communicates by teletype with a machine in a second room, and a person in a third room for some specified period (figure 2-1).

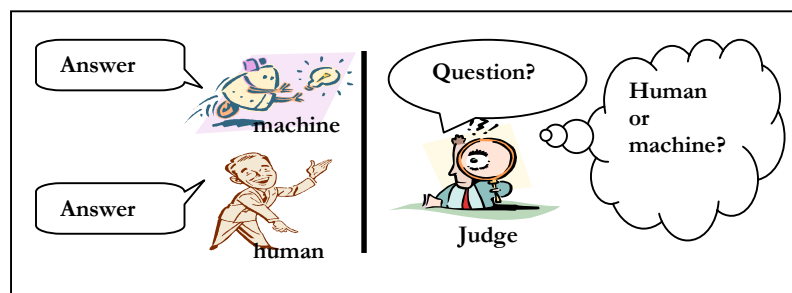


Figure 2- 1 Turing test

The machine is intelligent if and only if the judge cannot tell the difference between the machine and the person. Turing suggested replacing the concept of intelligence with the concept of passing the Turing test. However, there is a gap in Turing’s proposal: how the judge to be chosen. A judge who was a leading authority on genuinely intelligent machines might know how to distinguish a machine apart from people but a judge who is naïve about computers is easy to fool. The machine is not necessary to be intelligent or to operate exactly the same way as existing intelligence to pass the Turing test because of the possibility that some intelligent creature might not be able to participate correctly in the test for some physical reason. With this test, Turing proposed that it is far more important that a machine appears intelligent and cannot be identified as being artificial. The Loebner contest [LOE91] instantiates the Turing test for QA systems; conducted the first time in Boston’s Computer Museum on November 8, 1991.

How to build a believable QA system? There are marked individual differences when a human communicates with other. When a doctor talks to a patient is different from a teacher who talks to students. The way and content of a salesman’s speech is different with a politician. As indicated by Weizenbaum [WEI76] the specific aim and personality is related to the crucial role context plays in all conversations, because those influence how human selects linguistic actions to imply others and distinguish one with other people. The personality of a QA system defines the abilities and the skills of the system to discuss with the users. In the most cases, the system has the ability to realize the identity of the user and its own identity, to respond the time or the date and some other usable things (Nautilus’ chatterbot FAQ [COW95], table 2-1 below). According to Hutchens [HUT95] to build the “brain” of such a QA system, it is important to simulate what human-to-human discuss in daily life. We can use our own personally inspiration by involving ourselves at a chat room, listen to other chatting people or listen to other QA systems and watch their responses. As other sources, we can also use books, movies, magazine. Like human, the bigger the capacity of system’s brain, the more variation in conversation content, and the more realistic the conversation the users can feel.



Table 2- 1 List of skills a chatterbot should have

No.	List of skills
1.	Remember the actual time.
2.	Remember the actual date.
3.	Remember the current day of the week.
4.	Remember the actual month of the year.
5.	Remember the length of time the bot is talking with the user so far.
6.	Remember the name of the user who is actually chatting with the bot.
7.	By random choice the name of another user who is actually talking with it.
8.	By random choice selecting comments from a list of pre-defined responses.
9.	Remember the user's lines temporarily so it could be answered back in a personalized form and to talk about it some chat lines later.
10.	Respond the conversation based on its topic.
11.	Ability to makes reply concerning what the bot said previously based on general user statements like "I like it" or "That's cool".
12.	Ability to choose a lists of words accurately or by random within reply.
13.	Have a feature "silent mode": if the user do not want to talk with the bot but want to talk with other user on the system without responses from the bot.

In conversation, humans also show all kind of emotions on every verbal and nonverbal communication channel. Their brain is dedicated to those kinds of information processing. This reasoning is meant to be a universal motivational situation representing urges that impel human into different action. Adding facilities to recognize emotion and show appropriate action based on emotion reasoning to a QA system will increase users' belief (will be discussed in section 2.2). This means also to increase the system's degree of intelligence. Although the Turing test is usually performed with text communication, so that sensory expression such as voice intonation and facial expression does not play a role, this does not mean that emotions are not communicated (Picard [PIC97]). In fact, most users of text e-mail find that recipients infer emotion from the email, regardless of whether they intended to communicate emotion through the mail. A machine, even limited to text communication will communicate more effectively with humans if it can perceive and express emotions. Affective communication may involve giving computers the ability to recognize emotional expressions as a step toward interpreting the situation of what the user might be feeling. That situation directly affects to the system's controlling behavior.

## Memory Structure of a QA System

After the personality of the QA system is determined and the resources are collected, the next step is to choose an approach to represent them in the system's memory structure. The QA system retrieves the information from its memory and uses syntactic and semantic analysis to output a string as an answer to the user's string input. Robert F. Simmon [SIM70] has thoroughly reviewed most of this research; four major types of memory structure can be distinguished:

### 1. *Procedure oriented models*

This approach is derived from Chomsky's language structure<sup>1</sup>. Emerging from preoccupation of formal programming language and compilers, this model translates natural language input into predefined subroutines, which can be executed to accomplish interpretation of input and retrieval of stored information. This model requires numerous examples to be adequate for expressing complex data retrieval of the user input and for describing data for storage. One apparent weakness in this approach so far is the fact that the use of independent syntactic analysis implies that numerous syntactically valid but semantically impossible analyses would have to be considered.

---

<sup>1</sup> In 1965 for memory structures of content representation, Chomsky devised paradigm for linguistic analysis that includes syntactic, semantic and phonological components to account for the generation of natural language statement, which based on a deep structure of categorical component of lexicon and transformation rule [CHO65].





## **2. *Logic oriented system***

This type of memory structure is sometimes combined with a procedural approach, where input is translated into predicate calculus expressions. These expressions can be tested against axiomatic knowledge by automated theorem proving techniques, notably the resolution methods. This approach does not explicitly deal with semantic classes although word-classes that form the grammar rules may thus be semantic classes. They are only used to check semantic well-formedness against a model representing the true states of the relevant universe.

## **3. *Pattern matching operation system***

This approach uses a limited relational memory, in which concepts are usually represented in list structures of keyword or word categories and the simple relations of pattern operations rules that exist within these structures. An example of this system is Weizenbaum's *Eliza*. As has been described in section 1.1, the program structure of *Eliza* that support the conversation capability is a set of keywords and whose operations are the substitution of a partially composed input statement in conjunction with some portion of the input sentence. These transformations are provided to *Eliza* by a prepared script that consists of a high level program whose statements are not necessarily sequential. The flow of control among command statements in the script is guided by the keywords of the input. The semantic analysis of a word or a pattern of words for a computer is the selection of either a data structure or pattern of operations that it signifies. However, there are still limitation features for remembering previously topics and continuity of conversation flow. For Weizenbaum [WEI76], these limitations are not big problems since *Eliza*'s aim as conversation machine is to keep the conversation going, even at the price of concealing any misunderstandings on its own part.

## **4. *Network memory structures models***

This approach uses sophisticated network memory structures consisting of either words or concepts and the links between them. Interpretation and retrieval in this model often involve techniques for finding intersection within the network. An example of a model in this category is MUSE [MCC72]. The system encodes dictionary definitions of words as a network of word-nodes connected by syntactic and semantic relations. This approach uses the intersection techniques, which used to discern the context of words in the input and to eliminate syntactic ambiguity, sometimes is leading to wrong interpretation or failure to find valid comparison. Further research is needed to determine the appropriate behavior for this intersection function.

According to Chomsky [CHO65], basic language structure resides internally and basic patterns are innate to human beings. This theory is termed *universal grammar*. Essentially, humans, when born, have a set of rules already built into them. These rules allow human beings the ability to learn any language through interaction. A strong piece of evidence this theory is the fact that children, through a short period of time, have the ability to produce an infinite number of sentences, as well as perceive and comprehend an infinite number of sentences. However, to introduce natural language to a machine, the power of the pattern-operation rule and its simplicity have been appreciated widely and exploited in the applications to analyze semantic and the deductive operations that required answering questions. Most of the chatterbots spreading in the Internet use pattern-operation rule for synthesizing human natural language. They use large, more natural, and core bound memory structure consisting of a huge number of patterns, for examples those of A.L.I.C.E [WAL95], Julia [MAU94], Hex [HUT95], Fred [GAR97], PC Therapist [WIE86], etc. The pattern matching operation rule is today considered as the key to semantic analysis. It is the simplest of the four approaches reviewed above. Comparing to the linguistic notion of deep structure representation of Chomsky, which serves the purpose of showing the complexity of a natural language sentence, using a list structure of keywords or word categories is frankly simplistic. It is because of their linguistic richness, the Chomsky-type base structures are more suited for linguistic research than for representation of factual information.



## Conversation Flow in QA Systems

After the memory structure has been established, the next step is to make sure that the QA system reacts according to where the conversation flows, where the conversation's state is, what kind of situation we have, and what the conversation's topic is. Using dialogs analysis, it is possible for us to capture all situations and to find the appropriate reaction. As mentioned above, the pattern matching operation of the original Weizenbaum's Eliza still has three major problems [SIM70]:

1. Lack of anaphoric analysis. It cannot use previous question-answer to keep the continuity of the conversation content and to store information about the user.
2. Lack of ability to restrict the conversation on its topic.
3. Lack of ability to get the meaning beyond the sentence.

Tackling these problems, Wallace [WAL95] proposed the idea to expand the memory structure. Therefore, the system does not only have input pattern-operation rules (which in Eliza are called decomposition rules), it also has topic pattern-operation rules and history pattern-operation rules. He formulated his idea using extended-XML (Extensible Markup Language) script specification for programming the memory structure for a QA system, called AIML (Artificial Intelligence Markup Language). The most important AIML units are:

- **<aiml>** : the tag that begins and ends an AIML document.
- **<category>**: the tag that marks a "unit of knowledge" in the system's memory structure.
- **<pattern>** : the tag that contains a simple input pattern rule that matches what a user may type.
- **<topic>** : the tag that contains current conversation topic pattern rule.
- **<that>** : the tag that refers to system's previous reply as a history pattern rule. Using **<that index="nx, ny"/>** tag, we can refer to any system's reply on history.
- **<template>**: the tag that contains the response to a user input. There is considerable freedom of expression in the construction of response templates. In Eliza, this part is namely reassemble rule (see figure 2.2).

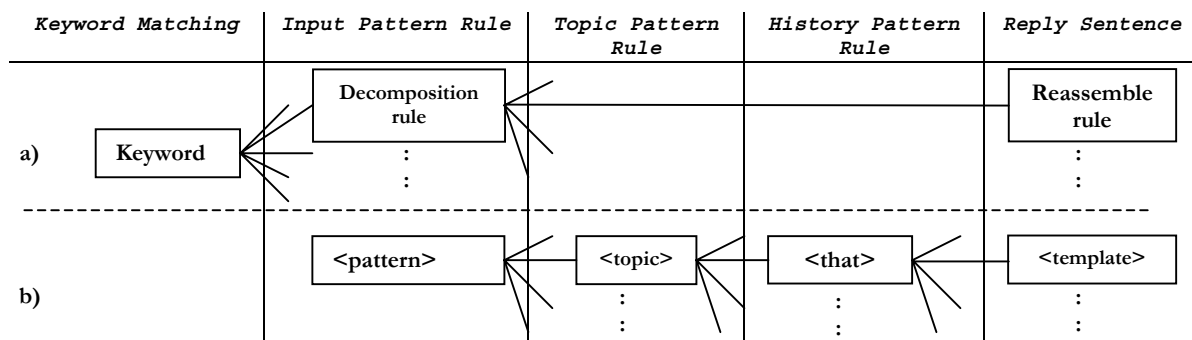


Figure 2- 2 Memory Structure of Eliza (a) and A.L.I.C.E (b)

Wallace built a QA System called A.L.I.C.E - Artificial Linguistic Internet Computer Entity. A.L.I.C.E asks and answers questions, acts as a secretary reminding people of appointments, spreads gossips and even tells lies. A.L.I.C.E won the 2000 and 2001 Loebner Prize. The basis for A.L.I.C.E's behavior is AIML. A.L.I.C.E's units of memory structures are the categories in AIML and the algorithm finds best-matching pattern for each input. The category ties the response template directly to the stimulus pattern. A.L.I.C.E is conceptually not much more complicated than Weizenbaum's Eliza. The main differences are (see figure 2-2) the more possibilities of reply sentences based on their topic and history. AIML also has the possibility to create new content by a dialog analysis. Using dialogs analysis, we can eliminate the blunder which occurs due to the fact





that the system does not store all the facts what user have been talking about in the previous conversation. We will apply this idea in our model, which will be discussed in chapter 4.

```
<topic name="FATHER">
  <category> <that>I AM SORRY</that>
    <pattern>FATHER WAS *</pattern>
    <template> <random>
      <li> Are you also </star>?</li>
      <li> Were you both close?</li>
      <li> What else do you remember about him?</li>
    </random> </template>
  </category>
  //more father categories....
</topic>
```

Figure 2- 3 Example of one unit memory structure of A.L.I.C.E

Humans can follow the conversation because they have the ability to remember the semantic and the connection to each sequential conversation line (Slugoski and Hilton [SLU00], Holland [HOL94]). Any form of learning or memory involves some form of internal representation of past events. It is very annoying if a person converse with no connection at all between his lines because there is conversational coherence result from a memory component that represents semantic relations among topical elements in the dialogue. Figure 2-3 shows A.L.I.C.E's unit memory structure when A.L.I.C.E was set up on talking about user's father and something bad has happened with the user (<that> tag contains "I AM SORRY"). In this example, A.L.I.C.E should continue to show empathy since the user expects the system to understand his/her feeling. It is difficult to achieve if A.L.I.C.E does not store and use the fact that has been mentioned in the previous dialog. <topic> tag allows A.L.I.C.E to prefer responses that deal with the topic currently being discussed. This creates topical conversation, yet still has the ability to move from one subject to another. <that> tag allows A.L.I.C.E to prefer responses that deal with her previous reply. Once the topic is set, when the client types in a statement for A.L.I.C.E to find a response for, the categories defined within the<topic> tags matching the current topic and the <that> matching the previous reply will be searched first before any of the non-topic categories, non-that category or other categories. When a user enters an input, the program scans the categories to find the best match. By comparing the input with the pattern in the following order, the algorithm ensures that the most specific pattern matches first. Basically it finds the longest pattern matching an input.

```
<topic name="FATHER">
  <category><that>YOUR FATHER</that>
    <pattern>PASSED AWAY</pattern>
    <template><think><set name="father">died</set><think/>
      I am sorry to hear that.
    </template>
  </category>
  <category><that>I AM SORRY</that>
    <pattern>HE WAS</pattern>
    <template>
      <condition name="father" value="alive">Is he now?</condition>
      <condition name="father" value="died">Were you close to each other?
    </condition>
    </template>
  </category>
</topic>
```

Example fragment:

Alice: Do you live with your father?  
User : My father passed away two years ago.  
Alice: I am sorry to hear that.  
User : He was a good friend rather than a good father.  
Alice: Were you close to each other?  
...

Figure 2- 4 Example of A.L.I.C.E's memory unit for storing information



Inside `<template>` tag we can use other tags (see AIML Specification [BUS01]) for set or get variable, refer to another memory unit, substitute word or phrase with other words or phrases, reply a pattern based on some conditionals or execute an operation. Therefore using AIML, the QA system can be developed into wide range functionality. Pragmatic issue of the conversation and storing information about the user while conversing are two examples of problems that can be solved in AIML programming. An example in figure 2-4 above is a new version of A.L.I.C.E unit memory structure in figure 2-3 that shows how the system can store information by simply using `<set>` tag. Using the `<think>` tag, which causes the system to perform whatever it contains but hide the result from the user, the system will set a variable name "father" in its memory to "died". This information can be used in other categories. In this example, a category uses this information as condition value when the system has to decide to choose the appropriate answer to avoid the pragmatic issue considering the tragic event of the user's father.

The schema above builds a cognitive model for the QA system to "perceive" the dialog content that connects the user's conversation lines. However still, there exists no natural language processing system exists that can go beyond sentence boundaries in semantic analysis. There is no intelligence for understanding metaphors or anecdotes only explicitly stated instantiations of emotion instances within them.

## 2.3 Aspects of Emotion Recognition and Facial Display Generation in a QA System

The main goal here is to explore the issue of design and implementation of a nonverbal QA system that could recognize the user's emotion and show a proper facial display accordingly. In general, three steps can be distinguished in tackling the problem. The first step is to define which and how many emotions can be recognized by the system. The next step is to define mechanisms for extracting emotion-eliciting factors in the observed text prompt, which devise the categorization mechanism and the emotion interpretation mechanism. Currently, the interpretation of the emotion-eliciting factor is still semi automatic since we assume to use the memory structure approach of Weizenbaum's [WEI67] or Wallace's [WAL95] pattern matching operation. The memory structure of this approach does not store the semantic meaning of the text. It needs human intervention to interpret the semantic meaning. Using the defined emotion reasoning, the final step is, to define some set of categories of emotions that we want to use for facial displays classification and facial displays generation mechanism.

Before a nonverbal QA system can be built, one should decide what functionality the system should have. A good reference point is the functionality of the human brain analyzer. After all, it is the best-known emotion recognizer and nonverbal generator. This section discusses the three basic problems related to the process of a QA system, which can recognize the user's emotion and generate system's facial display as well as the capability of the human brain with respect to these.

### Emotion Classification

Quoting from Meriam-Webster dictionary, emotion is a state of feeling. How many and what kinds of emotional expressions are to be treated in a QA system are interesting but difficult issues. The following are results of early research on emotion classification:

#### a. Basic Emotions

Reddy [RED01] quoted that most psychologists agree that emotions have a special relationship to goals. Emotions have a *valence* or hedonic tone that renders them either inherently pleasant or inherently unpleasant and they also have *intensity* that determines how easy or difficult it is for a person to override them. These characteristics distinguish emotions sharply from cognitions. Every emotion is either pleasant or unpleasant and every emotion has a varying



intensity regarded as either shaping one's goals or reflecting one's goals. These two types of emotion he considered as basic emotion.

**b. Universal Emotions**

Ekman and Friesen [EKM75] described basic emotions in terms of facial expressions, which uniquely characterizes these emotions with seven universal emotions: neutrality, happiness, sadness, anger, fear, disgust, and surprise. This research has mainly concentrated on primary or archetypal emotions, which are universally associated to distinct expressions. In this theory, the basic emotions are considered to be the building blocks of more complex feeling states.

**c. Emotions Based on Valenced Reaction to Situation**

Bazzan and Bordini [BAZ01] mentioned Ortony, Clore and Collins theory (OCC's theory) on their report to reason and grouping emotions in their framework for the simulation of agents. OCC's theory believes cooperative problem-solving systems must be able to reason about emotions. It is based on grouping human emotions by their eliciting conditions events, their consequences of their action, and their selections of computational implementation. OCC's model of emotions based on those three types of emotion has three branches:

1. *Attraction* relates to emotions that are arising from aspects of the object.
2. *Consequences of event* relates to reaction of others' fortunes.
3. *Attribution* relates to approval of self or other.

In addition, there is a *compound* class that involves the emotions of gratification, remorse, gratitude and anger. Those branches derive into twenty-four types of emotion on table 2-2 below.

**Table 2- 2 Twenty-four emotion types according to OCC's theory**

Group	Specification	Name and Emotion Type
Well-Being	Appraisal of a situation as an event	<b>Joy</b> : pleased about an event <b>Distress</b> : displeased about an event
Fortune-of- Others	Presumed value of a situation as an event affecting another	<b>Happy-for</b> : pleased about an event desirable for another <b>Gloating</b> : pleased about an event undesirable for another <b>Resentment</b> : displeased about an event desirable for another <b>Sorry-for</b> : displeased about an event undesirable for another
Prospect-based	Appraisal of a situation as a prospective event	<b>Hope</b> : pleased about a prospective desirable event <b>Fear</b> : displeased about a prospective undesirable event
Confirmation	Appraisal of a situation as confirming or disconfirming an expectation	<b>Satisfaction</b> : pleased about a confirmed desirable event <b>Relief</b> : pleased about a disconfirmed undesirable event <b>Fears-confirmed</b> : displeased about a confirmed undesirable event <b>Disappointment</b> : displeased about a disconfirmed desirable event
Attribution	Appraisal of a situation as an accountable act of some agent	<b>Pride</b> : approving of one's own act <b>Admiration</b> : approving of another's act <b>Shame</b> : disapproving of one's own act <b>Reproach</b> : disapproving of another's act
Attraction	Appraisal of a situation as containing an attractive or unattractive object	<b>Liking</b> : finding an object appealing <b>Disliking</b> : finding an object unappealing
Well-being/ Attribution	Compound emotions	<b>Gratitude</b> : admiration + joy <b>Anger</b> : reproach + distress <b>Gratification</b> : pride + joy <b>Remorse</b> : shame + distress
Attraction/ Attribution	Compound emotion extensions	<b>Love</b> : admiration + liking <b>Hate</b> : reproach + disliking



d. *Emotion-related words in Two Dimension Circular Order*

Russell [RUS80] argued emotion could be best characterized in terms of a multidimensional affect space rather than discrete emotion categories (such as Ekman's basic emotion). Figure 2-5 shows the result of Russell's work of a two-dimensional scaling solution for twenty-eight English emotion-related words. The left-right dimension represents *Pleasant*, with pleasant feelings at the right. The top-bottom dimension represents *Arousal*, with feelings of activation on the top.

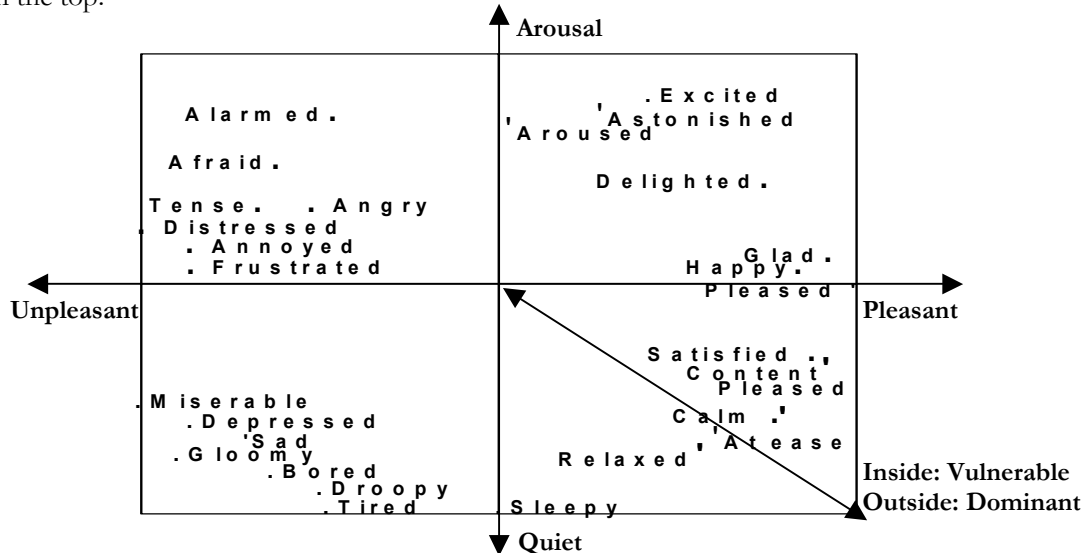


Figure 2- 5 Rusell's emotion-related word two-dimensional scaling

This research could give profound impact for the psychological implications and for the psychological understanding of facial affect. It has a direct implication for the field of affective computing.

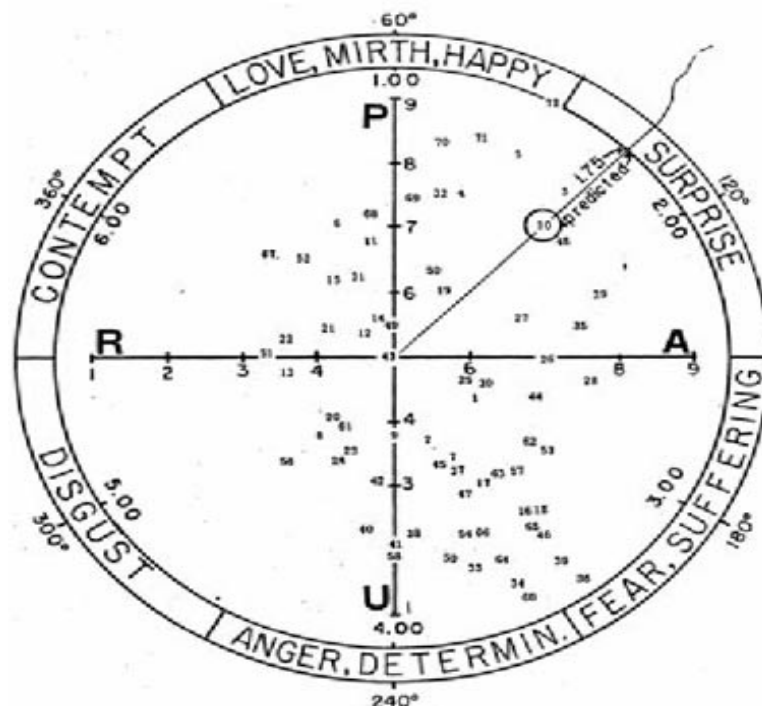


Figure 2- 6 The 'emotion circle' of Schlosberg (reproduced from [HEN98])



Russell's idea, actually, is based on the earlier work of the two dimensions of emotions 'pleasantness' and 'arousal' of Schlosberg (1952) [DES00]. The work of Schlosberg relates to an empirical analysis of six emotional expression, which provided a two dimensional representation of the expression space (see figure 2-6 above). This vein describes emotional states associated with points on a circle with reference to attention versus rejection and pleasantness versus unpleasantness [SCH52].

The question of how to characterize human emotions at best has clearly become an important concern for many researchers. Those categories are depended on semantic metalanguage we used. It is defined by using terms of words definition that are intuitively understandable as emotion state. In our case, the number of emotion categorization that we define related to the number of different facial displays we can be shown. However, this particular research is still in the gray area. This means we may find different context of emotions cannot be distinguished into different facial expressions, due to the fact that they are elicited by identical situation context.

### **Emotion-Eliciting Factor Information Extraction and Emotion Recognition**

After devising the emotion type that the system can recognize, the next step is to extract emotion-eliciting factors from the text prompt, to categorize and interpret those factors into one of those emotion types. These issues deal with the way QA systems represent and retrieve information that is transparent by their memory structure. This memory is the foundation of the ability level of the system to "recognize" emotions in human natural language. Human recognize the emotion from the text because they know the semantic meaning of each word and its relation to other words. In written conversation, some words' semantic precede emotions and the brain interprets said actions as emotions. A situation occurs and the brain interprets the situation, causing a characteristic physiological response in written sentences. Using regular matching pattern operation approach (see section 2.1), the QA system does not store those libraries. Therefore, at this moment it still needs human intervention to label each memory unit with emotions types. We left the implementation of context awareness as recommendation of future works.

In conversation human emotional expressions change continuously and many of these changes are synchronized to what is going on in current conversation. However most of the time, it does not change in a extreme manner. It depends on the intensity of the eliciting factor. In conversation, human reacts on emotions with certain intensity because they understand the intensity of the conversational context and the chronological relation between words and sentences. The intensity of the human emotion also depends on how important the goal is, does the situation uphold or violate the principles, does they get their preferences and what the current mood is [ELL93]. The QA system needs human to store and set those knowledge in its memory structure. During a conversation, the emotions intensity change depends of the situation and may activate other emotions.

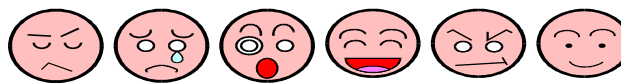
### **Facial Display Generation**

Face-to-face communication is primary a communication style. Facial displays can be viewed as communicative signals that help to coordinate the conversation [CAS94]. After the user's emotion state is recognized, the next step is to generate a facial display that reflects the arousal emotion. A fundamental issue in facial display generation is defining a set of categories we want to deal with. Human exhibit a wide repertoire of nonverbal behaviors consistent with their emotional state. They use facial signals to help to regulate the flow of conversation as much the same as intonation does, signaling emphasis and contrast, and as well as information related to turn-taking and control of the floor during an interaction (Pelachaud [PEL94], Takeuchi and Nagao [TAK93]). In conversation, the speaker may replace common verbal expressions by a specific facial expression to mention it even though it is not actually being felt at the present moment. Nonverbal cues also can be a substitute to contradict, emphasize, or regulate verbal messages. For this type of signal, in communication categorization almost all displays are situation/context dependent, except for those



displays known as facial emblems. It means that the appearance of such expressions is voluntary or learned (subconscious) and depends on what is being said. Facial signals also are used to express emotional signals, which may be used communicatively, to influence the other participant's behavior. Emotional facial displays are basically independent of the situation, that is, their meanings are the same wherever and whenever they appear.

Figure 2-7 shows the correspondence of some facial displays to emotions. Using the photographs displaying the facial expression identified by Ekman [EKM75], have shown that it is easiest to categorize happy faces whereas faces displaying disgust and fear were found to be most difficult. Faces displaying fear were often classified as sad and faces displaying disgust were often categorized as angry. However, there is some evidence to suggest that humans are sensitive to the degree of intensity of the facial displays. Off course, in some cases people are seemed to hide their emotion, for example when playing poker or fake another emotion. For reason of simplicity, we do not consider those cases here.



**Figure 2- 7 Some facial displays corresponding to emotions (disgust, sadness, surprise, happiness, angry and smile)**

In conversation, human shows two kinds of emotional expressions: first, related to stimulus response when they hear the utterance and second, related to cognitive processing when they realize the situation and the conversation content, and give the proper response/reply to the utterance. In order to be a believable system, a QA system should represent these aspects of the way human represent their feeling.

Another issue in the content of facial expression is regarding to the intensity of emotion. Most of the current systems use a certain threshold value to activate the emotion. When this occurs the corresponding facial display reflects the level of activation of the emotion. Once an emotion rises above its activation threshold, it decays over time back toward the base line level unless it continues to receive inputs from other processes or events.

## **2.4 Nonverbal QA Systems**

Before developing an automated system for emotion recognition and facial display generation in a QA system, one should decide what functionality it should have. A good reference point is the best-known emotion recognizer and facial display generator: the human brain. It may be not possible to incorporate all features of the human brain system into an automated system and some features may even be undesirable, but it can certainly serve as a reference point.

The first requirement that should be met in the development of an ideal automated emotion recognizer and facial display generator is that all of the stages of the facial display generation are performed automatically:

1. Emotion-eliciting factor information extraction
2. Recognition of emotion type
3. Facial display generation

In this section, an ideal nonverbal QA system is proposed (table 2-3 below) which could employ those three stages and has the properties of the human brain system. As the potential applications of a nonverbal QA system involve continues observation of the human user-system interaction over time, the memory structure should have enough information about human personality, system personality and human-to-human conversation. In order to be universal, the memory structure of





the system should contain every topic of conversation or at least it is capable to deliver appropriate answers and reactions. Despite of that, an ideal QA system should not contain everything, but should be adaptable of ability to learn from history. No constrain should be set on the interaction between the human user and the system, since the QA system should be the representation of the human in the form of a conversation machine.

**Table 2- 3 Properties of an ideal nonverbal QA system**

No.	Characteristic
1.	Unlimited number of memory structure units.
2.	Memory structure divided based on topics discussion.
3.	Correlation between current system's reply and entire conversation.
4.	Metaphor language recognition.
5.	No manual affective labeling in memory structure.
6.	Unlimited number of emotive lexicons dictionary.
7.	Stores user's personal data during conversation.
8.	Sets system's goals, principles, preferences and moods.
9.	Automatic affective recognition.
10.	Analyzes user's affective state based on user's string input and conversation content.
11.	Analyzes user's affective state based on user's personal data.
12.	Analyzes system's affective state based on system's goal, principle, preference and mood.
13.	Analyzes system's affective state based on reflection to user's affective and conversation content.
14.	Analyzes system's affective based on system's reply sentence.
15.	Replies sentence based on user's personal data.
16.	Replies sentence based on conversation flow.
17.	Replies sentence based on system's affective state and/or user's affective state.
18.	Feature adaptive learning facility from conversation history.
19.	Number of recognizable affective types.
20.	Shows stimulus reaction of facial display based on user's string input and system's affective state.
21.	Shows cognitive processed reaction of facial display based on user's affective analysis, conversation flow, conversation content and system's reply sentence.
22.	Intensity measurement.
23.	Unlimited corresponding emotions to facial displays.

An ideal system should perform robust automatic emotion recognition from arousal situations and show appropriate nonverbal displays accordingly. Considering the state of the art in a QA system development, the system should be able to converse every discussion type, deal with every human type and capable to “understand” every aspect of human emotion in language. An ideal system should capable of dealing with these and still perform an appropriate reaction.

An ideal system should be able to analyze all aspects of human emotions and distinguish one emotion from another emotion. A complete memory structure of the system is a prerequisite for achieving this. The memory structure of the system should contain enough information about human emotion on every memory unit, information about the system's goals, principles, preferences and moods, and enough information about the user's personality. Like human, the system also should learn to know the user from conversation. The longer the conversation the more knowledge of the system about the user, this knowledge establishes the information about the user's personality in the system's memory structure. In general, an ideal nonverbal QA system should:

1. Recognize user's affective state and its intensity from the user's string input and based on analysis of the entire conversation.
2. Set system's affective state and its intensity based on reflection of the user's affective state, and its own goals, principles, preferences and moods.
3. Reply the user's relevance to the user's string input and emotion situation. This reply also should be related to user's personality.



4. Show two kinds of appropriate facial displays: first, based on the user's string input and system's reaction affective state as stimulus response. Second, based on the user affective analysis and the system's reply sentence as cognitive processed response.
5. Analysis user's conversation and store important information that distinguish the user from other users.

In practice, it may not be possible to build a nonverbal QA system that can recognize every human emotion aspect and have a discussion in every topic. Still the development to such a system should be as broad as possible.

## 2.5 Related Work In Automatic Emotion Recognition and Facial Display Generation

In this section, approaches to automatic emotion recognition and facial display generation are discussed. The survey is divided into three parts, based on the ideal QA system discussed in section 2.3: emotion-eliciting factor information extraction, emotion recognition, and facial display generation. This section does not provide an exhaustive review of the past work on each of the problems related to automatic emotion recognition and facial display generation. Here, recently developed systems, which deal with both of them, are selectively discussed.

In table 2-4 below, • stands for 'yes', x stands for 'no' and – represents a missing entry. A missing entry either means that the matter at issue has not been reported or that the matter at issue is not applicable to the system in question. This table summarizes the characteristics of the surveyed papers regarding to three topics: QA system, emotion recognition and facial display generation. Since there is no research of emotion recognition in QA systems, the reviewed researches in this section do not only work in the field of emotion recognition from a text prompt or generating facial displays based emotion. However, we could possibly get our inspiration from the ideas of the researchers. Most of the works of these particular researches are on multi agents and character animation. In multiagents system, user's string input could be related to the action of one agent to another agent and reply sentence could be related the reaction of an agent to another agent.

**Table 2- 4 Properties of recently proposed approaches to a nonverbal QA system**

Reference	Properties of an ideal automated emotion recognition and facial display (table 2-3)																					
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
QA System/chatterbots																						
Weizenbaum'67	128	x	x	-	0	X	x	x	x	x	x	x	x	x	x	x	x	0	x	x	x	0
Wallace'95	>4100 0	x	•	-	0	X	x	x	x	x	•	x	x	x	x	x	x	0	x	x	x	0
Multi agents system																						
Elliott'92	-	-	-	•	128	•	•	•	•	•	•	•	•	•	x	•	-	24	x	x	•	72
Velasquez'93	-	-	x	-	0	•	•	•	•	•	x	x	x	x	•	x	•	6	x	x	•	0
Predinger'01	-	-	-	•	0	•	•	•	•	•	•	•	•	•	•	•	x	9	x	x	•	>9
Bazzan'01	-	-	-	-	-	-	•	•	-	-	•	-	-	-	-	•	x	24	x	x	•	0
Emotion recognition from human speech intonation																						
Nakatsu'99	0	-	x	-	0	X	x	•	x	x	x	x	x	x	x	x	•	8	x	x	•	0
Automated character animation system																						
Pelachaud'94	0	-	x	-	0	X	x	x	x	x	x	x	x	x	x	x	x	6	x	x	x	>7
Communicative facial display system																						
Nagao'93	61	-	x	-	0	X	x	x	x	x	x	x	x	x	x	x	x	0	x	x	x	26

Legend: • ='yes', x = 'no', - = 'missing entry'





## Emotion Eliciting Factor Information Extraction

Most of developed systems that are able to devise emotion-eliciting factor information still need manual human intervention. Following three experiments dealing with representing and extracting emotions' information on the system's memory structure:

### 1. *Emotive Lexicon Dictionary Look-up Parser*

This approach uses a list of lexicons, which are associated to different type of emotions. Those lexicons, which are composed by words or phrases, are selected from the way human people expresses their feelings. There is a comprehensive set of emotion categories that allows for the mapping of emotion eliciting situation to the emotion expression lexicon, and vice versa. For that purpose, a large dictionary is developed and classified by both emotion types and their intensity. The system uses a shallow word matching parser to extract affective state from the context.

An example of such a system using this model is that of Elliott [ELL95]. Elliot modeled a multi-agent world [ELL92], where each agent is able to reason about emotion episodes that take place in one another's lives. The implementation includes representations for twenty-four emotion types based on work of Ortony et al (see table 2-2). Each of the twenty-four emotion types has a set of eliciting conditions. When the eliciting conditions are met, and various thresholds have been crossed, corresponding emotions result (see [ELL93]). He used an extended base lexicon of spoken phrases that includes 198 emotion words associated with twenty-four OCC's theory emotion types. Those words describe relationship, mood and emotional intensity. To express the emotions, he used a schematic facial expression supporting over sixty expressions and as over three thousand dynamically constructed morphs. The system applies minimal the detection of user's emotional inflection. His work was not an attempt to parse affective information that the user is trying to mask. Rather it was an attempt to be an open channel for the user to communicate affective information to a computerized partner. Using this approach, it allows the user to teach the computer keywords in a new vocabulary relatively quickly and the system remains understandable no matter in which context the user is. Complexity of this approach focuses on the emotion meta-domain; therefore a relatively rich affect component can be developed using only a few tokens to identify the personality, the goal and so forth.

### 2. *Emotive Labeling Memory Structure*

This approach labels each unit of memory structure with one or more of the emotions types. Most of the examples for systems using this approach are automatic story telling systems and automatic digitizer for cartoon movies. Each dialog sentence of each actor is labeled with an emotion type and decomposed in its phonological representation. Therefore, the system can show appropriate intonation and nonverbal display when it reads the dialog. One of the examples is the work of Pelachaud et.al.[PEL94]. In their research, the input is assumed as a file containing an utterance already decomposed and written in its phonological representation with its prosody in its bracketed elements. At each input, it specifies the desired affectual parameters and their intensity. The system uses a set of rules to process those parameter and phonological representation to produce high quality animation of facial expressions and head movements as automatically as possible in conjunction with meaning-based speech synthesis, including spoken intonation. Using this approach means to sidestep the issue of emotion recognition. The modeling of affect is not meaning based and it needs human manual work to label each memory unit.

### 3. *Goal-Based Emotional Reasoning*

This approach sets some goals, principles, preferences and moods in the system. Extraction of emotion-eliciting factors leading to emotions falls into four major categories: those rooted in the effect of an event on the goal of the system, those rooted in the standard and principle



invoked by an act of the user, those rooted in tastes and preferences with respect to an object (including the user treated as an object), and lastly a selected combination of the first three categories. Another way to view these categories is that they are rooted in the system's assessment of the *desirability or undesirability* of some event, the *praiseworthiness or blameworthiness* of some act, the *attractiveness or unattractiveness* of some object, or selected combinations of these assessment. This approach based on most human interaction deliberates around people's individual need and goals. These lead the QA system to idiosyncratic, internally motivated behavior and to emotional responses to situation that arise.

An example of the system using this approach is the work of Elliot [ELL93a]. This is also a model of *emotive lexicon dictionary look-up parser*. He developed his simulation model using this approach to simulate social interactions between agents in incorporated models of individual affect and personality. Each agent interprets situations that are characterized in terms of the way they may or may not meet the eliciting conditions of emotions. The emotions result may be expressed through activated behavioral channels as new simulation events that might perturb future situations. In addition, agents use a case based heuristic classification system to reason about the emotions of other agents' personalities that will help them to predict and explain future emotion episodes involving observed agents. Embodied in the simulation system, Elliott used a set of rules for the mapping from four categories emotion eliciting factors above into twenty-four emotion types, based on the work of Ortony, Clore and Collins (OCC theory - table 2-2). Mapping this simulation to a QA system, we could consider the interaction of the system and the user as social interactions of two agents, which both of them have their own goals, principles, preferences and moods.

## Emotion Recognition

For the activation of an emotion, Elliott [ELL93], Velaquez [VEL98], Predinger & Ishizuka [PRE01], and Bazzan & Bordini [BAZ01] proposed the use of threshold values by counting all associated elicitation factors, the excitatory (positive) and inhibitory (negative), from other emotions. They used an activation level range  $[0, \text{max}]$  where *max* is an integer value determined empirically. All emotions are always active, but their intensity must exceed a threshold level before they are expressed externally. The activation process is controlled by a knowledge-based system that synthesizes and generates cognitive-related emotions in the system.

Regarding the value of emotion-eliciting factors, Hendrix and Ruttkay [HEN98] worked on the research to test the perception-based closeness matches of six universal emotional expression space and delivered discreet values of distances between those six emotional expressions (the researcher have no information about fear since no fear was included in their dataset). In their research, they used the work of Schlosberg, which provided a two dimensional representation of the expression space (see figure 2-6), based on the observation that people made very confined mistakes when identifying expression from photos of emotion faces. Each emotional expression was only mistaken for two others and such a way that easy to mistake 'neighboring' emotional expression result in a circular graph (see figure 2-6). The neighboring emotional expression can be said to be close to each other with respect to perception. These distance values can be used to set a weight factor to the changes of one emotion to other emotions and to set the threshold as corresponding to one of the six basic emotional expressions (see table 2-5).

Table 2- 5 Distances between six emotional expressions

	Happiness	Surprise	Anger	Disgust	Sadness
Happiness	0	3.195	2.637	1.926	2.554
Surprise		0	3.436	2.298	2.084
Anger			0	1.506	1.645
Disgust				0	1.040
Sadness					0



## Facial Display Generation

In most of the works in facial display generation are used one to one corresponding facial display and emotions, distinguished by intensity ([Elliott [ELL93] and Predinger & Ishizuka [PRE01]). The other works used the correspondence between communication categorization and Ekman & Friesen's Facial Action Coding System (FACS) (Takeuchi & Nagao [TAK93]) and between emotions with FACS (Pelachaud et al. [PEL94]). FACS is a notation to describe visible facial expression based on anatomical studies. FACS describes temporary changes in facial appearance, how a feature is affected by specifying its new location and the intensity of changes.

## 2.6 A New Approach: My\_Eliza

Most of human communications are about emotional information, which in turn is a powerful motivator in human behavior. We base our feelings and emotional responses not so much upon what another person *says*, but upon what another person *does* [KIN97]. Engaging a QA system with capability to capture human's affective states from written texts and show appropriate nonverbal signals for them, would therefore be highly beneficial for enhancement communication skill in applications such as the therapy system (stress monitoring), education (long distance learning), movie making (cartoon) and development of advanced multi-modal human-computer interfaces (HCI), such as talking faces, personal assistance, and multimodal online helpdesk.

In the previous section, various issues related to the problem of introducing nonverbal signals to QA systems have been discussed. All of them are intriguing and none has been solved the general case. A number of conclusions have been reached.

- Most of the existing QA systems have no ability of anaphoric analysis and ability to restrict the conversation on its topic.
- None of the QA systems has the capability to capture user's affective states from written text. Most of the existing QA systems that have the capability to show nonverbal signals are not based on affective reflection of user's affective states nor based on affective state of the system as the reaction.
- Most of the methods to detect human's affective states are not based on written text.
- A number of nonverbal analyzing systems require user effort to label the input text manually.

The main goal in the development of my\_Eliza presented in this thesis was the enhancement of the state of the art of automated nonverbal signal construction during a dialog. To wit, the aim was to develop a fully automated system for nonverbal signal construction based on the affective state during a dialog that easily can be used for behavioral research purposes and integrated into multi-modal HCI systems. As a result, the research was focused on:

1. User-independent, fully automatic affective information extraction from user texts in a dialog.
2. Automatic reply construction, automatic human computer interaction based on typed natural language.
3. Semi automatic user's affective analysis.
4. Automatic construction of two types my\_Eliza's nonverbal signals (based on user's affective analysis): first, its spontaneous reaction to the user input text. This reaction based on spinal brain reasoning (stimulus-response). Second, to convey its reply sentence. This type of nonverbal signals is the result of cognitive process based on system's goal, status and preferences (GSP).

Those design requirements (see figure 2-8 below) emerges from a QA system's application domain for the development of my\_Eliza. My\_Eliza performs human computer conversation based on natural typed language combined with appropriate nonverbal signals, as a woman psychoanalyst.



The system's aim is to keep the conversation going and make the user comfortable to talk with her. The user should consider my\_Eliza as realistic figure in order to engage in natural conversation.

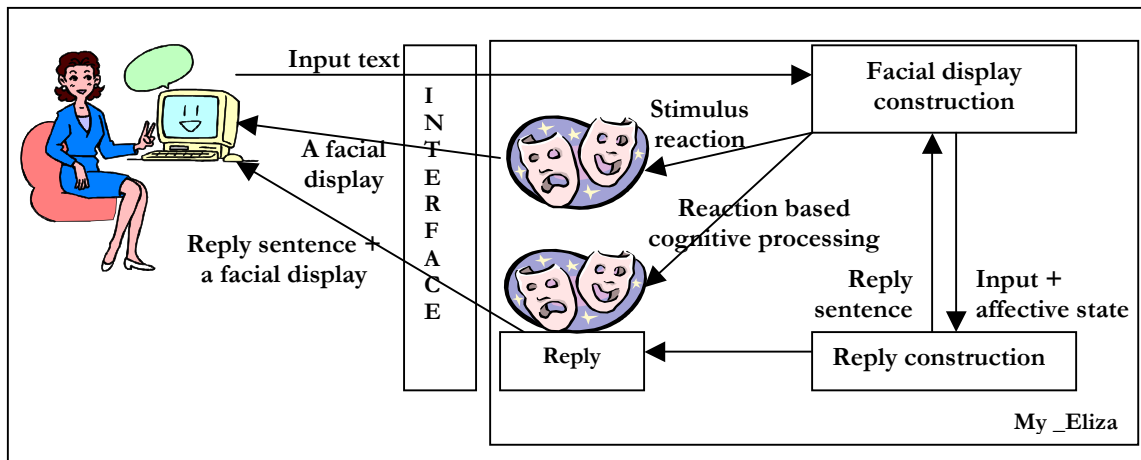


Figure 2- 8 My\_Eliza's design requirements

In figure 2-9, like original Weizenbaum's Eliza, the system poses analytical questions for every answer the user gave it. This certainly guarantees the continuity of the conversation. My\_Eliza used facial displays to show reflection or concern of the user's feelings, for example if the user is distressed then my\_Eliza will show her sorry for his/her sadness, if the user is happy then my\_Eliza will be happy for his/her happiness and so on. When the users typed their text, the system analyses the user's affective state based on the text and generates facial displays in response to the situations that impinge on the system's concerns. This facial display is a spontaneously reaction based on my\_Eliza's "spinal brain" reasoning process. After my\_Eliza has knowledge about user's affective state, the system will construct the reply sentence and generate facial displays based cognitive processing.

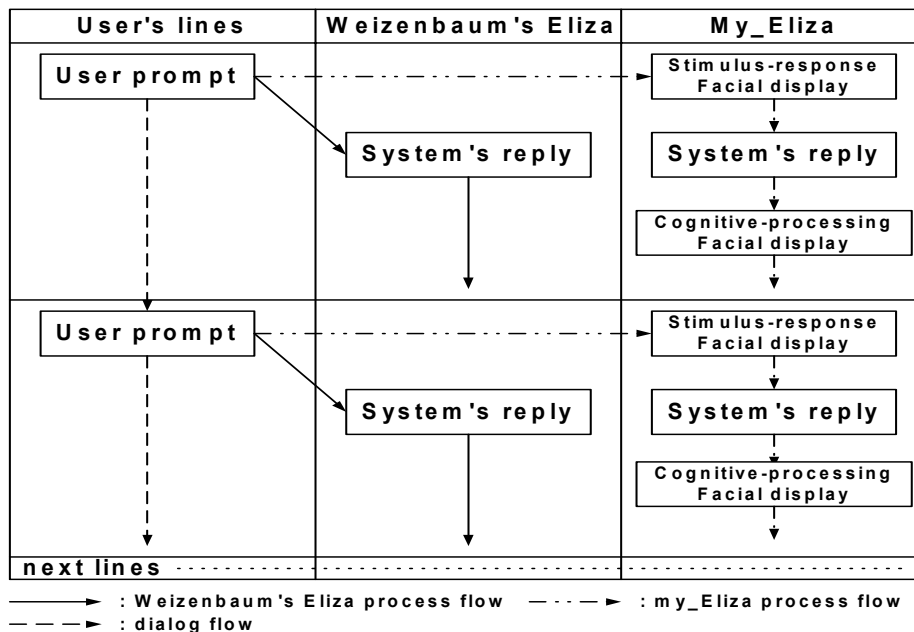


Figure 2- 9 Comparing the process flow schema between Eliza and my\_Eliza



My\_Eliza has not performed in full range of an ideal QA system yet; the system still has limited number of memory structure units. As a result there may occur misunderstanding or irrelevant conversation with the user. Off course, this may also happen in human conversation, for example if people do not listen or when we are not willing to process linguistic inputs. The facial display generation still uses the corresponding one-to-one with six universal emotions distinguished by three level intensity: low, medium and high. As a first step to a nonverbal QA system, a first prototype has been developed. We hope and expect that many improved versions will appear, similar to the work of Weizenbaum's Eliza. The implementation of the first prototype my\_Eliza prototype-1 is explained in chapter 6. Nevertheless, table 2-6 below shows the property of my\_Eliza global design as an ultimate system, which will be discussed in chapter 4.

Table 2- 6 My\_Eliza's properties

No.	An Ideal QA System Characteristic	My_Eliza
1.	Unlimited number of memory structure units.	>1200 <sup>2</sup>
2.	Memory structure divided based on topics discussion.	Yes
3.	Correlation between current system's reply and entire conversation.	Yes
4.	Metaphor language recognition.	No
5.	No manual affective labeling in memory structure.	No
6.	Unlimited number of emotive lexicons dictionary.	>100 <sup>3</sup>
7.	Stores user's personal data during conversation.	Yes
8.	Sets system's goals, principles, preferences and moods.	Yes
9.	Automatic affective recognitions.	Yes
10.	Analyzes user's affective state based on user's string input and conversation content.	Yes
11.	Analyzes user's affective state based on user's personal data.	Yes
12.	Analyzes system's affective state based on system's goal, principle, preference and mood.	Yes
13.	Analyzes system's affective state based on reflection to user's affective and conversation content.	Yes
14.	Analyzes system's affective based on system's reply sentence.	Yes
15.	Replies sentence based on user's personal data.	Yes
16.	Replies sentence based on conversation flow.	Yes
17.	Replies sentence based on system's affective state and/or user's affective state.	Yes
18.	Feature adaptive learning facility from conversation history.	No
19.	Number of recognizable affective types.	26 <sup>4</sup>
20.	Shows stimulus reaction of facial display based on user's string input and system's affective state.	Yes
21.	Shows cognitive processed reaction of facial display based on user's affective analysis, conversation flow, conversation content and system's reply sentence.	Yes
22.	Intensity measurement.	Yes
23.	Unlimited corresponding emotions to facial displays.	18 <sup>5</sup>

<sup>2</sup> The memory structure is easy to be extended

<sup>3</sup> The number of lexicons in dictionary is easy to be extended

<sup>4</sup> Using OCC theory on table 2-1 plus two additional emotions: uncertainty and neutrality

<sup>5</sup> Extendable, current number of facial displays only corresponding one-to-one six universal emotion distinguish each of them by three level intensity: low, medium, and high



## Chapter 3 Artificial Intelligence

No one knows a strict definition of Artificial Intelligence (AI). Russel and Norvig 1995 [RUS95] summarized AI definitions into two main dimensions with respect to possible goals to pursue in the AI area (see table 3-1). The ones on top are concerned with thought *processes and reasoning*, whereas the ones on the bottom address *behavior*. Also, the definitions on the left measure success in terms of human performance, whereas the ones on the right measure against an ideal concept of intelligence, which is called *rationality*. A system is rational if it does the right thing. Based on an ideal nonverbal QA system specification in section 2.4, a nonverbal QA system at least has to perform the ability to act like human. The issue of acting like a human comes up primarily when a nonverbal QA system has to interact with people, as a natural language processing system having a dialogue with a user, or as facial display generator showing emotional based appropriate facial expressions to a user based on current user's affective state determined from text prompts. This system must behave according to certain normal conventions of human people interaction in order to make them understood. The underlying representation and reasoning in such a system is based on a human model.

**Table 3- 1 Four possible goals to pursue in Artificial Intelligence**

Systems that think like humans	Systems that think rationally
Systems that act like humans	Systems that act rationally

In order to act like humans, a nonverbal QA system needs to possess capabilities in natural language processing, knowledge representation and automated reasoning for emotion recognition, and machine learning to adapt to new circumstances from conversation history. Therefore, this chapter contains an introduction to natural language processing in section 3.1 and an introduction to knowledge based system in section 3.2. We do not present machine-learning subjects in this thesis since current development of my\_Eliza still does not have this ability (see table 2-6, my\_Eliza's properties). Since a QA system is known also as chatterbot that has a level of intelligence and autonomy, this system also can be viewed as an *agent*. This issue is discussed in section 3.3. Finally, section 3.4 discusses some general peculiarities of the AI application development process.

### 3.1 Natural Language Processing

Mary D. Harris in [HAR85] referred *Natural Language* as conversational speech and *Natural Language Processing* (NLP) referred to a program that deals with natural language in some way or another. As language is central to so much of human activities, NLP researchers have put a major research focus on the ability of intelligent system to handle human language based interaction. Getting computers to understand everyday language in contrast to computer code has occupied them. They strive to develop computer programs that understand language as it is used in everyday discourse with all its ambiguities, inference and omissions.

Central to NLP is a parser, a computer routine that takes a sentence from everyday language as input and extracts a meaning for the sentence as output. It parses a sentence into a sequence of tokens. The token can be a character, word, or phrase. Each token has some set of distinguishable properties and the sentence meaning is interpreted by figuring out what meaning corresponds to those tokens in that order. The underlying meaning of the sentence often goes beyond the words and the grammatical structure of the sentence. For the parser to adequately understand a sentence it must in addition to computing grammatical structures take into account





the discourse and interaction factor that provide clues to the speaker's intention, meaning and purpose. The parser allows the system to both understand the user's input and to generate responses to that input. Any human language uses a vast range of grammatical, lexical, and discourse devices, alone and in combination, to express an even greater range of meanings, intentions, desires etc. The ultimate goal of NLP research is to develop a parser that can take any utterance from everyday language and make sense out of it in the same way as a person does. Although great improvements have been made in the development of parsing technologies, NLP still remains far short of this goal and the program design must take into account the constraints of the parser.

Original Weizenbaum's Eliza was known, as the first attempt of a NLP program, however cannot be told as an intelligent machine. Technically, psychoanalyst Eliza was actually unable to understand people's problems to the depth of any other human being. Eliza could only manipulate syntax (grammar) by checking some input keywords. If the appropriate keyword has been selected, the parser continues to check the sentence pattern against each decomposition rule of that keyword. If not, the parser must explore another keyword node in the search space. My\_Eliza system uses the parser not only for generating a reply sentence but also to extract words or phrases that have correspondences to one or more emotion types. In order to know which emotion type a word or phrase belongs, this system has an Elliott-like emotive lexicon dictionary (see section 2.5 about Elliott's work). The system receives a sentence and checks word by word against the list in the dictionary (see chapter 4 for detail explanation). From this operation, my\_Eliza could conclude what type of emotions appears in the sentence.

## 3.2 Knowledge Based System

A knowledge-based system is a computer program that consists of problem solving knowledge in a specific domain. The knowledge comes from many knowledge sources as an expert/specialist. A knowledge-based system can solve a problem by a reasoning process that depends on the knowledge base domain. Sestito and Dillor in [SES94] determined the basic structure of a knowledge-based system. It consists of two components (see figure 3-1), such as: knowledge base and inference engine.

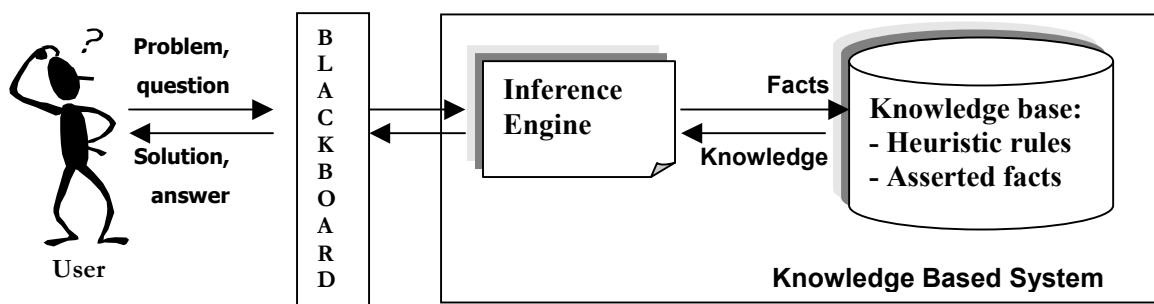


Figure 3- 1 Basic schema of a knowledge-based system

A knowledge base is a database in a specific knowledge domain. It contains a representation of the state of the problem domain in the form of heuristic rules. The inference engine uses the rules to infer appropriate conclusions based on relevant portions of the knowledge base and a set of facts that forms the current input to the system (stored on a so called blackboard). Each of the rules in a knowledge base maps a specific state in the problem domain to one or more actions the system performs. The rules take the following form:

if <list of conditions> then <list of actions>

where <list of conditions> are associated the asserted facts in the knowledge base and <list of actions> are actions that may update other facts in the knowledge base or influence



the actions of the system. The inference engine makes the connection between the facts and the rules in the knowledge base. Upon assertion of facts, the inference engine considers all rules. When a state of problem domain matches a rule, the rule is said to be *fired*.

In a knowledge-based system, correctness and completeness of the knowledge base is the most important factor. The development process of knowledge base takes place through several steps; the first step is knowledge acquisition to get representation of the knowledge from knowledge sources. The next step is to transform the knowledge representation into codes of knowledge based on rules and facts form and to store them into a database. Finally, the database can be used to solve problems in its domain by executing the inference process. Frank Puppe in [PUP93] defines knowledge acquisition as a process of collecting knowledge and a process of developing a construction model using a transformation from the collection of knowledge to the knowledge representation. The knowledge acquisition can be done in two ways, such as:

- (1). Experts construct their knowledge into a knowledge representation forms, transform them into rules and facts and store them into a knowledge base.
- (2). A knowledge engineer gives assistance to experts. The experts communicate all about their knowledge to a knowledge engineer. The engineer is responsible to construct the knowledge and stores them into a knowledge base.

However, it is very difficult for the expert(s) to learn something that is not connected with their field, such as computer programming. In a common situation, they just trust a knowledge engineer to transform their knowledge into a representation on knowledge base.

For my\_Eliza, we have defined a set of rules that specify the emotion recognition process of the system, according to the current dialog between my\_Eliza with the user. We call these rule-sets **preference rules**, since they indicate preferences to exhibit system's "preference" reaction affective state rather than performing explicit actions, such as facial displays. Every rule in the set defines several conditions to activate the rule and a preference that is expressed upon activation. The use of preferences reaction affective state instead of facial display actions in the rules arises from the desire to simulate the way how human's reaction are based on his/her feelings. These preference rules will be explained in more detail in section 6.2.5.

### 3.3 Intelligence Agent

The term *agent* has become widely used in computer science and AI, without a universally accepted definition. According to Wooldridge and Jennings [WOO94], in computer science, the term *agent* is generally restricted to computer systems, which possesses the properties of:

- *Autonomy* - being able to operate without the direct intervention of humans or others,
- *Social ability* - being able to interact with other agents,
- *Reactivity* - being able to perceive and respond to their environment, and
- *Pro-activeness* - being able to exhibit goal-directed behavior by taking the initiative.

These four properties lead to two main characteristics of an agent: a level of *intelligence* and *autonomy*.

Russel and Norvig [RUS95] define an agent as anything that can be viewed as *perceiving* its environment through *sensors* and *acting* upon that environment through *effectors*. See figure 3-2 below for a schematic view of an agent interacting with its environment. The definition above allows the term to refer to entities, which are humans or software as an abstraction of the classes of intelligent entity, which includes both humans and programs. Viewed this way the agent is a mapping from percepts to action, that can influence the environment in which the agents live. But also this definition may include a rather broad range of software because it depends on what is exactly meant with the environment in which the agent's lives.



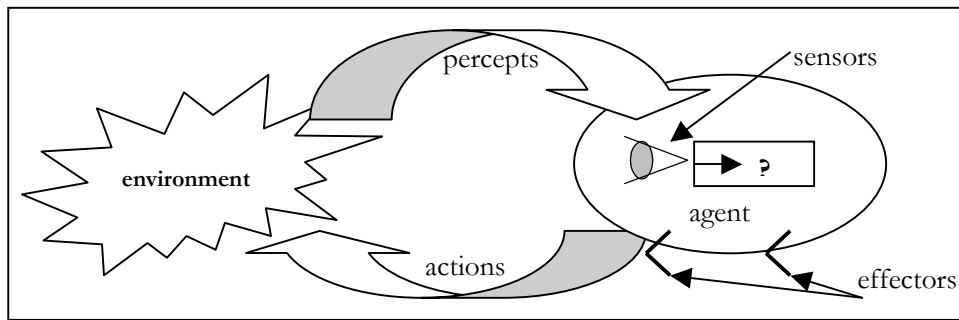


Figure 3- 2 Agent interacts with environment through sensors and effectors

For Russel and Norvig, *intelligent* means rational and automatic. As mentioned in the beginning of this chapter, to act rational means the agent should do the ‘right’ thing. It is the ability to behave appropriately in an uncertain environment, where appropriate behavior is that which maximizes the likelihood of success in achieving the agent’s goal. This definition of intelligence recognizes degrees/levels of intelligence that is sufficient in making the agent succeed in solving problems, anticipating the future and acting so as to maximize the likelihood of achieving goals. Performing actions in order to obtain useful information is an important part of rationality.

Russel and Norvig also state that a system is autonomous to the extent that its behavior is determined by its own experience. Franklin and Graesser [FRA96] conclude some definition of autonomous agent from several textbooks as “a system situated within and a part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to effect what it senses in the future”. For an agent to be act of its own agenda this would mean that it would be able to cope with unforeseen changes in its environment without the need for help from its engineer. A clear distinction can therefore be made between the concepts of autonomy and automatic. An automatic system is something whose predictable behavior as soon as its internal basis of decision-making is known. An autonomous system on the other hand is a system that capable to “make up” its own mind. An autonomous system has its own relevant self-knowledge and motivations to be able to control itself.

My\_Eliza system can be viewed as an agent since its behavior has to be autonomous and it has its own level of intelligence. My\_Eliza can perceive the conversation with the user though its sensor to receive user’s text prompts. The system formalizes the reasoning process that leads to perform a particular communicative act. The system analyzes current conversation flow using its own knowledge to recognize the emotion. From this point, it combines all knowledge to construct appropriate reply sentences and appropriate nonverbal signals, then delivers them as the real time feedback to the user through its effectors: text prompts and facial displays. As non-talking computer character, my\_Eliza is capable of various forms of expressive and communicative behavior. Some nonverbal signals are closely tied to spoken utterances. Those kinds of signals will be tried to be drawn by my\_Eliza as emotive behavior to exhibit contextually appropriate nonverbal signals. These actions are relatively unpredictable and would effect what it senses later in the conversation between my\_Eliza and its users.

### 3.4 AI Application Development

According to Roger Pressman [PRE98], the whole process of developing and maintaining a software product is called a *software engineering*. This process troughs several phases, which are called *life cycle*. All the activities of those phases are required to define, design, develop, test, operate and maintain a software application. There are many different models and variations of those models emphasize different aspects of software life cycle and are appropriate for a range of situations.



Three of them commonly used software life cycle models: waterfall, prototyping and incremental development.

*Waterfall* is a conventional software engineering paradigm, which is a highly structured life cycle model. At each phase, the result must be built and verified correctly. “Correct” means the application meets the specifications and all the requirements. Waterfall is a sequential model in one direction to final stages. When developing an AI system like a knowledge-based system, the expert is usually closely involved in all stages of the development [PAN01]. Such constant involvement results in a dynamic change of specifications and requirements, in contrast to conventional software engineering where specifications and requirements are essentially static. Hence, as a conventional software engineering, the waterfall is not the most suitable for an AI application development. It is because of a dynamic change of specification therefore the engineers will often require revisiting prior phase. This action is not intended if the engineer uses the waterfall model.

*Prototyping* is a multiple iterations software engineering paradigm, which has to be performed through the phases of knowledge acquisition, coding and testing until a final prototype of intended AI application is obtained. This method has impediment since the complexity of the application to be built, the time and costs per prototypes do not scale linearly from the effort to encode the partial prototypes. Consequently, prototyping usually is used as a life cycle model when a problem is sufficiently small or as a part of a more complex model.

*Incremental development* is a refinement of the waterfall model as the prototyping is integrated. In this model the application development phases of requirement analysis and global design are the same as in the waterfall model. Successive incremental approach splits the phases concerning the detailed design and implementation. An important characteristic of this model is that it allows the addition of new requirements in the developing process. Thus, it facilitates dynamic development of system specifications.

Quoting from Pantic [PAN01], independently of the chosen software engineering model, the development of an AI application may be split into following phases:

1. *Assessment and scooping.* In this phase, the primary actions are: selects the target user, analyzes requirements, estimates potential in future, and selects the development team.
2. *System specification.* In this phase, the actions are: makes a conceptual model of the application, designs model of the application, selects appropriate AI techniques to be employed and the appropriate software tools for developing the future system, and formalizes an initial prototype in the form of an offer. The first prototype, as the result of this phase, should focus on a general description of the future system’s functionality, behavior and structure. It is reasonable to make such a prototype before proceeding with more detailed design of the system.
3. *Refinement and implementation.* This phase involves the development of a detailed design of the application. For example, in the incremental development model, this phase involves selecting the appropriate successive increments that form a proper functional extension of the design model. Each increment further has to be coded and tested. Note that all increments should fully compatible with all previous and future development. When the last increment is implemented, the whole specification must be as complete as it should be in a waterfall model. All subsystems can now be integrated and tested as a whole.
4. *Final system, integration, testing and transfer.* In this phase, the final system of the AI application has to be validated and verified whether it satisfies all the specified requirements and all of the additional specification during the development process.
5. *Maintenance, enhancement and support.* In order to evolve with the environment, the application needs this phase, since the working environment of an AI application is usually highly dynamic.



We also develop my\_Eliza using this approach and through five phases above. We take incremental development model as my\_Eliza engineering life cycle model (see section 4-1 for detail explanation) and currently, my\_Eliza development is in the system specification phase, which indicated by the first prototype of my\_Eliza as the result of this thesis work (see chapter 6). The assessment and scooping of my\_Eliza development can be found in chapter 1 and chapter 2, while the report of system specification phase can be found in the rest of this thesis report.



## Chapter 4 My\_Eliza Global Design

As mentioned in section 3.4, we use an incremental development life cycle model in my\_Eliza development. Using this model, my\_Eliza development is split into two main procedures. The first procedure is dealing with the requirement analysis and global design, which uses the waterfall model. The last procedure is dealing with successive incremental approach of the detailed design and implementation. The requirement analysis of the system was determined in chapter 1 and 2, while this chapter will discuss the global design of my\_Eliza system. The detailed design and implementation of each incremental step will be discussed in chapter 5 and 6. As explained in section 3.4, independent of the chosen software engineering model, at this point we enter the *system specification* phase. Thus, this chapter will formulize the first and second action of system specification phase: makes a conceptual model of the application and designs model of the application.

This chapter is organized as follows. The issue of incremental development of my\_Eliza implementation is explained in section 4.1. This section also discusses my\_Eliza system architecture, which consists of two main layers: (1) a layer to construct reply sentences and (2) a layer to construct facial displays. The next step, the data or knowledge extraction that will be used in both layers, is presented in section 4.2. Finally, the detailed explanations about both layers are presented sequentially in section 4.3 and 4.4.

### 4.1 System Architecture

Based on system's requirement in section 2.5, our research approach can be outlined into three major (incrementally) implementation layers (figure 4-1):

1. Create a dialog box that can engage in human conversation based on typed natural language and recognize the user's affective state and the system's reaction affective state.
2. Build a stimulus-response of facial displays based on spontaneously spinal brain reasoning on user's string input.
3. Build a cognitive processor of facial displays based on anaphora analysis, pragmatic analysis, dialog content and system's goals, status, and preferences.

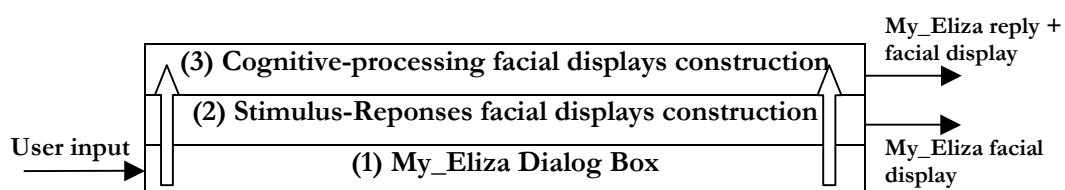


Figure 4- 1 My\_Eliza three major implementation layers

The development of implementation layer 2 and 3 can be released in five steps:

1. Study many samples of conversation with appropriate nonverbal signals.
2. Try to establish some relatively general rules on conversation.
3. Design and implement a process model.
4. Encode those rules into the process model.
5. Test the model.
6. Analyze if this model accounts for certain aspects of human verbal and nonverbal conversation.
7. Analyze the model for further development.

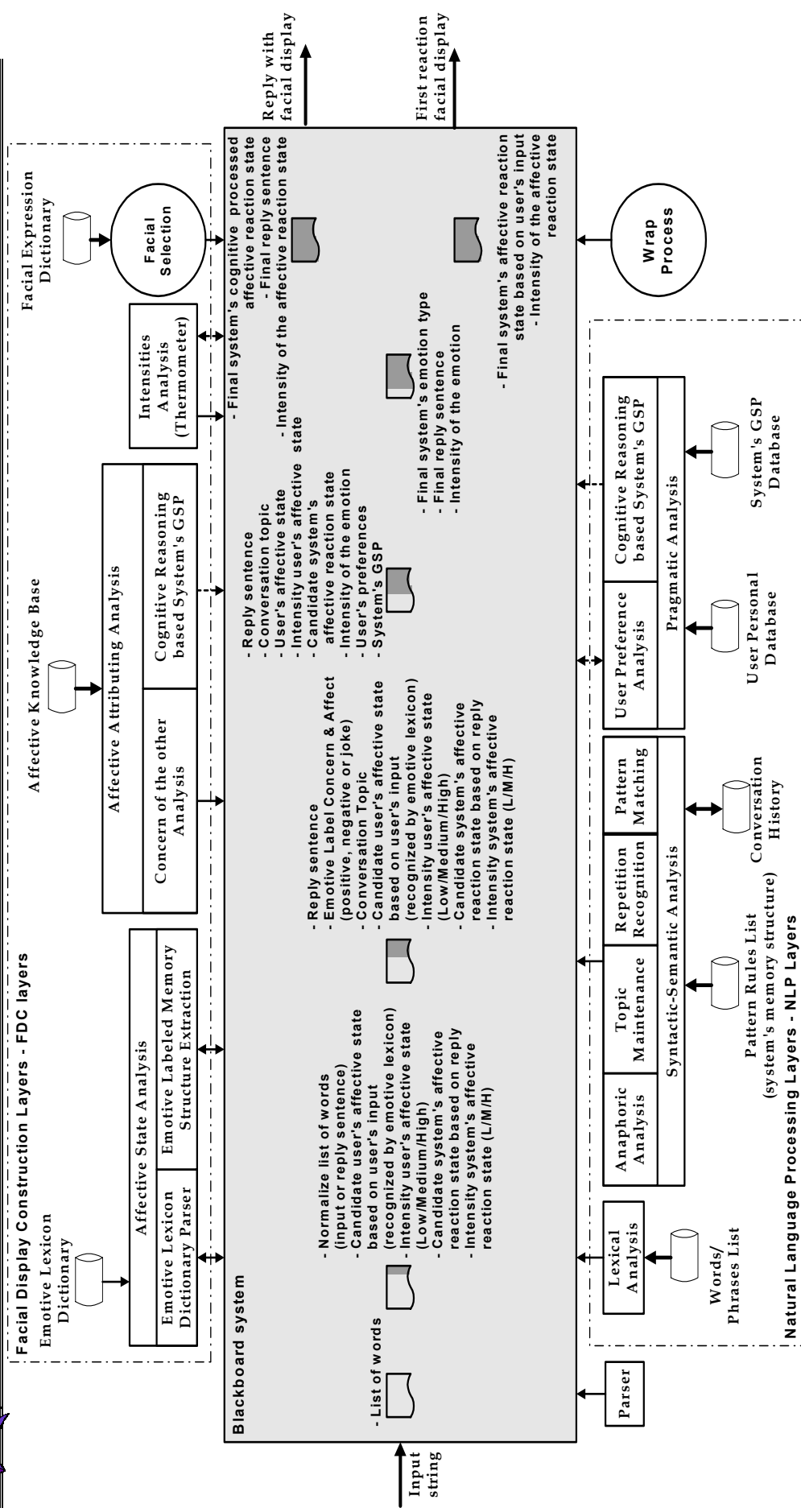
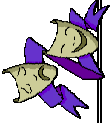


Figure 4-2 My\_Eliza blackboard system architecture



The architecture of my\_Eliza is illustrated in figure 4-2 above. By taking the idea of message passing on a blackboard system, the message flow and message process are always on the blackboard. If a new message comes, it will be analyzed, synthesized, and the result will always be put back on the blackboard. Therefore, every successor process can process the message based on the results of its predecessor process. Each process is assigned into a specific duty and resulting specific output. At the end of the message flow, there is a correct output ready to deliver out of the system. Using that idea, in my\_Eliza, the message is the user's **string input** and the results are the **reply sentences and facial displays**. The shadow on the message object figure shows that it is processed gradually until the shadow covers all the surface of the object figure; it means that the result has been generated.

My\_Eliza consists of two major parts: Natural Language Processing Layers (NLP-Layers) and Facial Display Construction Layers (FDC-Layers). When the user types an input string, my\_Eliza puts it on my\_Eliza blackboard system and the parser parses it into words. NLP-Layers accept the result to construct the reply sentence. NLP-Layers are built up by three layers: *lexical analysis layer*, *syntactic-semantic analysis layers* and *pragmatic analysis layer*. The Lexical Analysis layer normalizes the input by eliminating incorrect or incomplete words or phrases and checks relations between words or phrases. The syntactic-semantic analysis layer uses the result of the Lexical Analysis layer to perform a pattern-matching operation of input text against a *list of pattern rules* based on current topic and the system's previous response. Both of them are stored in the *conversation history* log. The pragmatic analysis layer checks a candidate reply sentence from syntactic-semantic analysis layer whether it is relevant with the current system's GSP and the user preferences. All information about users' preferences are stored in a *user preferences database*.

FDC-Layers work simultaneously parallel to construct my\_Eliza facial display. FDC-Layers consist of four layers: *user's affective state analysis layer*, *affective attributing analysis layer*, *intensity analysis layer* and *facial selection layer*. The affective state analysis layer identifies emotive lexicons in user input text and system's reply sentence against the *emotive lexicon dictionary*. This layer also extracts emotive labels from the system's memory structure. The affective attributing analysis layer uses all those information to construct two types of my\_Eliza's facial display: *stimulus-response facial display* and *cognitive-processed facial display*. The stimulus-response facial display is generated by a rule-based system using the information about the current user's affective and system's reaction affective state. In this process emotion-based reasoning plays a very important role. The cognitive-processing facial display is also generated by a rule-based system using the information about the current user's affect, system's reply sentence, system's GSP and the system's reaction affective state.

The Intensity analysis layer uses six Ekman's universal emotions' "thermometer" (happiness, sadness, disgust, surprise, fear and anger) to measure emotion's intensity. The highest intensity of all the six emotions of the thermometer tells which emotion type is activated. If the degrees of the six thermometers are equal, this means neutrality is active. The facial selection layer selects an appropriate facial expression in my\_Eliza's facial display files based on the result of affective attributing analysis layer and intensity analysis layer. The facial selection layer has twenty-four my\_Eliza's facial display files (three level of intensity for each of the six emotion types and for two additional emotion types: uncertainty and neutrality). The *wrap process* is responsible to deliver the final results (combination of the NLP-Layers result with the FDC-Layers result) and displays them to the user. Note, that all the processes in the FDC-layers will deal with Ekman's universal emotion types (happiness, sadness, disgust, surprise, fear and anger plus neutrality) except the **affective attributing layer**. This layer identifies one of twenty-six emotion types in table 4-1 (below) as system's reaction affective state. Table 4-1 actually is the list of OCC's theory emotion types plus two additional emotion types: normal and uncertainty. The replies and facial displays construction process in my\_Eliza system is explained clearly in the system's flowcharts below (figure 4-3 and figure 4-4).





Table 4- 1 My\_Eliza's twenty-four emotion type to classify system's reaction affective state

No.	Name and Emotion Type
1.	<b>Joy:</b> pleased about an event
2.	<b>Distress:</b> displeased about an event
3.	<b>Happy-for:</b> pleased about an event desirable for another
4.	<b>Gloating:</b> pleased about an event undesirable for another
5.	<b>Resentment:</b> displeased about an event desirable for another
6.	<b>Sorry-for:</b> displeased about an event undesirable for another
7.	<b>Hope:</b> pleased about a prospective desirable event
8.	<b>Fear:</b> displeased about a prospective undesirable event
9.	<b>Satisfaction:</b> pleased about a confirmed desirable event
10.	<b>Relief:</b> pleased about a disconfirmed undesirable event
11.	<b>Fears-confirmed;</b> displeased about a confirmed undesirable event
12.	<b>Disappointment:</b> displeased about a disconfirmed desirable event
13.	<b>Pride:</b> approving of one's own act
14.	<b>Admiration:</b> approving of another's act
15.	<b>Shame:</b> disapproving of one's own act
16.	<b>Reproach:</b> disapproving of another's act
17.	<b>Liking:</b> finding an object appealing
18.	<b>Disliking:</b> finding an object unappealing
19.	<b>Gratitude:</b> admiration + joy
20.	<b>Anger:</b> reproach + distress
21.	<b>Gratification:</b> pride + joy
22.	<b>Remorse:</b> shame + distress
23.	<b>Love:</b> admiration + liking
24.	<b>Hate:</b> reproach + disliking
25.	<b>Normal:</b> do not have dominant emotion
26.	<b>Uncertainty:</b> confuse or unsure

Algorithmic procedure of my\_Eliza blackboard system is defined by the following steps:

1. Generating a stimulus-response nonverbal signal (see flowchart in figure 4-3, below):
  - User types a *string input* and puts it on the *blackboard system*.
  - The *Parser* parses the input into words and puts it on the list on the blackboard system.
  - The *Lexical Analysis layer* normalizes the string input by eliminating incorrect or incomplete words or phrases and checking relations between words or phrases. This layer puts the result on the blackboard system.
  - The *Affective State Analysis layer* activates its two sub layers: *Emotive Lexicon Dictionary Parser* and *Emotive Labeled Memory Structure Extraction*. *Emotive Lexicon Dictionary Parser layer* identifies the emotive lexicons from the user's input and the reply sentence (after it has constructed the reply sentence). The *Emotive Labeled Memory Structure Extraction layer* extracts the label from the system's memory unit. Those results are put on the blackboard system.
  - The *Syntactic-Semantic Analysis layer* performs a pattern matching operation based on the user's string input pattern to add the user's affective information on the blackboard system. In this step, the system starts to work in parallel with the process to construct system's reply sentence. As a result, the *Syntactic-Semantic Analysis layer* performs pattern-matching operation to generate system's reply sentence and puts its candidate on the blackboard system.
  - The *Affective Attributing Analysis layer* activates its sub layer, the *Concern of the other Analysis layer*, to perform emotion-based reasoning to deliver current system's reaction affective state and put it on the blackboard system. This analysis based on stimulus response and directly related to the user's input string.
  - The *Intensity Analysis layer* processes the message and calculates the current system's affective 'thermometers' and puts the calculation result on the blackboard system.
  - The *Facial Selection* selects my\_Eliza's facial display and puts it on the blackboard.
  - The *Wrap Process* delivers and displays my\_Eliza's **stimulus-response facial display**.



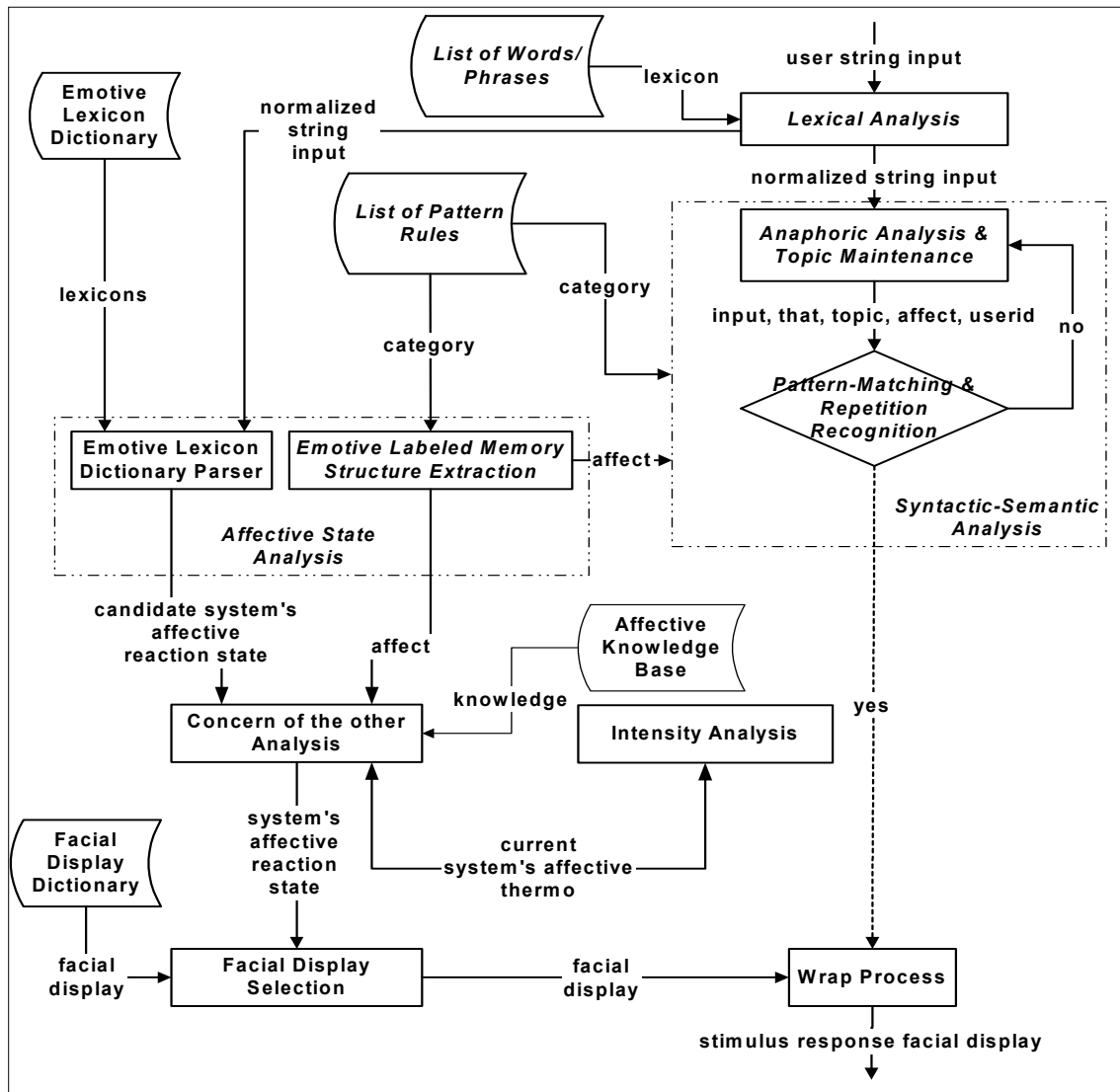


Figure 4- 3 Flowchart of My\_Eliza's stimulus-response for a user's string input

2. The system continues to process the candidate of reply sentence and generates a cognitive-processed facial display (see the flowchart in figure 4-4, below):
  - The *Pragmatic Analysis layer* processes the candidate reply sentence and puts the result on the blackboard system. This layer reviews the candidate of reply sentence whether it violates user's preference and/or system's goals, status and preferences, system's GSP. If it does, it will be sent back to the *Syntactic-Semantic Analysis layer* to get a new reply sentence.
  - The *Affective Attributing Analysis layer* activates its sub layer, the *Cognitive Reasoning layer*, to perform emotion-based reasoning to deliver current system's reaction affective state and put it on the blackboard system. This analysis is based on cognitive processing of system's GSP, system's reply sentence and user's affective state.
  - Again, the *Intensity Analysis layer* processes the message and calculates the current system's affective 'thermometer' level. This layer also puts the result of calculation on the blackboard system.
  - The *Facial Selection* selects my\_Eliza's facial display and puts it on the blackboard.
  - The *Wrap Process* delivers and displays my\_Eliza's **reply sentence** and **cognitive-processing result facial display**.

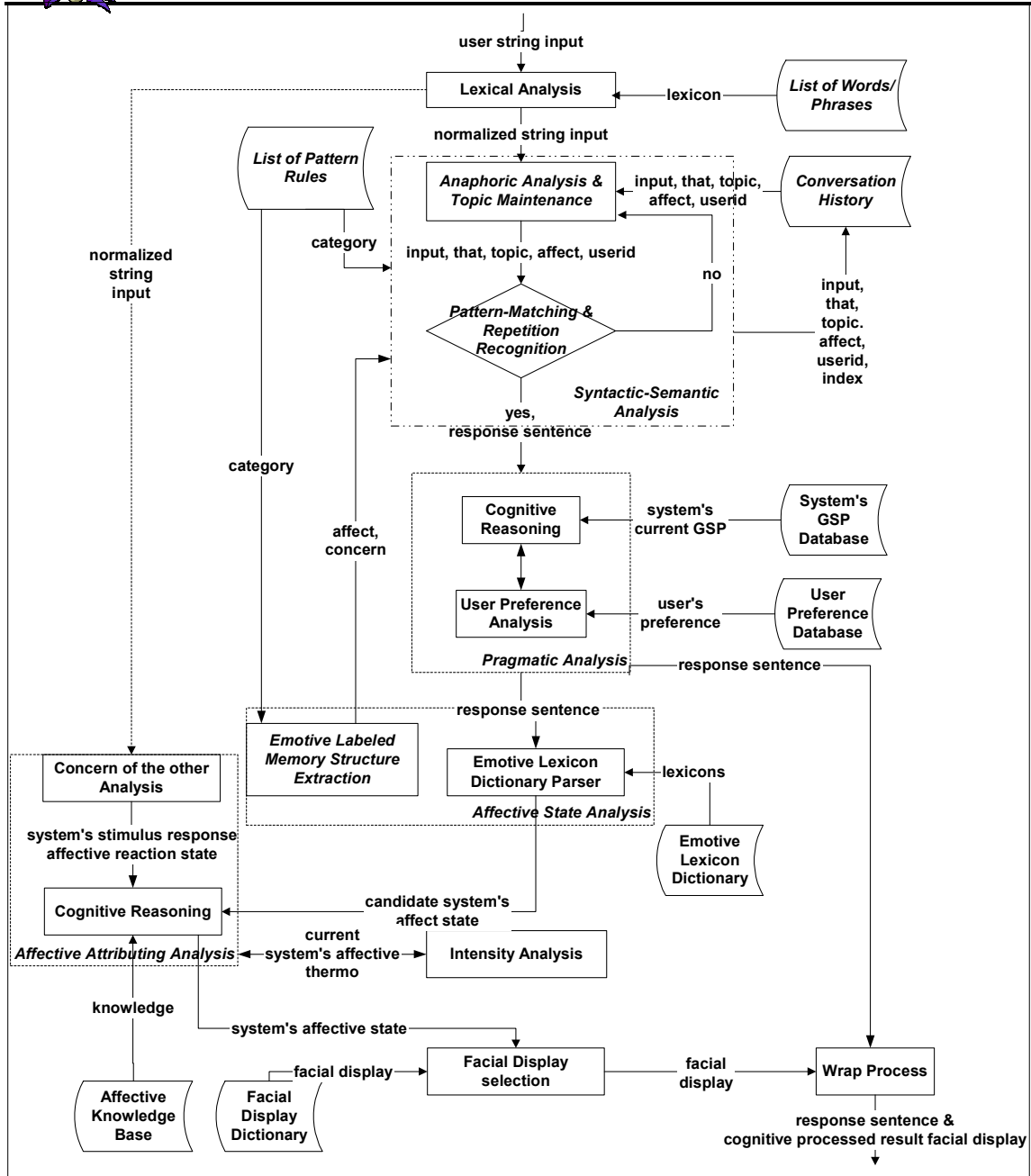


Figure 4- 4 Flowchart of My\_Eliza's cognitive processing for user's string input

In the next chapter we will discuss about all the knowledge that is used by the system. The NLP-layers and FDC-layers will be discussed in detail sequentially in section 4.3 and 4.4.

## 4.2 Knowledge Extraction

### 4.2.1 Databases in Natural Language Processing Layers

Most of the databases of my\_Eliza used in these layers are based on a script of Richard Wallace's AIML (Artificial Intelligence Markup Language). As mentioned in section 2.2, AIML is extended-XML (Extensible Markup Language) script specification for programming a memory



structure for a QA system. The adaptation of AIML in my\_Eliza is discussed step by step in chapter 5. There are four databases that are used in the reply construction process:

1. List of words/phrases
2. List of pattern rules (system's memory structure)
3. User Personal Database
4. System's GSP database

One by one will be deliberately discussed on four subsections below.

#### 4.2.1.1 List of Words/Phrases

A List of words or phrases is used in the *Lexical Analysis layer* to eliminate incorrect or incomplete words or phrases and checking the relation between words or phrases (figure 4-5 below). The most important units of AIML in this list are:

- **<substitutions>**: the tag that begins and ends list of words/phrases.
- **<substitute find="name" replace="name">**: the tag that tells the AIML interpreter to replace any word(s) in *find-value* by word(s) in *replace-value*.

```
<substitutions>
  <input>
    <substitute find=":-)" replace=" smile "/>
    <substitute find="wanna" replace="want to"/>
    <substitute find="becouse" replace="because"/>
    <substitute find=",and" replace="."/>
    <substitute find="mr." replace="mister"/>
    ...
  <input/>
  <gender>
    <substitute find="to him" replace="to her"/>
    <substitute find="with her" replace="with him"/>
    ...
  <gender/>
  <person>
    <substitute find="I was" replace="she or he was"/>
    <substitute find="she is" replace="I am"/>
    ...
  <person/>
  <person2>
    <substitute find="I am" replace="you are"/>
    <substitute find="you were" replace="I was"/>
    ...
  <person2/>
</substitutions>
```

Figure 4- 5 Example of list of words/phrases

- **<input>**: the tag that tells the AIML interpreter to substitute the contents of a user input. It consists of:
  - *Spelling checker*, for example “becouse” will be substituted by “because”.
  - *List of known codes*, for example smiley’s or emoticons :-) is ☺, and :-( is ☹, etc.
  - *Synonym*, the list contains words or expressions of the same language that have the same or nearly the same meaning in some or all senses. For example: the synonym list for member of a family are parent, mother, father, son, daughter, brother, sister, grandparent, etc.
  - *List of unknown codes/words*, the list contains codes or words that must be eliminated or normalized before the input is processed by the syntactic-semantic analysis layer, for example: code “%” will be replaced by a blank. This list also contains substitution of ending characters/words that will not be identified as sentence enders, for example in a compound sentence using “but”, “and” or “,”. Each sentence of the compound input will be split and processed one by one by the syntactic-semantic analysis layer.



- *Abbreviation list*, the list contains abbreviations and extensions of it. For example: “Mr.” abbreviation of “Mister”.
- *List of slang words form*, the list contains converted words that must be replaced by the English correct form. For example “wanna” substituted by “want to”, “dont” substituted by “do not” and so on.
- **<gender>**: the tag that tells the AIML interpreter to replace female-gendered words in the result of processing the contents of the `gender` element with the grammatically-corresponding male-gendered words and vice versa male-gendered words to male-gendered words (for example “her” substitute into “him” and vice versa). This substitution is used due to the fact that my\_Eliza uses the English language.
- **<person>**: the tag that tells the AIML interpreter during processing the content of the `person` element to replace words with first-person aspect with words with the grammatically-corresponding third-person aspect and vice versa third-person aspect to first-person aspect (for example “I” substitute into “she or he” and vice versa).
- **<person2>**: the tag that tells the AIML interpreter during processing the content of the `person` element to replace words with first-person aspect with words with the grammatically-corresponding second-person aspect and vice versa second-person aspect to first-person aspect (for example “I” substitute into “you” and vice versa).

The best heuristic process to collect this list is using a lot of text/written conversation, from stories and to seek common knowledge about the English language.

#### 4.2.1.2 List of Pattern Rules (System’s Memory Structure)

This list is the memory structure of my\_Eliza controlled by the AIML script. This list contains the rules that are used to recognize the input and to construct a reply sentence. This list is also used to recognized user’s and system’s affective situation type. The most important units of AIML [BUS01] are:

- **<aiml>** : the tag that begins and ends an AIML document.
- **<category>**: the tag that marks a "unit of knowledge" in a system’s knowledge base.
- **<pattern>** : the tag that is used to contain a simple pattern that matches what a user may type
- **<topic>** : the tag that is used to represent current conversation topic. Using `<topic name="XXX">` where “XXX” is a topic name, a memory unit is classified into one particular topic. We also can use `<settopic>` tag to tell the interpreter that current conversation is about one particular topic. `<gettopic>` tag is a tag to access current topic (see appendix A.2 for more information about AIML tags).
- **<that>** : the tag that refers to system’s previous reply. Using `<that index="nx, ny"/>` tag, we can refer to any system’s reply on history.
- **<template>**: the tag that contains the response to a user input. There is considerable freedom of expression in the construction of response templates.

AIML broadly breaks down into two parts:

- *Pattern side AIML expression* – PSAE that can appear with or in the tag `<pattern>`, `<that>`, and `<topic>`.
- *Template side AIML expression* – TSAE that appear inside the tag `<template>`. Inside the template are comprised of ordinary text, optionally marked up with all the other AIML tags.

In addition, there are two forms of the AIML `<that>` tag: a paired form `<that>...</that>` appearing in a category and an atomic form `<that/>` always appearing in a template. Two



categories in figure 4-6 show how to implement those tags. The first fragment seems not natural, therefore my\_Eliza uses the second category with tag `<that>...</that>`, which produces a more natural dialog fragment.

```
<category>
  <pattern>WHY</pattern>
  <that>*</that>
  <template><that\>?Why?</template>
</category>
<category>
  <pattern>WHY</pattern>
  <that>DO NOT ASK ME ANY MORE QUESTIONS PLEASE</that>
  <template>Because I would rather talk about you</template>
</category>
```

The first category with tag `<that/>` produces the following dialog fragment:

```
My_Eliza: Do not ask me any more questions please!
User: WHY?
My_Eliza: Do not ask me any more questions please? Why?
```

The second category with tag `<that>...</that>` produces the following dialog fragment:

```
My_Eliza: Do not ask me any more questions please!
User: WHY?
My_Eliza: Because I would rather talk about you.
```

Figure 4- 6 Two forms of the AIML `<that>` tag

There are also more additional tags often found in AIML files (see appendix A-2 or in [BUS01]). In our prototype, we use the list of pattern rules of Weizenbaum's Eliza. Those rules are converted into AIML form (see chapter 5 for detailed explanation). Eliza's *decomposition rules* are converted into my\_Eliza's *patterns* and Eliza's *reassemble rules* are converted into my\_Eliza's *templates*. Each Eliza's *keyword* is converted into one category. Using other important units of AIML could develop this list of pattern rules in order to increase the volume of my\_Eliza's memory structure. This effort is necessary to get a more naturalistic conversation between my\_Eliza and the user since our prototype is designed as a "proof of concept".

### Emotive Labeled Memory Structure

As described in section 2.2, the syntactic and semantic analysis that uses the pattern matching operation approach does not store the affective information in the memory structure. Therefore, the system still needs human intervention to introduce emotion to the system. Tackling this problem, the basic idea is to label the memory structure with affective information similar to the work of Pelachaud et. al [PEL98] (see section 2.5). For this purpose we extend the AIML form with two new tags, such as:

- **`<affect name="XXX"></affect>`:** labels the user's affective situation, XXX is the value of the current situation type which is guessed from the pattern of user's string input. This tag appears in PSAE. We can also use `<setaffect>...</setaffect>` in TSAE as the inheritance of `<affect>` tag to set current user's affective situation type. As an additional tag, there is `<getaffect>` tag to get the information about the current user's affective situation type.
- **`<setconcern>XXX</setconcern>`:** labels the system's affective reaction situation, XXX is the value of the situation type to convey system's reply sentence. This tag appears only in TSAE. Also as an additional tag, there is `<getconcern>` tag to get the information about the current system's affective situation type.

There are four-possibility emotive labels for both tags, such as:

- 1) A positive situation is labeled with "+"
- 2) A negative situation is labeled with "-"
- 3) A joke is labeled with "#"
- 4) A neutral situation is labeled with "\*"



We use those situation types to classify the semantic meaning of the sentence. Based on the emotion classification of Reddy [RED01], that classifies emotions into “pleasant” or “unpleasant”. We translate a pleasant situation as positive situation type. Happiness and empathy are desirable conditions that belong to this type of situation. For unpleasant situation, we translate them into a negative situation type, where sadness, anger, disgust and fear are the conditions that belong to this type of situation. We add two situation types: a neutral and a joke. A neutral situation type is chosen if there is not a dominant emotion type in the sentence. The joke situation type is meant to distinguish between happiness that makes people smiling with the situation that makes people laughing.

The emotion labels are manually set to each of my\_Eliza’s memory unit (category). This work needs high human consistency. The best heuristic method to label the situation type to every category is by looking for emotive lexicon(s) in the text. If we find positive emotive lexicon (that shows positive emotion) we label that category (affect or concern tag) with “+” and in the other way around, if we find a negative emotive lexicon, we label that category with “-“. Ekman [EKM75] and Mehrebian in [KIN97] identified another heuristic method, such as:

1. **Recognize immediate cues that communicate liking-pleasure and disliking-hate from the text**, for example with words: “love”, “like”, “hate”, and etc. If from a category we find liking-pleasure cues we label that category with “+” and if we find disliking-hate cues we label that category with “-“.
2. **Recognize arousal cues that communicate complement-encourage and anger-discourage-fear**, for example for complement-encourage cues we may find phrases “you are right”, “that’s good” and etc. If from a category we find those kind of cues we label that category with “+”. For anger-discourage cues we may find phrases “shut up”, “never mind”, “don’t ... !” or usually with negative emotion appreciated language. If from a category we find those kind of cues we label that category with “-“.
3. **Recognize dominance cues that communicate pride and insulting**, for example “I am the best psychoanalyst in the world”, “you are weird”, and so on. Due to the role of my\_Eliza as psychoanalyst she has to be wise and depends on the context if we find those kind of cues we label that category with “#”.
4. **Look for a cluster**, there is relation between affect tag (user’s affective situation type) and concern tag (system’s affective situation type) and intention. If the affect tag shows a negative situation then my\_Eliza intends to be nicer, encourage and reflecting to the user’s feeling. If the affect tag shows a positive situation or the user makes a joke, my\_Eliza intends to be relaxed and replies the user’s jokes.
5. **Consider past experience**, using the <that> tag we will know the system’s previous reply sentence between my\_Eliza and the user. If the previous conversation was negative situation then my\_Eliza intends to be nicer, encourage and reflect to the user’s feeling. If the previous conversation intended to positive situation or the user made a joke then my\_Eliza intends to be relax and reply the user’s jokes.

In addition, if my\_Eliza shows her empathy or her sorry-for the user’s feeling we label the concern tag with “+”, similar to the positive situation of my\_Eliza. If the pattern or the template shows that it can be said in any situation, we label it with “\*”. Figure 4-7 shows an example of two memory units with the same topic (two category bounded in one topic name= “SECRET”) and the same reference to the system’s previous reply (showed by <that>...</that> tag), however they have a different label of user’s affective state. That difference influences the system’s reply sentence and its affective state (shown by various templates with different values of <setconcern>...</setconcern> tag). In the first category, <setaffect>...</setaffect> tag is used to change current user’s situation type. From the example, we know that if this category is chosen, my\_Eliza concludes that the user is in a negative situation and hopefully if she says something nice then the user will change to a positive situation.



```
<aiml>
<topic name="SECRET">
<category>
<effect name="-">
<that>YOUR SECRET *</that>
<pattern>YOU WILL NOT * WONT YOU</pattern>
<template><random>
<li><think><setconcern></setconcern></think>But I did, some time ago.</li>
<li><think><setconcern>+</setconcern></think>
    You're probably right, I won't, but why does it concern you?</li>
<li><think><setconcern>+</setconcern><setaffect>+</setaffect></think>
    Don't worry, I won't</li>
...
</random></template>
</effect>
</category>

<category>
<effect name="+">
<that>YOUR SECRET *</that>
<pattern>YOU WILL NOT * WONT YOU</pattern>
<template><random>
<li><think><setconcern>#</setconcern></think>But I did, some time ago.</li>
<li><think><setconcern>#</setconcern></think>
    Are you some kind of predictor or what?</li>
<li><think><setconcern>#</setconcern></think>
    I sure will, and you'll be a witness for that.</li>
<li><think><setconcern>#</setconcern></think>
    <get name="name"/>, do you really think I won't?</li>
...
</random></template>
</effect>
</category>
</topic>
</aiml>
```

Figure 4- 7 Example of two memory units with the same topic in my\_Eliza's list of pattern rules

### Three Types of Categories

Given only the <pattern> and <template> tags, there are three general types of categories: **atomic**, **default** and **recursive**, (those three types have some overlap because atomic and default refer to <pattern> and recursive refers to the <template>):

1. **Atomic** categories are those with atomic patterns with no wildcard "\*" or "\_" symbol. In the figure 4-8, the pattern shown will match *only* the exact phrase "what are you" (capitalization is ignored). In any case, if this category is called, it will produce the response "I am the latest result in artificial intelligence..." shown above. Using the <think> tag, the system will set the "topic" in its memory to "Me" but hide the result from the user. This allows any categories elsewhere with an explicit "topic" value of "ME" to match better than categories with the same patterns that are not given an explicit topic.

```
<category><affect name="*">
  <pattern>WHAT ARE YOU</pattern>
  <template>
    <think><setconcern>+</setconcern><set name="topic">Me</set></think>
    I am the latest result in artificial intelligence,
    which can reproduce the capabilities of the human brain
    with greater speed and accuracy.
  </template></affect>
</category>
```

Figure 4- 8 Example of atomic category





2. **Default** category derived from the fact that its pattern has a wildcard “\*” or “\_”. The ultimate category is the one with `<pattern>*</pattern>`, which matches any input. The more common default categories have patterns combining a few words and a wild card (see in figure 4-9).

```
<category><affect name="-">
  <pattern>I NEED HELP *</pattern>
  <template><think><setconcern>+</setconcern></think>
    Can you ask for help in the form of a question?
  </template></affect>
</category>
```

Figure 4- 9 Example of default category

The category in figure 4-9 responds to a variety of inputs, such as: “I need help on cooking” or “I need help debugging my program”.

3. **Recursive** categories are those that “map” inputs to other inputs, either to simplify the language or to identify synonymous patterns. Synonymous input have the same response, which is accomplished with the recursive `<srai>` tag, for example “GOODBYE”, “BYE”, “DAG”, “CYA” and soon, are mapped to the same output for “GOODBYE” (see the example (1) in figure 4-10).

```
(1) <category><affect name="+">
      <pattern>DAG</pattern>
      <template><srai>GOODBYE</srai></template>
    </category>
(2) <category><affect name="*">
      <pattern>DO YOU KNOW WHAT * IS</pattern>
      <template><srai>WHAT IS <star/></srai></template>
    </category>
(3) <category>
      <pattern>HELLO * </pattern>
      <template><srai>HELLO</srai></sr></template>
    </category>
```

Figure 4- 10 Example of recursive category

Simplification or reduction of complex input pattern (namely in term *symbolic reduction*) is another common application for this category. For example, the question “What is X” could be asked by “Do you know what X is”, “Tell me about X”, “Describe X”, and etc, the `<srai>` function maps all these forms to the base form (see the example (2) in figure 4-10). The `<star/>` tag substitutes the value matched by “\*”, before the recursive call to `<srai>`. This category transforms (for example) “Do you know what a circle is?” to “WHAT IS A CIRCLE”, and then finds the best match for the transformed input. Another fairly common application of recursive is called “parsing” (actually AIML does not really parse natural language) or “partitioning”, because these AIML categories break down an input into two or more parts, and then combine their responses back together. If an input sentence starts with “Hello ...”, it does not matter what comes after the first word, in the sense that the system can respond to “Hello” and whatever is after “...” independently. “Hello my name is Carl” and “Hello how are you” are quite different, but they show how the input can be broken into two parts (which assumes there is an ATOMIC category of `<pattern>HELLO</pattern>`). The category below accomplishes the input partitioning by responding “HELLO” with `<srai>HELLO</srai>` and to whatever matches “\*” with `</sr>` (see example (3) in figure 4-10). The response is the result of the two partial responses appended together.

The categories written in an AIML file do not need to be in alphabetical order by pattern because the program will maintain that order internally. The wildcard character “\*” comes before “A” in alphabetical order, for example, the “WHAT ” pattern is more general than “WHAT IS ”. The



default pattern “<pattern>\*</pattern>” is the first in alphabetical order and the most general pattern. “\_” form will comes after “Z” in alphabetical order.

### 4.2.1.3 User Personal Database

For each user id, there is a user’s preferences database stored in a log file. This log file is established gradually in the conversation process as a learning process of the system to know the user better. In figure 4-11 there is an example of the category used to collect user’s information about their favorite movie.

```
<topic name="MOVIE">
<category><effect name="+">
  <that>* FAVORITE MOVIE</that>
  <pattern>*</pattern>
  <template>
    <think><setconcern>+</setconcern></think>
    <set_property name="favorite_movie"><star\></set_property>? That's mine too?
  </template></affect>
</category>
</topic>
```

which can be used on the fragment below:

```
My_Eliza: What is your favorite movie?
User      : Indiana Jones.
My_Eliza: Indiana Jones? That's mine too.
```

Figure 4- 11 Example of category for storing information during conversation

That information will be stored in user personal database distinguished by user id. Figure 4-12 below is an example of a user personal database. The most important units of AIML in this database are:

- **<users>**: the tag that begins and ends user’s preference database.
- **<user id="name" enable="true/false">**: the tag that starts one user’s preference database known by the id of the user and end with the AIML **<user/>** tag.
- **<property name="name" value="name">**: the tag that contains user’s personal database with *name* with its *value*.

```
<users>
  <user id="name" enable="true">
    <property name="favorite_book" value="Harry Potter"/>
    <property name="favorite_movie" value="Indiana Jones"/>
    <property name="friend" value="dave, A.L.I.C.E, eliza"/>
    <property name="mother" value="alive"/>
    <property name="father" value="died"/>
    <property name="hobby" value="reading"/>
    . . .
  </user/>
  . . .
</users/>
```

Figure 4- 12 Example of user’s preferences, which are stored by my\_Eliza per user id

### 4.2.1.4 System’s GSP Database

My\_Eliza also stores its current own goal, status and preferences in a database. The goal and the status of the system always change during conversation. The explanation about what the goals of the system are, is presented in section 4.3. The status of the system is actually the emotional status, which is indicated by its affective reaction emotion type (resulted by the *Affective Attributing layer*, see section 4.4). Those values are stored to be used in reviewing the reply sentence



whether it is consistent with system's current goal and status (processed by the *pragmatic analysis layer*, see section 4.3).

The preferences database is stored in a log file and will be initiated when the system is executed the first time. Some of these preferences can influence the system's liking or disliking emotion state, for those she hates she will show disliking and for those of her favorite things she will show liking (figure 4-13). The most important units of AIML in this database are:

- **<bots>**: the tag that begins and ends my\_Eliza's preference database.
- **<bot id="name" enable="true/false">**: the tag that begins on my\_Eliza's preference database known by the id of the bot (reserved for future development whenever the system can provide more than one bot id(s)) and end with the AIML **<bot/>** tag.
- **<property name="name" value="name">**: the tag that contains my\_Eliza's personal database with *name* and its *value*.

```
<bots>
  <bot id="my_Eliza_ver_1" enable="true">
    <property name="name" value="my_Eliza"/>
    <property name="birthday" value="August 22, 2002"/>
    <property name="birthplace" value="Delft"/>
    <property name="favorite_color" value="yellow"/>
    <property name="favorite_movie" value="your life story"/>
    <property name="hate_attitude" value="lie, distress"/>
    <property name="hate_food" value="list of bad words"/>
    . . .
  <bot/>
  . . .
</bots>
```

Figure 4- 13 Example of my\_Eliza's preferences database

This database is also used to answer questions about my\_Eliza's personality from the user. In figure 4-14, the user asks my\_Eliza's birthday. The system cannot answer it perfectly if it did not store the information before. Using system's GSP database in figure 4-13, **<bot name="XXX">** tag can deliver the value of the property of the system, where "XXX" is the name of the property. The best heuristic process to build this database is by experiencing conversation frequently in chat-rooms.

```
<topic name="BIRTHDAY">
<category><effect name="+">
  <that>*</that>
  <pattern>WHEN IS YOUR *</pattern>
  <template>
    <think><setconcern>+</setconcern></think><bot name="birthday">. When is yours?
  </template></affect>
</category>
</topic>
```

which can be used on the fragment below:

```
User      : When is your birthday?
My_Eliza: August 22, 2002. When is yours?
```

Figure 4- 14 Example of the use of system's GSP database in a dialog

## 4.2.2 Databases and Knowledge Base in Facial Display Construction Layer

There are three knowledge bases that are used in the reply construction process and in detecting user's affective state:

1. Emotive lexicon dictionary



2. Affective knowledge-base
3. Facial display dictionary

One by one will be deliberately discussed in three subsections below.

#### 4.2.2.1 Emotive Lexicon Dictionary

The emotive lexicon dictionary is used to recognize the emotive lexicon in a sentence, which may the user or my\_Eliza use to show their affective state during conversation (see section 2.4). Besides labeling the memory structure with emotive information, we use also this dictionary to extract emotion-eliciting factor information in the text prompt in the conversation both of the user and the system. The basic idea is to classify and label the text prompt based on the emotive lexicon in it. This idea is based on the work of Elliott [ELL93]. Since the first prototype is dedicated as a “proof of concept”, only six universal emotion types (Ekman’s) will be used in facial displays and emotive lexicons classification instead of twenty-four OCC’s theory emotion types.

Table 4- 2 Corresponding twenty-four OCC’s theory emotion types into six clusters emotion types

Cluster: Universal Emotion Types	OCC’s Theory Emotion Types	
	Specification	Emotion Types
Happiness	Appraisal of a situation as an event	<b>Joy</b> : pleased about an event
	Presumed value of a situation as an event affecting another	<b>Happy-for</b> : pleased about an event desirable for another <b>Gloating</b> : pleased about an event undesirable for another
	Appraisal of a situation as confirming or disconfirming an expectation	<b>Satisfaction</b> : pleased about a confirmed desirable event <b>Relief</b> : pleased about a disconfirmed undesirable event
	Appraisal of a situation as an accountable act of some agent	<b>Pride</b> : approving of one’s own act <b>Admiration</b> : approving of another’s act
	Appraisal of a situation as containing an attractive or unattractive object	<b>Liking</b> : finding an object appealing
	Compound emotions	<b>Gratitude</b> : admiration + joy <b>Gratification</b> : pride + joy
	Compound emotion extensions	<b>Love</b> : admiration + liking
Sadness	Appraisal of a situation as an event	<b>Distress</b> : displeased about an event
	Presumed value of a situation as an event affecting another	<b>Resentment</b> : displeased about an event desirable for another <b>Sorry-for</b> : displeased about an event undesirable for another
	Appraisal of a situation as confirming or disconfirming an expectation	<b>Disappointment</b> : displeased about a disconfirmed desirable event
	Appraisal of a situation as an accountable act of some agent	<b>Shame</b> : disapproving of one’s own act
	Compound emotions	<b>Remorse</b> : shame + distress
Disgust	Appraisal of a situation as containing an attractive or unattractive object	<b>Disliking</b> : finding an object unappealing
	Compound emotion extensions	<b>Hate</b> : reproach + disliking
Surprise	Appraisal of a situation as a prospective event	<b>Hope</b> : pleased about a prospective desirable event <b>Fear</b> : displeased about a prospective undesirable event
Fear	Appraisal of a situation as confirming or disconfirming an expectation	<b>Fears-confirmed</b> : displeased about a confirmed undesirable event
Anger	Appraisal of a situation as an accountable act of some agent	<b>Reproach</b> : disapproving of another’s act
	Compound emotions	<b>Anger</b> : reproach + distress



Using twenty-four OCC's theory emotion types, then we have to add the corresponding facial displays and emotive lexicons classifications to twenty-four emotion types. To supply the situation, the twenty-four emotion types are reduced to six clusters corresponding to Ekman's universal emotion types, see table 4-2 above. We take assumption the situation that is affected by another or some agents means as a situation that is affected by the user during conversation.

The most important units of AIML in this dictionary are (figure 4-15 below):

- **<emotions>**: the tag that begins and ends emotive lexicon dictionary.
- **<emotion-dictionary name="name" value="[name1: intensity1], [name2: intensity2], [name3: intensity3], ...">**: the tags contains emotion type *name* with *value* are the set of emotive lexicons and their intensity. The intensity is an integer value [1..3], which 1 means low intensity, 2 means medium intensity and 3 means high intensity.
- **<emotion-comparative name="name" value="name1, name2, name3, ...">**: the tags contain the words to show intensity of emotions.

```
<emotions>
  <emotion-dictionary name="happiness" value="[well:1], [lucky:1], [fortunate:1],
    [happy:1], [joy:1], [joyful:2], [delighted: 3], [satisfied:1], ..." >
  <emotion-dictionary name="sadness" value="[sad:1], [blue:2], [down:2],
    [unhappy:1], [sick:2], [mourning:3], [not well:2], [depress:3], ..." >
  <emotion-dictionary name="anger" value="[angry:1], [huh:1], [upset:2],
    [piss:2], [piss off:3], [not sad: 1], [shut up:3], [shit:3], ..." >
  <emotion-dictionary name="disgust" value="[not like:1], [disgust:1],
    [disgusting:2], [outrage:3], [make me sick:1], [turn my stomach:1], ..." >
  <emotion-dictionary name="surprise" value="[surprise:1], [no way:2], [why:1],
    [what is wrong:1], [hah:2], ..." >
  <emotion-dictionary name="fear" value="[afraid:1], [panic:2], [horror:3],
    [hii:1], [please:1], [terrified:1], ..." >

  <emotion-comparative name="low" value="bit, little, quite, few ..." >
  <emotion-comparative name="high" value="huge, very, big, ..." >
</emotions>
```

Figure 4- 15 Emotive lexicons dictionary

We collect every lexicon from spoken words or phrases by considering how human expresses emotion using words with an emotional loading. We also collect negation of those lexicons and categorize them into one of the six universal emotion types, such as: "not well" is categorized into sadness with low intensity, "not bad" is categorized into happiness with low intensity, etc. Obviously, there are gray areas as well and multiple meaning of words that would place them in different emotion categories. The best heuristic method to collect the lexicon according to Elliot is using a thesaurus, dictionary, and stories. In addition, we also analyze multiple-person brainstorming sessions. The dictionary also contains comparative lexicon to show emotion intensity, such as high comparative (very, huge, big) and small comparative (bit, quite, small, little, few).

#### 4.2.2.2 Affective Knowledge-Base

This knowledge base controls the whole emotion recognition from a dialog state in the IF-THEN form. The IF part is the condition of all combinations of emotion-eliciting variables and the THEN part is one of twenty-six emotion types in table 4-1. Note, this is the only process in FDC-layers that delivers a result based on the OCC's theory emotion type. We use this classification theory instead of Ekman's because two reasons: (1) it has more number of emotion types and (2) it is classified based on human people's view about their goal, emotional status and preferences. Therefore, the knowledge base is expected to have a wider range of application.



There are two kinds of knowledge bases:

**1. Stimulus response knowledge base**

This knowledge base is used to classify the system's reaction affective state based on stimulus response to user's input string. The emotion-eliciting variables that activates such emotion type, are:

- *User-affective state candidate* – the result of emotive lexicon dictionary parser parses user's string input.
- *Current user's situation type* – the value of emotive label in <affect> tag.
- *Current system's affective state* – the emotion type with highest degree of the six "thermometers" of intensity analysis layer.

**2. Cognitive process knowledge base**

This knowledge base is used to classify the system's reaction affective state based on cognitive-processing of previous conversation (previous the user's affective state and previous the system's reaction affective state), the user's preference, the current user's affective state, and the system's goal, state and preference (GPS). The emotion-eliciting variables that activates such emotion type, are:

- *System-affective state candidate* – the result of emotive lexicon dictionary parser parses user's string input.
- *Current user's situation type and current system's situation type* – the values of emotive labels in <affect> tag and <concern> tag.
- *Last user's situation type and last system's situation type* – the values of emotive labels of the previous reply sentence in <affect> tag and <concern> tag.
- *Last system's reaction affective state* - from previous dialog state.
- *Thermometer* - one of six emotion types with the highest degree of the six thermometers.
- *User's preferences* – based on user's personal database whether the user likes or dislikes the conversation content.
- *System's preferences* – based on system's GSP database whether the system likes or dislikes the conversation content.
- *System's goal* – based on system's GSP database whether the goal of the system is appealing or not appealing.

The adaptation of this knowledge base in my\_Eliza system is clearly explained in section 6.2.4.

### 4.2.2.3 Facial Display Dictionary

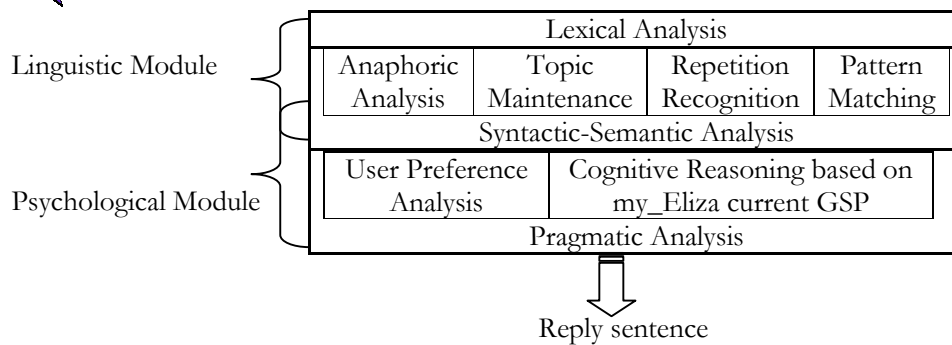
This dictionary contains various facial display files distinguished by Ekman's seven universal emotion types (neutrality, happiness, sadness, fear, surprise, disgust and anger) plus one additional emotion type: uncertainty. Each emotion type has three levels of intensity: low, medium and high except for neutrality. In our prototype, the facial display file is stored as a nonverbal picture to represent those universal emotion types.

## 4.3 Natural Language Processing Layers

The NLP layer accepts a user's string input to construct a reply sentence. This layer is composed of several layers (figure 4-16). The NLP layers are divided into two big parts:

- **Linguistic Module**, to improve users' belief.
- **Psychological Module**, to improve the sense of reality of my\_Eliza behavior. The purpose of this part is to allow my\_Eliza to perceive the environment by understanding the user preferences and cognitive reasoning based on my\_Eliza's goals, state and preferences (GSP).





### Figure 4- 16 Natural Language Processing Layers

## Lexical Analysis Layer

This layer uses a list of words or phrases. This layer has the task to normalize the input by eliminating incorrect or incomplete words or phrases and checking the relation between words or phrases. This layer receives a string and parses the string word by word (see normalization algorithm in figure 4-17). If it finds the token on the list of words, the system will replace it with the correlated substitution word.

```

Get the string; Set token as first word in the string
While the token is not last words of the sentence
    If this token is found in the list of words
        Replace token with get_fromListOfWords(token)
    Next token
//end while
return the string

```

**Figure 4- 17 Algorithm for the normalization of a string**

## Syntactic Analysis and Semantic Analysis

Like the original Weizenbaum's Eliza, there is not a clear separation between Syntactic Analysis and Semantic Analysis in my\_Eliza reply construction process. My\_Eliza performs a shallow syntactic and semantic analysis of a user's string input based on pattern matching. In this layer there are four operations, which will be executed to process user's string input:

1. **Anaphoric analysis**, this analysis has to guarantee that my\_Eliza responds consistently on prompts by referring on the precedent conversation content.
2. **Topic maintenance**, this analysis guarantees that my\_Eliza be focused or maintained properly on significant issues during a dialog.
3. **Repetition recognition**, this analysis uses dialog history to make sure that the dialog never gets in a loop.
4. **Pattern matching**, it can be said that this process is the lead of this analysis layer. It controls all three processes above to get the semantic result of user's input. The result is a candidate of reply sentence that will be used by my\_Eliza to reply user's string input.

The first two operations are executed simply because the structure of my\_Eliza's memory structure and the pattern matching process. <that></that> tag makes sure that my\_Eliza reply is always based on what has been said before and <topic></topic> tag makes sure that my\_Eliza always replies according to the current topic. Due to the addition of the <affect> tag, my\_Eliza's pattern-matching operation is slightly different with A.L.I.C.E's (see in section A.3). By comparing the input with the pattern in a fixed order, the algorithm in figure 4-18 below ensures that the most specific pattern matches first, basically it finds the longest pattern matching in the input.





```
ATOMIC CATEGORY with the same TOPIC and THAT and AFFECT
ATOMIC CATEGORY with the same TOPIC and AFFECT
ATOMIC CATEGORY with the same AFFECT
DEFAULT CATEGORY with the same TOPIC and THAT and AFFECT
DEFAULT CATEGORY with the same TOPIC and AFFECT
DEFAULT CATEGORY with the same AFFECT
ATOMIC CATEGORY with the same THAT
ATOMIC CATEGORY
DEFAULT CATEGORY with the same THAT
DEFAULT CATEGORY
```

Figure 4- 18 My\_Eliza's pattern matching algorithm

The atomic category will always take precedence over any other type of category. If there are two identical patterns of atomic categories but one has the same THAT, then the THAT category will take precedence over the other atomic categories, if the THAT matches my\_Eliza's previous response. If neither of the statements above are true, then a reduction category that matches a part of the pattern will give its response, and finally if none of them matches, then the ultimate category with `<pattern>*</pattern>` (which matches any input) will take it over. Any category that is contained within a TOPIC section will be searched first if the current setting of TOPIC matches a TOPIC section. Moreover, any category that has the same AFFECT included will be searched first if the current setting of AFFECT matches the category's affect.

Repetition recognition is guaranteed by using semi random selection of one of the templates in the selected category. The system will not use the same template to reply to the user if that template has been used unless there is not any. The system logs all conversation and can be reached by `<that index="nx,ny">` (see algorithm on figure 4-19).

```
Found = true;
While there are still templates and found in history log do
    While not found in history log do
        If the same topic and that[nx,ny] = template then found
    If template empty then select any template
```

Figure 4- 19 Repetition recognition algorithm

## Pragmatic Analysis

One of the important aspects of natural language is pragmatic analysis. In this layer there are two operations to post process the reply candidates from the semantic analysis layer, such as:

### 1. *User preferences analysis.*

Actually, this operation is also a part of semantic analysis because it is stored in my\_Eliza's memory structure. As mentioned in section 4.2.1.3 above, during conversation my\_Eliza also stores user information in the User Personal Database. The use of that database is the purpose of this analysis. For example, from (in figure 4-20 below) this database my\_Eliza knows the information that the father of the user has already died.

The tag `<condition>` will refer to the user personal database to seek whether the property name is defined and then get a value. The system will use the template if the condition is true. This analysis prevents the blunder that may occur if the system at that time did not store and use the fact that the father was dead. My\_Eliza has changed and improved her knowledge and behavior on the basis of her past experience.



Example the content of user's personal database:

```
. . .
<property name="favorite_movie" value="Indiana Jones"/>
<property name="hate_animal" value="cat"/>
<property name="friend" value="dave, A.L.I.C.E, eliza"/>
<property name="mother" value="alive"/>
<property name="father" value="died"/>
<property name="hobby" value="reading"/>
. . .
```

That information can be used in the category below:

```
<topic name="FATHER">
<category><effect name="-"> <that>TELL ME *</that>
  <pattern>HE WAS *</pattern>
  <template>
    <condition>
      <li name="father" value="alive"><think><setconcern>+</setconcern><think>
        Is he now?</li>
      <li name="father" value="died"><think><setconcern>+</setconcern><think>
        I am sorry that he has already passed away. Were you close to each
        other?</li>
    </condition>
  </template></affect>
</category> </topic>
```

Example fragment:

```
User      : When my father died my mother was sick for a long time.
My_Eliza: Tell me about him.
User      : He was a successful a write.
My_Eliza: I am sorry that he has already passed away. Were you close to each other?
User      : Yes, when I was a little boy.
```

Figure 4- 20 Example of a User Personal Database unit and how to use it in a conversation

## 2. Cognitive Reasoning using system's goal, states and preferences.

This operation is executed to guarantee the reply sentence is consistent to system's goal, status and preference. This layer uses system's GSP database (see figure 4-21 below). Similar to Weizenbaum's Eliza, we give my\_Eliza the role as a psychoanalyst that as aim to keep the conversation going. To pursue this role, my\_Eliza has several goals during conversation with a user, such as:

- **Answering questions** – if the user asks something, my\_Eliza's goal is to answer it, for example patterns like: "HOW ARE YOU?", "WHAT IS \*", "DO YOU \*".
- **Persuasive agreement** – if the user persuades to do something or invites my\_Eliza to do something, my\_Eliza's goal is to show whether she agrees or not, for example patterns like: "LET'S \*", "CAN YOU \*", "WOULD YOU \*".
- **Topical focus** – to keep on conversing on the same topic and beware if it is changing.
- **Explanation statements** – to reply the user's statements that require specification and explanation.
- **Reflecting feeling** - to keep consistent with the user's current affective state.
- **Alignment** - to keep consistent with the system's current reaction affective state (system's status) and system's preferences.

The system recognizes those types of sentences using a dialogue scheme. A dialogue scheme is a code of dialogue corpora for higher-level dialogue structure into a firmer basis. An example such a system using this approach is Carletta et.al [CAR95] based on Anderson's HCRC Map Task Corpus 1991. In their research, the coding distinguishes three levels of dialog structure. At the highest-level dialogues are decided in **transactions** or sub dialogues corresponding to major steps in the participants' plan for completing the task. At the intermediate level transactions are made up of the **conversational games**, a sequence of utterances, which consist of initiations and responses. At lowest level conversational **moves** are utterances, which



are initiations and responses named according to their purpose, for instance retrieving information or providing information. In my\_Eliza, transactions correspond to major steps in the dialogue, like posing the query or presenting the topic discussion. Moves correspond to dialogue acts, which are in our case also initiations and responses making up smaller process to accomplish a transaction. My\_Eliza's dialog scheme has to cover all type of human spoken natural language. The main elements of our coding scheme are moves or dialogue acts, like question, statement, agreement, acknowledgement, pause and so on. The coding scheme we intend to use this layer, therefore my\_Eliza is able to give reply sentence more accurately. Other purpose is to recognize whether the goal is appealing or not appealing, which will influence system's affective state as reaction to the situation. The detailed design about a dialogue coding in my\_Eliza is left out as a recommendation for future work of this system.

In system's GSP database, my\_Eliza also defined what she likes or dislikes. This information can be used in conversation (see figure 4-21 below). The `<condition>` tag and the `<li bot .. >` tag will refer to the system's GSP database to seek whether the "bot" property name is defined and then get a value. The system will use the template if the condition is true. In the example above, the system concluded banana is not one of her favorite fruit and in fact she hates it.

```
Example the content of system's GSP database:
<bots>
  <bot id="my_Eliza_ver_1" enable="true">
    <property name="name" value="my_Eliza"/>
    <property name="birthday" value="22 August 2002"/>
    <property name="birthplace" value="Delft"/>
    <property name="master" value="Ska"/>
    <property name="favorite_color" value="yellow"/>
    <property name="favorite_fruit" value="orange, grape"/>
    <property name="hate_attitude" value="lie, distress"/>
    <property name="hate_fruit" value="banana, watermelon"/>
    ...
  </bot>
  ...
</bots>

That information can be used in category below:

<topic name="FRUIT">
<category><effect name="-"> <that>HOW ABOUT YOU</that>
  <pattern>I LIKE BANANA *</pattern>
  <template><think><setconcern>+</setconcern><think>
  <condition>
    <li bot name="hate_fruit" value = "banana">Hoek! .. I hate it very much!</li>
    <li bot name="favorite_fruit" value = "banana">Oh, I love it!</li>
    <li>I never eat it, how does it taste?</li>
  </condition>
  </template></affect>
</category> </topic>

Example fragment:
User   : What kind of fruit do you like?
My_Eliza: I like orange a lot. How about you?
User   : I like banana more.
My_Eliza: Hoek! .. I hate it very much!
```

Figure 4- 21 Example of system's GSP database and how to use it in a conversation

## 4.4 Facial Display Construction Layers

The system constructs a facial display through some layers of processes, which are called Facial Display Construction Layers (figure 4-22). The layers (described below) responsible to synthesize the current user's string input and deliver an appropriate facial display.

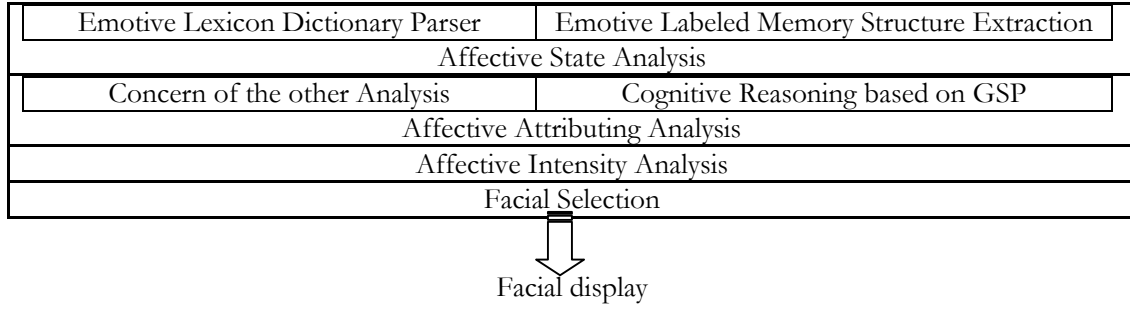


Figure 4- 22 Facial display construction layers

## Affective State Analysis

This layer is responsible to extract emotion-eliciting factor information from the user's input string, the system's reply sentence and the memory structure by executing two tasks: *Emotive Lexicon Dictionary Parser* and *Emotive Labeled Memory Structure Extraction*, which will be discussed below.

### 1. *Emotive Lexicon Dictionary Parser*

The basic idea is this layer has six counters, which work like a thermometer [0..MAXVALUE] and each counter is dedicated to one of Ekman's universal emotion type: happiness counter, sadness counter, fear counter, disgust counter, surprise counter and anger counter. This layer receives a string, parses it into tokens and searches the correspondence tokens in the emotive lexicon dictionary (see section 4.2). If the parser found the token in the dictionary belonging to one of the six universal emotion types then the counter of this emotion type will be incremented (see algorithm on figure 4-23 below). The layer uses a summation factor,  $s$ , as an incremental value. It means every emotive lexicon  $l_i$  that belongs to dictionary  $d_i$  in a sentence contributes a typical value  $s$  to the  $C_i$  counter. For example: if the string is "I am happy" and the parser finds token "happy" in happiness dictionary then the happiness counter will be incremented using the equation:

$$\forall \text{Lexicon } l_i \in d_{\text{happiness}} \mid C_{\text{happiness}}(t) = C_{\text{happiness}}(t-1) + s$$

While the other counters will be decremented using the equation:

$$\forall i \neq \text{happiness} \mid C_i(t) = C_i(t-1) + \text{distance\_value}[i, \text{happiness}]$$

$i = \{\text{happiness, sadness, anger, fear, disgust, surprise}\}$

```

Get the string; Set token as the first word of the string
while token is not the last word of the string
  if it is negation then negation_flag = true
  else if it is value high comparative then comparative_value = HIGH
  else if it is value low comparative then comparative_value = LOW
  else found=false
    while there is emotion type in the dictionary and not found
      while there is emotive lexicon and not found
        If match or partition match with the token then
          CalculateCounters(emotion type, negation_flag, comparison_value)
          negation_flag = false
          next emotive lexicon
        next emotion type
      next token
    //end while
  if there is no maximum counter then emotion type is neutral
  else send maximum counters with the emotion type

```

Figure 4- 23 Emotive lexicon dictionary parser

The changes of one emotion influence other emotions' degree. If the system finds new emotive lexicon then all counters will be calculated. We use the distance values between emotions



dataset based on the work of Hendrix and Ruttkay [HEN98] (see table 4-3) for decremental value.

Table 4- 3 Distances dataset between six universal emotions

	Happiness	Surprise	Anger	Fear	Disgust	Sadness
Happiness	0	3.195	2.637	?	1.926	2.554
Surprise		0	3.436	?	2.298	2.084
Anger			0	?	1.506	1.645
Fear				0	?	?
Disgust					0	1.040
Sadness						0

The changes of counters depend on the intensity value of active emotion type. This intensity is determined from emotion lexicon dictionary, which defines each lexicon in the dictionary [ $\langle \text{lexicon} \rangle$ :  $\langle \text{intensity value} \rangle$ ] with  $\langle \text{intensity value} \rangle$  is an integer value [1..3]. It might also be determined by the comparative value before the token, for example the word “very” in the string “I am very delighted” shows high intensity and the phrase “a little bit” in the string “my parents are a little bit disappointed” shows low intensity. There are three types of intensity values used in this layer:  $I_h$  for high intensity,  $I_m$  for medium intensity and  $I_l$  for low intensity, where  $I_h > I_m > I_l$ .

Therefore the equations above become:

$$\begin{aligned} \forall \text{Lexicon } l_i \in d_i \mid C_{i(t)} &= C_{i(t-1)} + I_i \cdot s ; i = \text{active emotion type} \\ \forall j \neq i \mid C_{j(t)} &= C_{j(t-1)} - \text{distance\_value}[j, i] \\ j &= \{\text{happiness, sadness, anger, fear, disgust, surprise}\} \end{aligned}$$

These equation shows the possibility that at a certain moment the counters goes up quickly and/or goes down quickly, which means that the active emotion type is very dominant and has high intensity. The incremental process is calculated until the counter value = MAXVALUE and the decremented process is also calculated until the counter value = 0.

The parser also identifies negation of a lexicon. We assume that emotive lexicon dictionary has covered this type of lexicon, with “[not  $\langle \text{lexicon} \rangle$ ,  $\langle \text{intensity-value} \rangle$ ]” form, for example “[not bad; 1]” belongs to happiness dictionary and “[not well, 1]” belongs to sadness dictionary. The result of this layer is the emotion type with the highest counter value. If there is no maximum counter, which means the parser do not find any emotive lexicon, then the default emotion type is neutrality. The counters’ difference sets the threshold value to activate certain emotion types as the affective state. An emotion type is the active emotion type if its counter difference is bigger than other counters’ difference. Hence, if this layer parses the user’s string input then the results are the current user’s affective state and if it parses the system’s reply sentence then the results are the candidate of system’s reaction affective state. Those results are accompanied by their intensity. This active intensity  $I'_i$  is defined by comparing the active emotion type counter  $C_i$  with the other counters  $C_j$ , where  $j \neq i$ , by the equation:

$$\begin{aligned} I'_i &= \Sigma(C_i - \forall C_j) / (n-1) \mid j \neq i, n = \text{number of the counters} = 6 \\ j &= \{\text{happiness, sadness, anger, fear, surprise, disgust}\} \end{aligned}$$

A counter [0..MAXVALUE] is divided into four slots: 0, 50%, 75% and 100% of MAXVALUE. Therefore:

- The intensity = LOW if and only if  $I'_i < 50\%$  of MAXVALUE
- The intensity = MEDIUM if and only if  $50\%$  of MAXVALUE  $\leq I'_i \leq 75\%$  of MAXVALUE
- The intensity = HIGH if and only if  $I'_i > 75\%$  of MAXVALUE



## 2. *Emotive Labeled Memory Structure Extraction*

The basic idea is to extract the memory structure unit and find an emotive label. When the user feeds a string to the system, the Syntactic-Semantic Analysis Layer will pattern match the string input with the pattern of the categories in the system's memory structure. This operation will matches to one category. This category contains also two tags that store an emotive label between: <affect> tag and <concern> tag. This layer is responsible to get the value inside those tags. The results of this layers are the value current AFFECT (positive, negative or joke emotive label) and the value current CONCERN (positive, negative or joke emotive label).

## Affective Attributing Analysis

This layer is a knowledge-based system that controls all processes in the FDC-Layers using an affective knowledge base (see section 4.2.2.2). This layer receives the results from all layers in the FDC-Layers. It uses them to process and deliver a result based on twenty-six emotion types in table 4-1. Note, this is the only layer in the FDC-layers that deliver a result based on OCC's theory emotions classification.

This layer has two tasks:

### 1. *Concern of the other analysis*

This task is to analyze the user's affective state based on the user's string input. It receives emotion-eliciting variables that have been extracted by other FDC-layers (see section 4.2.2.2). Those variables are used in the reasoning process to determine system's reaction affective state as system's first reaction to user's string input, for example:

- IF *user's affective state candidate* is happiness and *current user situation type* is positive and *current system's affective state* is happiness  
THEN *system's reaction affective state* is joy.
- IF *user's affective state candidate* is sadness and *current user situation type* is negative and *current system's affective state* is happiness  
THEN *system's reaction affective state* is fear.

### 2. *Cognitive reasoning based on GSP*

This task is to analyze the system's reaction affective state based on the system's reply sentence, the user's previous affective state, the user's preferences, the current user's affective state and the system's GSP (see section 4.2.2.2). The user's preferences are determined from user's personal database whether she/he likes or dislikes the conversation content (see the example category about favorite movie in figure 4-11 and 4-12). If it does not find anything about the content in the database, the system will assume "don't care". As mentioned in section 4.2.1.4, the system's GSP database stores system's goal, current emotional status and its preferences. From the system's goals, this layer has to decide whether the goal is appealing or not. From the system's status, this layer will use the system's previous reaction affective state in the reasoning process. The final step is, from the system's preferences database, this layer determines whether the system likes or dislikes the conversation content (see the example category about favorite fruit in figure 4-21). If this layer does not find anything about the content in this database, it will assume "don't care".

This task receives those emotion-eliciting factors from the blackboard system. Those factors are used for the reasoning process to determine the second system's reaction affective state as cognitive processed facial display to convey the system's reply, for example:

- IF *the candidate of system-reaction affective state* is happiness and *current user's situation type* is positive and *current system's situation type* is positive and *last user's situation type* is positive and *last system's situation type* is positive and *last system's reaction affective state* is joy and *current system's affective state (thermometer)* is happiness and *user's preference* is don't care and *system's preference* is don't care and *system's goal* is appealing  
THEN *system's reaction affective state* is happy-for.





- IF the candidate of system-reaction affective state is sadness and current user's situation type is negative and current system's situation type is positive and last user's situation type is negative and last system's situation type is positive and last system's reaction affective state is fear and current system's affective state (thermometer) is sadness and user's preference is don't care and system's preference is don't care and system's goal is appealing  
THEN system's reaction affective state is sorry-for.

The implementation of this layer is clearly explained in section 6.2.4.

### Affective Intensity Analysis

This layer is responsible to determine the emotion intensity and at the same time identify system's reaction affective state based on conversation flow. The basic idea is using six "thermometers" [0..MAXDEGREE] where each of them is dedicated to Ekman's six universal emotion types: happiness thermometer, sadness thermometer, fear thermometer, disgust thermometer, surprise thermometer and anger thermometer. While the conversation is going on, this layer observes and calculates the degree of six emotion types based on system's previous affective reaction system that was identified by the affective attributing analysis layer (see algorithm on figure 4-24).

```
Set_Thermometer subroutine (emotion type = new system's reaction affective state)
  Calculate_degree(Happiness_thermometer, emotion type)
  Calculate_degree(Sadness_thermometer, emotion type)
  Calculate_degree(Fear_thermometer, emotion type)
  Calculate_degree(Surprise_thermometer, emotion type)
  Calculate_degree(Disgust_thermometer, emotion type)
  Calculate_degree(Anger_thermometer, emotion type)
//end subroutine
```

Figure 4- 24 Algorithm to set system's reaction affective state thermometer

The calculation process is similar to the counter in the *emotive lexicon dictionary parser* for every thermometer  $T_i$ , using the equation:

```
IF an active emotion type is i
  THEN  $T_i(t) = T_i(t-1) + I_i \cdot s$ 

 $\forall j \neq i \mid T_j(t) = T_j(t-1) - \text{distance\_value}[j, i]$ 
Where  $j = \{\text{happiness, sadness, anger, fear, disgust, surprise}\}$ 
```

Where  $s$  is a summation factor as an incremental value and  $I_i$  is the intensity value. There are three types of intensity value used in this layer:  $I_h$  for high intensity,  $I_m$  for medium intensity and  $I_l$  for low intensity, where  $I_h > I_m > I_l$ . This layer also uses the distance values between emotions dataset based on the work of Hendrix and Ruttkay [HEN98] in table 4-3 for decremental value, since the changes of one emotion thermometer influence the other emotions' degree. If the *affective attributing layer* defines a new cognitive processing of system's reaction affective state then all thermometers will be calculated.

It is also possible that the thermometer degree goes up and/or goes down quickly. The maximum incremental process is until the thermometer degree = MAXDEGREE and the minimum decremented degree = 0. The result of this layer is the highest degree of six active thermometers to observe the current system's reaction affective state. This result will be used by the affective attributing analysis to analyze current system's reaction affective state. If all thermometers have equal degree, then the default emotion type is neutrality. This result is also accompanied by its intensity. This active intensity  $I'_i$  is defined by comparing the active emotion type thermometer  $T_i$  with the other thermometers  $T_j$ , where  $j \neq i$ , by the following equation:

$$I'_i = \frac{\sum (T_i - \forall T_j)}{(n-1) \mid j \neq i, n = \text{number of the thermometers} = 6}$$

$$j = \{\text{happiness, sadness, anger, fear, surprise, disgust}\}$$



A thermometer [0..MAXDEGREE] is divided into four slots: 0, 50%, 75% and 100% of MAXDEGREE. Therefore:

- The intensity = LOW if and only if  $r'_i < 50\%$  of MAXDEGREE
- The intensity = MEDIUM if and only if  $50\% \text{ of MAXDEGREE} \leq r'_i \leq 75\% \text{ of MAXDEGREE}$
- The intensity = HIGH if and only if  $r'_i > 75\% \text{ of MAXDEGREE}$

Facial selection layer will use this intensity values to select an appropriate facial display. Human emotion deals with mood and character. A mood is stable emotion for a long time and human character is a tendency to react emotionally. For reason of simplicity, we do not consider those cases here. For its substitution, we take three assumptions below:

1. The character of my\_Eliza similar to a psychoanalyst; she does not show her actually feeling.
2. My\_Eliza only considers the current user's affective state from the current user's string input. The system does not consider user's current mood or character.
3. My\_Eliza's mood can be analyzed from the result of the *intensity analysis layer*. This mood is not stored in system's database. It does not influence directly to the reply sentence construction or the facial display selection.

## Facial Selection

This layer is responsible to select a facial display. The selection depends on the result of two previous layers: Affective Attributing layer and Intensity Analysis layer. This layer receives two variables: current system's reaction affective state and its intensity.

Table 4- 4 Corresponding between six universal emotion types and OCC's theory emotion types

No.	Name and Emotion Type	Universal emotion type
1.	<b>Joy:</b> pleased about an event	<b>Happiness</b>
2.	<b>Distress:</b> displeased about an event	<b>Sadness</b>
3.	<b>Happy-for:</b> pleased about an event desirable for another	<b>Happiness</b>
4.	<b>Gloating:</b> pleased about an event undesirable for another	<b>Happiness</b>
5.	<b>Resentment:</b> displeased about an event desirable for another	<b>Sadness</b>
6.	<b>Sorry-for:</b> displeased about an event undesirable for another	<b>Sadness</b>
7.	<b>Hope:</b> pleased about a prospective desirable event	<b>Surprise</b>
8.	<b>Fear:</b> displeased about a prospective undesirable event	<b>Surprise</b>
9.	<b>Satisfaction:</b> pleased about a confirmed desirable event	<b>Happiness</b>
10.	<b>Relief:</b> pleased about a disconfirmed undesirable event	<b>Happiness</b>
11.	<b>Fears-confirmed:</b> displeased about a confirmed undesirable event	<b>Fear</b>
12.	<b>Disappointment:</b> displeased about a disconfirmed desirable event	<b>Sadness</b>
13.	<b>Pride:</b> approving of one's own act	<b>Happiness</b>
14.	<b>Admiration:</b> approving of another's act	<b>Happiness</b>
15.	<b>Shame:</b> disapproving of one's own act	<b>Sadness</b>
16.	<b>Reproach:</b> disapproving of another's act	<b>Anger</b>
17.	<b>Liking:</b> finding an object appealing	<b>Happiness</b>
18.	<b>Disliking:</b> finding an object unappealing	<b>Disgust</b>
19.	<b>Gratitude:</b> admiration + joy	<b>Happiness</b>
20.	<b>Anger:</b> reproach + distress	<b>Anger</b>
21.	<b>Gratification:</b> pride + joy	<b>Happiness</b>
22.	<b>Remorse:</b> shame + distress	<b>Sadness</b>
23.	<b>Love:</b> admiration + liking	<b>Happiness</b>
24.	<b>Hate:</b> reproach + disliking	<b>Disgust</b>
25.	<b>Normal:</b> do not have dominant emotion	<b>Neutrality</b>
26.	<b>Uncertainty:</b> confuse or unsure	<b>Uncertainty</b>

The facial display dictionary of this layer only stored twenty-two facial displays distinguished by Ekman's seven universal emotion types (: neutrality, happiness, sadness, fear, disgust, surprise and anger and an uncertainty facial display) and three types of intensities: LOW, MEDIUM and HIGH. Except for neutrality emotion type, it does not have any intensity value. Since the affective



attributing layer gives the result of system's reaction affective state as one of twenty-six emotion type in table 4-1 (twenty-four emotion types of OCC's theory plus uncertainty and normal), this layer uses the rule on table 4-4 (above) to get one-to-one corresponding between emotion type and facial display dictionary. This mapping is based on the clustering process of OCC's theory emotion types to six Ekman's universal emotion types in table 4-2.



## Chapter 5 My\_Eliza Dialog Box: Preprocessing

As mentioned in chapter 4, our research approach can be outlined into three major (incrementally) implementation layers (figure 4-1): creating a dialog box, build a stimulus response generation of facial displays and build a cognitive processor of facial displays. The first step (preprocessing) in building a multimodal communication system is to create a dialog box. This dialog box is an automatic human computer interaction based on typed natural language, which is able to construct a reply sentence automatically every time the user sends a string input.

This chapter is organized as follows. First, the issues of automatic reply construction based on Weizenbaum's Eliza are summarized in section 5.1. Then selection and arrangement of the approach for constructing a reply sentence based on anaphoric analysis, topic maintenance and repetition recognition is discussed. In our prototype we take the dialog box of Wallace's A.L.I.C.E as a starting point and transformed it to develop a robust dialog box for automating emotion recognition and generating facial displays. Next, the list of pattern rules of my\_Eliza as system's memory structure units is presented in 5.2. This list is constructed based on A.L.I.C.E's brain structure in AIML form with two extended tags: <affect> tag and <concern> tag. The reply sentence construction of my\_Eliza using both extended tags is explained also in this section. Finally, section 5.3 describes the implementation of my\_Eliza's Dialog Box.

### 5.1 From Eliza to A.L.I.C.E

As mentioned in section 2.1, the pattern matching operation of the original Weizenbaum's Eliza [WEI65] still has three major problems [SIM70]: lack of anaphoric analysis, lack of ability to maintain the topic, and lack of ability to get the meaning beyond the sentence.

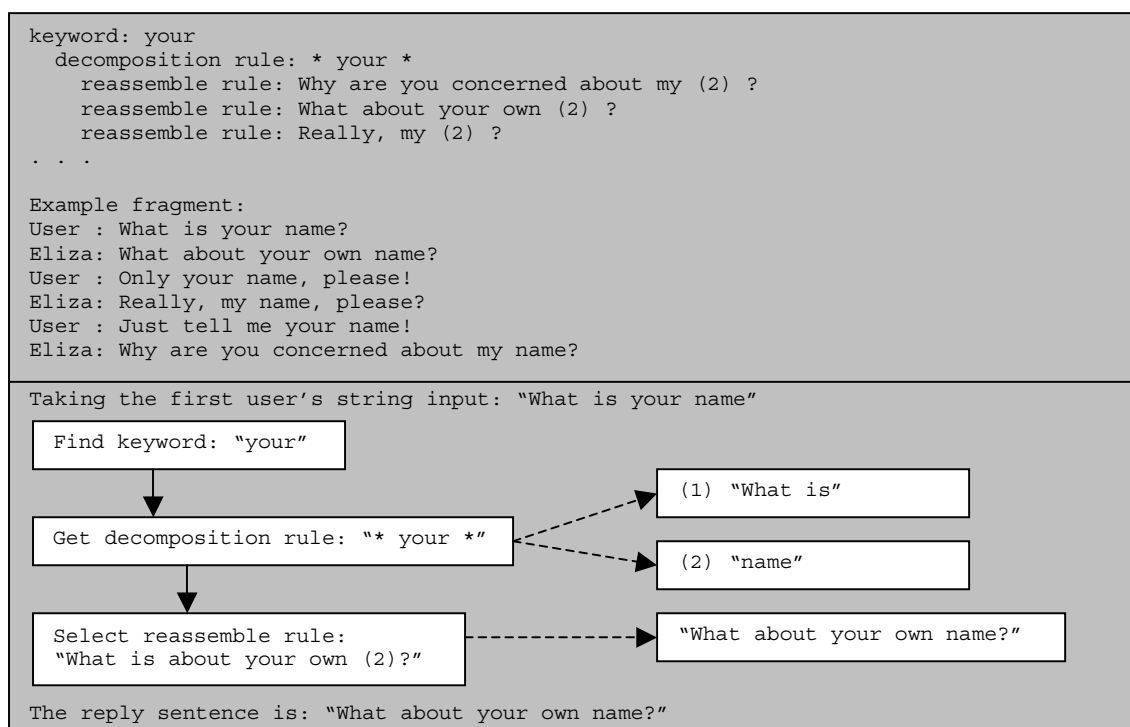


Figure 5- 1 Example of Eliza's memory unit and fragment between Eliza and the user



These minus abilities make Eliza's interaction with human users often ambiguous, sometimes not naturalistic and lead to a lot of misunderstanding. However Weizenbaum has introduced the personal pronoun transformations common to many QA system implementations. As described in section 1.1, Eliza's memory structure consists of a list of input keywords (see figure 1-2). Each of the input keywords is linked to a list of decomposed rules; each rule is a possible input pattern that contains the keyword. Each of the decomposed rules consists of reassemble rules; these rules are the reply sentences that will be selected randomly. The example of Eliza's memory structure unit is in figure 5.1 above. It takes input keyword "your" with decomposition rule "\* your \*". With asterisk sign (\*), the decomposition rule tells the program to divide the user's string input into two phrases: one phrase before the word "your" and another phrase after the word "your". According to the rule the input "What is your name" can be divided into two phrases: first, "what is" and second, "name". One reassemble rule randomly selected from the list in our example is "What about your own (2)". Finally, the shown number indicates that the reassemble rule will be substituted by the second phrase: "name". Then the reply sentence is "What about your own name?".

As mentioned in section 2.2 in order to eliminate those lacks, Wallace [WAL95] proposed a new approach by expanding the memory structure with those abilities. The idea was put in AIML (Artificial Intelligence Markup Language) - an XML (Extensible Markup Language) specification for programming the QA system behavior. The behavior is controlled by a huge number of memory structures in AIML form. The transformation from the user input into a reply sentence remains similar with Weizenbaum' Eliza. However, the memory structure unit (inside <category> tag) is expanded to support the three abilities above (see figure 5-2 below). AIML forms do not use input keyword-matching operations, but instead they use directly pattern-matching operation. An AIML unit consists of two sides: *pattern side AIML expression* - PSAE that can appear in the tag <pattern>, <that>, and <topic>, and *template side AIML expression* - TSAE that appear inside the tag <template> (see Appendix A). Inside <pattern> tag is the user input pattern, inside <that> tag is a reference to the system's previous reply, and inside <topic> tag is reference to the current topic of the conversation. Inside <template> tags is the reply sentence of the system. This tag can contain more than one possible reply sentence. Each input pattern can have a collection of different reply sentences based on its topic and/or previous reply sentence.

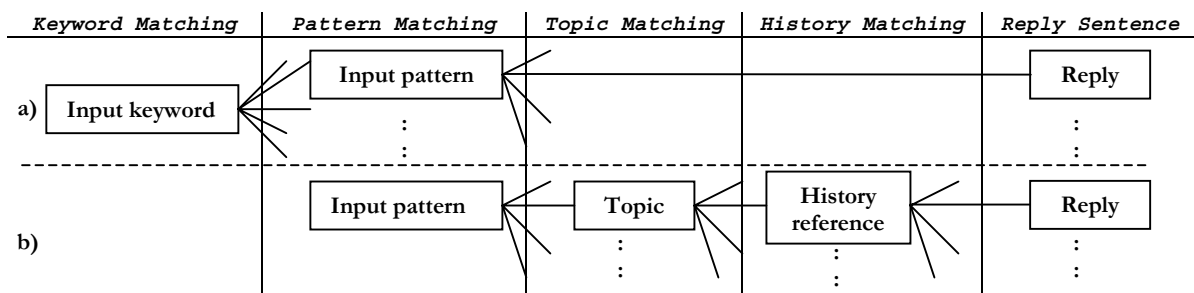


Figure 5- 2 Comparing Eliza's memory structure (a) and A.L.I.C.E's memory structure (b)

An example of AIML's memory structure unit is on the figure 5.3 below. It takes the input pattern "call you \*" with the topic "NAME" and system's previous reply "MY NAME IS \*". With an asterisk sign (\*), the input pattern tells the program put any phrases, which will be substituted by it, on the *star stack* (the stack that is used to save all asterisk sign substitution) with index n (n is incremented if the system pushes a new element). The usage of asterisk sign is not very strict in AIML since the algorithm of pattern matching operation uses the longest best fit. The asterisk sign is used if and only if the reply sentence refers to the phrase that is substituted by the sign. To refer such phrase we can use <star/> tag to get the top element in *star stack*. Taking one of user's string input in the figure, "Can I call you Madonna?" and using anaphoric analysis, the system already stored that current topic "NAME" (see the first category on figure 5-3, when memory unit tells the



system to set current topic using `<set_topic>` tag) and the system's previous reply sentence is "MY NAME IS \*" (the asterisk sign shows that the rest of the sentence will be ignored).



Figure 5- 3 Example of A.L.I.C.E's memory unit and fragment between A.L.I.C.E and the user

The system will try to execute the best-matching operation against the topic, the system's previous reply and the input pattern (see Appendix A.3) using the algorithm in the figure 5-4 below. Using that algorithm, in our example the system finally finds the second category and selects one of the templates randomly: "`<star/>? Huh! Like I've told you my name is <bot name='name'>`". Finally, it replaces `<star/>` tag with the top element of *star stack*. AIML also provides tags to get the system's personality profile using `<bot name="name">`, means the system searches the system's name on the system's personality profile (all list of tags in AIML can be seen in table A-1, see Alicebot's structure in appendix B). Finally, the system sends the reply sentence to the user: "Maddona? Huh? Like I've told you my name is Alice".





```
Find ATOMIC CATEGORY with the same TOPIC and THAT, or
Find ATOMIC CATEGORY with the same TOPIC, or
Find DEFAULT CATEGORY with the same TOPIC and THAT, or
Find DEFAULT CATEGORY with the same TOPIC, or
Find ATOMIC CATEGORY with the same THAT, or
Find ATOMIC CATEGORY, or
Find DEFAULT CATEGORY with the same THAT, or
Find DEFAULT CATEGORY
```

Figure 5- 4 The algorithm of A.L.I.C.E pattern matching operation

Using this specification language, Wallace built A.L.I.C.E (Artificial Linguistic Internet Computer Entity) – a QA system that is controlled by a collection of autonomous client and server communicating via TCP/IP. The user can communicate with A.L.I.C.E's server through HTTP server (figure 5-4). There are several versions of A.L.I.C.E implemented (see appendix A), however in our prototype we only use the approach of Program D written in Java language (see appendix B for detailed explanation about Program D). The program D A.L.I.C.E (called Alicebot) provides a huge list of pattern rules in AIML form and the engine. The engine is responsible to establish server-client communication via a HTTP server, extracts user's input into string input pattern, provides the algorithm for best-matching pattern for each input, interprets AIML language specification and constructs the reply sentence to the user.

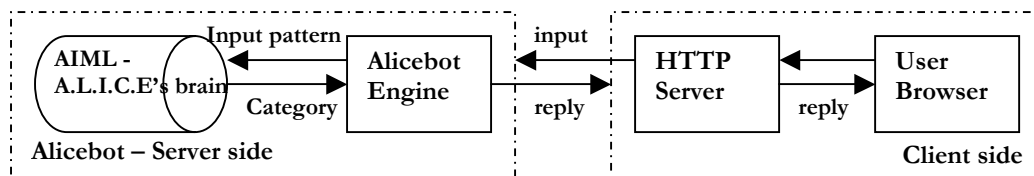


Figure 5- 5 The process involved in a typical transaction with A.L.I.C.E

## 5.2 From A.L.I.C.E to My\_Eliza

The Implementation of My\_Eliza is based on Program D - A.L.I.C.E. Since the source program of Program D is shareware and freeware, we can find it in [WAL95]. Wallace gave away the secret of A.L.I.C.E chat robot development to anyone who wants it, permitting the greatest possible dissemination, utilization and technical improvement of the A.L.I.C.E technology. Therefore, all the structure of My\_Eliza's dialog box program is based on Program D (the detail structure of Program D – A.L.I.C.E can be found in Appendix B).

My\_Eliza is also controlled by a collection of autonomous client-server communication via TCP/IP. The user can communicate with my\_Eliza's server through the HTTP server (see figure 5-6). My\_Eliza server provides the blackboard system (to be shorted: the system). The system has the same responsibility as the engine of Alicebot in order to maintain user-system interaction.

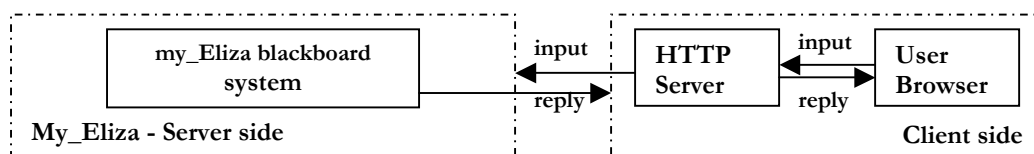


Figure 5- 6 The process involved in a typical transaction with my\_Eliza

In order to fulfill my\_Eliza's design requirements that have been mentioned in section 2.6, there was still a lot of work to be done. Figure 5-7 below shows that the implementation of Program D only covers about 25% of the work (see the shadowed layers).

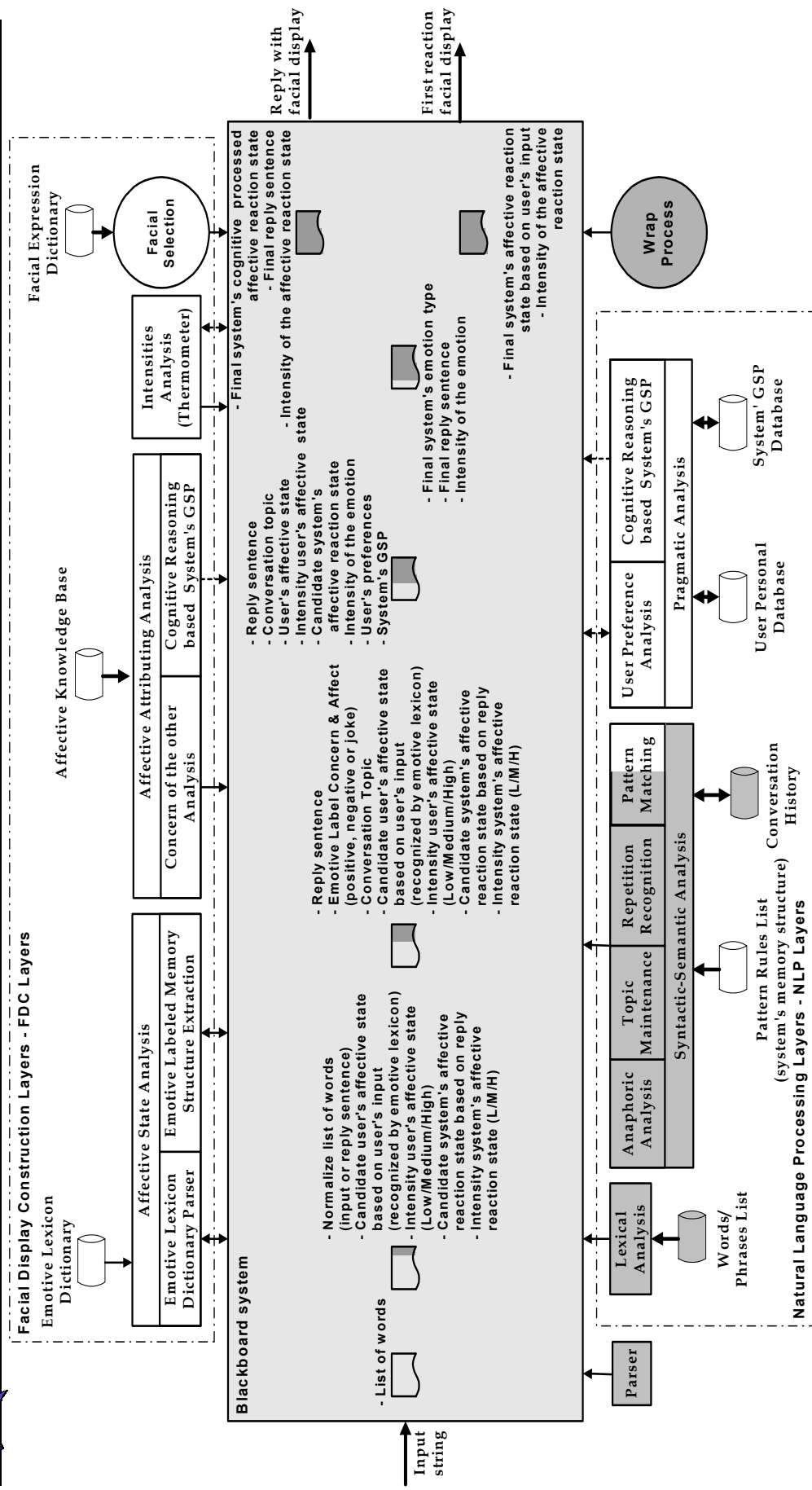
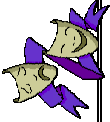


Figure 5- 7 My\_Eliza blackboard system (the shadowed layers have been implemented in A.L.I.C.E)



The programmers of Program D have implemented a robust parser, the lexical analysis layer and most of the Syntactic-Semantic Analysis Layer. Alicebot also has provided a robust list of word and phrases, which will be used in the lexical analysis layer, and a database of conversation history, which will be used to refer to the current topic, user's previous input and system's previous reply. However as mentioned in the general design of my\_Eliza (see section 4.2.1.2), we expand the AIML structure with two new extended tags to build a robust dialog box for a multimodal communication system that can recognize emotions and generate facial displays automatically. Those new tags are: first <affect> tag to show current user's affective situation type as a positive or negative affect and second <concern> tag to show current system's affective reaction to the user's situation type also as a positive and negative affective reaction. <affect> tag can appear in PSAE and TSAE, but <concern> tag only appears in TSAE.

The dialog box of my\_Eliza is modeled after a certain personality, i.e. psychoanalyst (similar to the role of Weizenbaum's Eliza). Therefore, the contents of memory structure of my\_Eliza itself do not use A.L.I.C.E's memory structure, but using Eliza's memory structure instead. Figure 5-8 shows the transformation of Eliza's memory structure into my\_Eliza's. Basic structure of my\_Eliza's memory unit is:

```
<topic name="TOPIC">
<category>
  <affect name="AFFECT">
    <that>THAT</that>
    <pattern>PATTERN</pattern>
    <template><think><setconcern>CONCERN</setconcern></think>TEMPLATE </template>
  </affect>
</category>
</topic>
```

We may not use <topic> and <that> tags in a category.

```
Example of Weizenbaum's Eliza's memory structure:
keyword: your
decomposition rule: * your *
reassemble rule: Why are you concerned over my (2) ?
reassemble rule: What about your own (2) ?
reassemble rule: Are you worried about someone else's (2) ?
reassemble rule: Really, my (2) ?

Translated to my_Eliza's memory structure:
<category>
<affect name="*">
<pattern>YOUR *</pattern>
<that>*</that>
<template><random>
  <li><think><setconcern>#</setconcern></think>Why are you concerned over my
    <star/>?</li>
  <li><think><setconcern>+</setconcern></think>What about your own <star/>?</li>
  <li><think><setconcern>+</setconcern></think>Are you worried about someone else's
    <star/>.</li>
  <li><think><setconcern>#</setconcern></think>Really, my <star/>.</li>
</random></template>
</affect>
</category>
```

Figure 5- 8 Example of a transformation from Eliza's memory structure unit to my\_Eliza's

In my\_Eliza's memory unit does not use a keyword input matching operation (similar to A.L.I.C.E, see figure 5-10 comparing the memory structure of Eliza, A.L.I.C.E and my\_Eliza). Figure 5.9 shows how we can use Eliza's memory structure in my\_Eliza. First we determine user's situation type when he/she sends such an input pattern to the system. There are four situation types that



may appear in user's situation type (see section 4.2.1.2): positive (+), negative (-), joke (#) and any or normal (\*). In our example we determine user's situation type when he/she types the input pattern "\* your \*" is normal (see the heuristic method to decide user's situation type in section 4.2.1.2). The decomposition rule in Eliza's memory unit is a pattern input in my\_Eliza and written inside <pattern> tag. We may determine the topic of the memory unit and the previous reply that triggers it. The reassemble rule in Eliza is the reply sentence of my\_Eliza and written inside <template> tag.

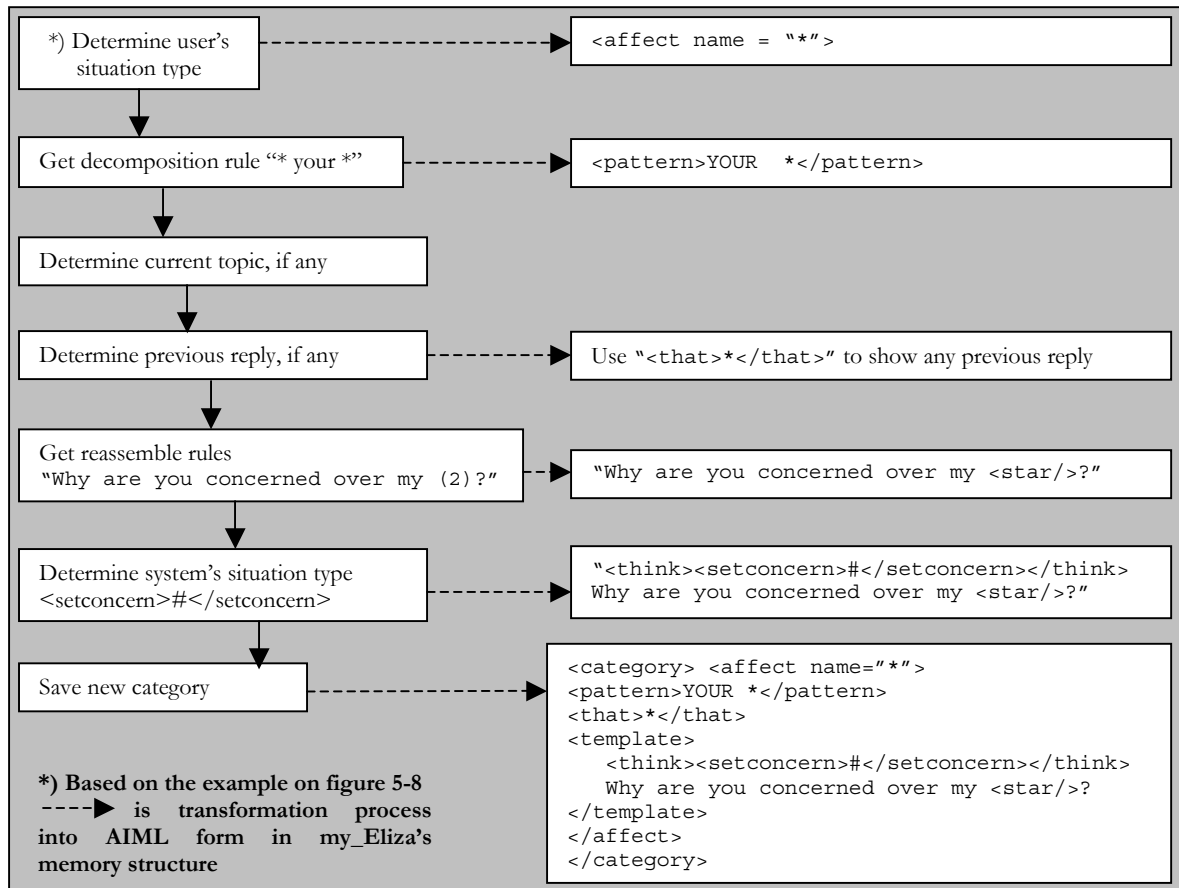


Figure 5- 9 Step by step transformation from Eliza's memory structure to my\_Eliza's

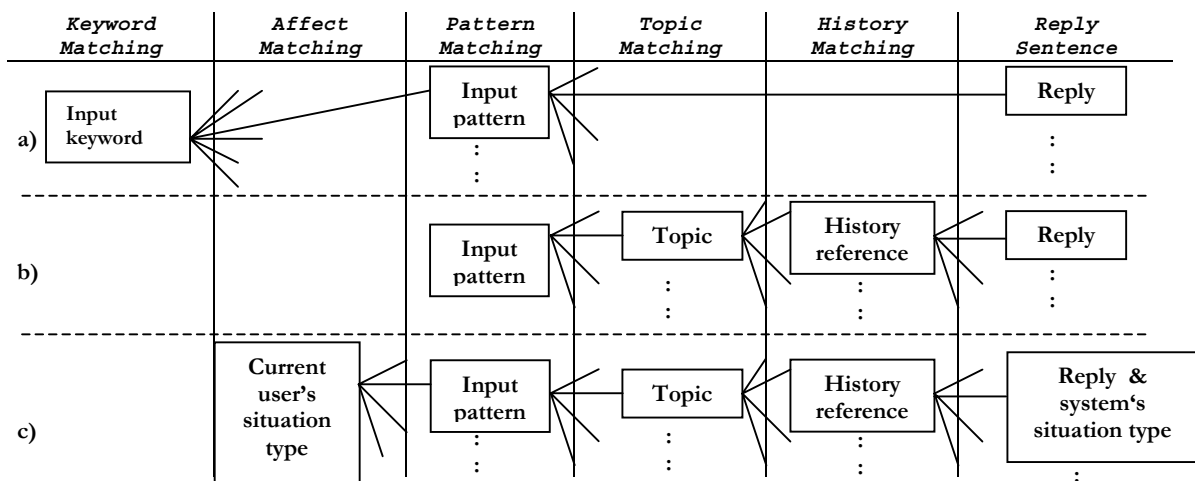


Figure 5- 10 Comparing the memory structure of Eliza (a), A.L.I.C.E (b) and my\_Eliza (c)



The numeric sign in Eliza's reassemble rule is substituted by a `<star/>` tag or `<star index="n">`, where  $n$  is the index of phrase in *star stack*. By analysis of the system's reply sentence (see section 4.2.1.2), we determine the situation type of the reply sentence as a reaction to a user's input sentence. There are four situation types that may appear in system's situation type (see section 4.2.1.2) the same values as user's situation type. In our example we determine that the system's situation type is sending a joke to the user (see the heuristic method to decide system's situation type in section 4.2.1.2). Figure 5-10 above shows that using extra two tags, my\_Eliza has the possibility to have a bigger memory structure than A.L.I.C.E.

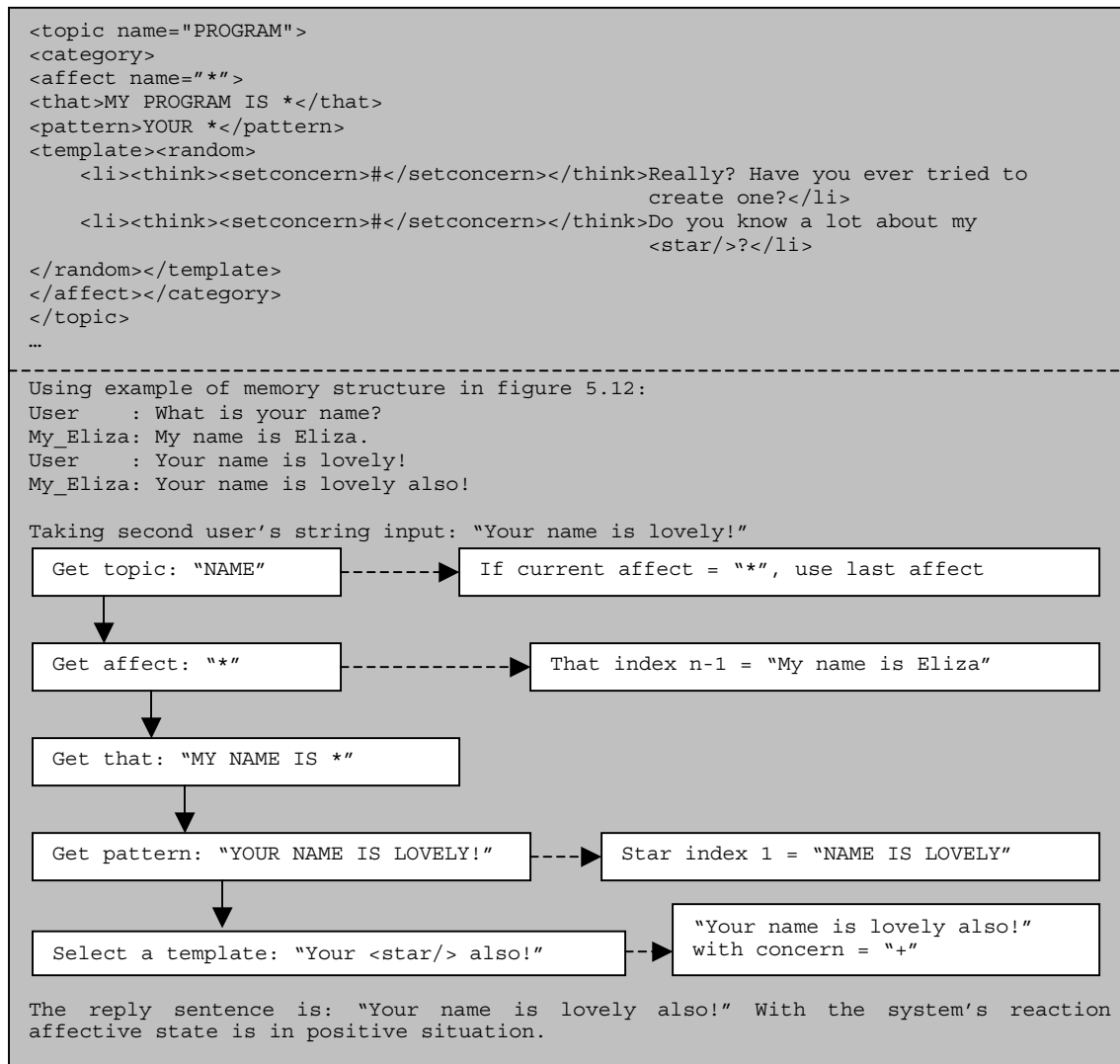


Figure 5- 11 Example of a my\_Eliza's memory unit with different topic

By different topic and/or different previous reply, the same input pattern can be created into a different variety of categories. It means also a different variety reply sentences of the system. In example displayed in figure 5-11, we take the same input pattern with the example of memory units in figure 5-8 ("YOUR \*"). By indicating different topic ("PROGRAM") and different previous reply that may trigger this category ("MY PROGRAM IS \*"), the new category has completely different reply sentences and reaction to the user's string input. It is also possible to have the same reply sentences with different system's situation type, which means more variation of possible my\_Eliza's facial displays for the facial display selection layer in my\_Eliza prototype-1 (see example in figure 5-12 and table 5-1 below).



```
<category> <affect name="*">
<that>*</that>
<pattern>WHAT IS YOUR NAME</pattern>
<template><think><setconcern>+</setconcern><setaffect>+</setaffect></think>
    My <set_topic>name</set> is <bot name="name">.
</template></affect>
</category>

<topic name="NAME">
<category> <affect name="*">
<that>MY NAME IS *</that>
<pattern>YOUR *</pattern>
<template><random>
    <li><think><setconcern>#</setconcern></think>Your <star/> also!</li>
    <li><think><setconcern>#</setconcern></think>Mind that, what is your name?</li>
    <li><think><setconcern>#</setconcern></think>Can you spell my name? I need to
        know that you can write it correctly</li>
    <li><think><setconcern>+</setconcern></think>Can you spell my name? I need to
        know that you can write it correctly</li>
</random> </template>
</affect>
</category>

<category> <affect name="*">
<that>MY NAME IS *</that>
<pattern>I HATE YOUR *</pattern>
<template><random>
    <li><think><setconcern>-</setconcern></think>I don't care, you can only call me,
        <bot name="name">.</li>
    <li><think><setconcern>-</setconcern></think>Why? A bad memory perhaps?</li>
    <li><think><setconcern>-</setconcern></think>That's a lovely name, why do you
        hate it?</li>
</random><think><setaffect>-</setaffect></think></template>
</affect>
</category>

<category> <affect name="-">
<that>MY NAME IS *</that>
<pattern>YOUR *</pattern>
<template><random>
    <li><think><setconcern>-</setconcern></think>Do you have problem with my
        <star/>?</li>
    <li><think><setconcern>#</setconcern></think>Mind that, what is yours?</li>
    <li><think><setconcern>-</setconcern></think>Don't be a witty!</li>
</random> </template>
</affect>
</category>
</topic>
...
```





Figure 5- 12 Example of a negative and positive situation type in my\_Eliza's memory structure

If there are more than one reassemble rules in Eliza's memory unit, we can use <li> tag to enumerate the entire reassemble rules and this enumeration is written inside <random> tag. <random> tag tells the system to select one of reply sentence randomly. <think> tag is used inside <template> tag to tell the system that whatever inside this tag must not be shown. Since the same pattern input may have different user's situation types, it means more variation of reply sentences with different system's situation types (see example in figure 5.12). With the same input pattern "YOUR \*", the two categories in the figure is indicated by different situation type of the user's affective state. Positive situation type stimulates the system to react in a positive sense also (see table 5-1 (a) below), but a negative situation type may stimulate negative reaction (see table 5-1 (b) below). Once the user's situation type shows one of three situation types: "+", "-", and "#", the memory structure with the same situation type will be checked first. If none of them is matches, the system picks a category with a normal situation type (\*).











Table 5- 1 Example of fragment in positive and negative situation type

User:	What is your name?	 My_Eliza
 My_Eliza:	My name is Eliza.	
User:	Your name is lovely!	 My_Eliza
 My_Eliza:	Your name is lovely also!	

(a) The above table is an example of fragment when the user is in a positive situation type

(b) The right table is an example of fragment when the user is in a negative situation type

User:	What is your name?	 My_Eliza
 My_Eliza:	My name is Eliza.	
User:	I hate your name.	 My_Eliza
 My_Eliza:	Why? Bad memory perhaps?	
User:	Your name reminds me to my enemy.	 My_Eliza
 My_Eliza:	Mind that, what is yours?	

When a user sends an input string, the system will have an access to the database of conversation history to get the current topic, last user's affective state, and system's previous reply. The second category displayed in figure 5.12 takes the input pattern "YOUR \*" with the topic is "NAME" and system's previous reply is "MY NAME IS \*". Taking one of user's string input in the figure, "Your name is lovely?" and using anaphoric analysis, the system already stored that current topic is "NAME" and the system's previous reply sentence is "MY NAME IS ELIZA". The system put "NAME IS LOVELY" in the star stack. The system will try to execute the best-matching operation against the topic, the system's previous reply and the input pattern using the algorithm in figure 5-13.

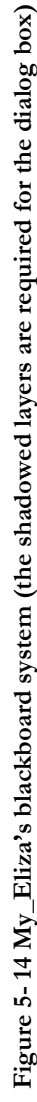
```
Find ATOMIC CATEGORY with the same TOPIC and THAT and AFFECT, or
Find ATOMIC CATEGORY with the same TOPIC and AFFECT, or
Find ATOMIC CATEGORY with the same AFFECT, or
Find DEFAULT CATEGORY with the same TOPIC and THAT and AFFECT, or
Find DEFAULT CATEGORY with the same TOPIC and AFFECT, or
Find DEFAULT CATEGORY with the same AFFECT, or
Find ATOMIC CATEGORY with the same THAT and AFFECT, or
Find ATOMIC CATEGORY, or
Find DEFAULT CATEGORY with the same THAT and AFFECT, or
Find DEFAULT CATEGORY
```

Figure 5- 13 Algorithm of my\_Eliza's pattern matching operation

Using that algorithm, in our example the system finally finds the best-fit category and selects one of its templates randomly: "Your <star/> also!". The system replaces <star/> tag with "name is lovely". Finally, the system sends the reply sentence to the user: "Your name is lovely also!".

### 5.3 Implementation of My\_Eliza's Dialog Box

After selection and arrangement of the approach for constructing reply sentence, the next step is to implement the dialog box. As mentioned above, the implementation of My\_Eliza's dialog box is based on the implementation of Program D A.L.I.C.E.





In order to build a robust basic layer to perform emotion recognition and facial display generation, my\_Eliza's dialog box must fulfill the requirements below (see the shadowed layer on figure 5-14):

1. The ability to parse a string both the user's string input or the system's reply sentence.
2. The ability to recognize the entire user's string input and to eliminate the user's typo, by the Lexical Analysis Layer.
3. The ability to generate system's reply sentence automatically based on current topic, system's previous reply and user's situation type, by Syntactic-Semantic Analysis Layer.
4. Ability to extract the emotive label on the memory structure to get the current user's situation type and system's situation type, by emotive labeled memory structure extraction layer.

Comparing the shadowed layer in figure 5.15 to figure 5.7, the first two requirements above and most of the third requirement have been fulfilled by Program D A.L.I.C.E.

This section will be organized as follows. First, the architecture of my\_Eliza's dialog box based on Program D A.L.I.C.E and my\_Eliza global design (described in chapter 4) is explained in section 5.3.1. Next, the design of the dialog box in UML – Unified Modeling Language, a tool to model the system in Object Oriented Paradigm, is described in section 5.3.2. This design includes context, use cases, object model and dynamic model of the system. Some examples of creating new a category in my\_Eliza list of pattern rules are presented in section 5.3.3. Finally, the programming process of the dialog box and testing the system that leads to the conclusion is summarized in section 5.3.4.

### 5.3.1 My\_Eliza's Dialog Box Architecture

Based on figure 5.7, figure 5.14, and the requirements above, figure 5.15 shows the implementation scope of my\_Eliza's dialog box. Since part of the implementation area is in conjunction with Program D, there are some changes in Program D especially those for the parsing process to interpret AIML and pattern-matching operation, which both deal with new extended tags: <affect> tag and <concern> tag.

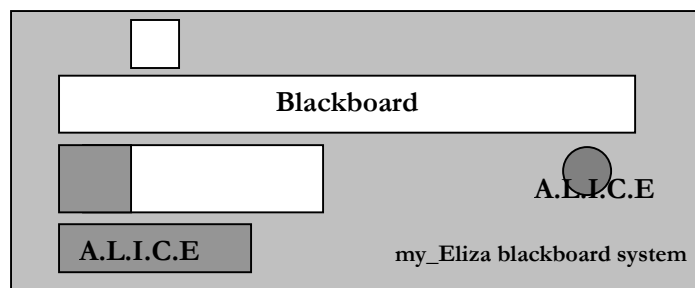


Figure 5- 15 My\_Eliza's dialog box implementation scope (white areas)

The flowchart in figure 5.16 below shows how the new version of Program D – A.L.I.C.E constructs a reply sentence that uses new two tags (comparing to figure A-4 (a) A.L.I.C.E's response construction). Italic words are the names that we use in my\_Eliza global design (compare to figure 4-3 and figure 4-4). The Syntactic-Semantic Analysis Layer in my\_Eliza includes Program D's routines, such as: get internal reply, get reply, pattern fit, get template, process response into XML, post process the response and push new data into stack. The usage of the blackboard system cannot be seen clearly in this flowchart diagram. However, as mentioned in my\_Eliza's global design chapter 4, in my\_Eliza system, the result of every layer goes to the blackboard. Therefore, the normalized string input as the result of the Lexical Analysis will be put on the blackboard. This works also for the Emotive Labeled Memory Structure Extraction Layer and the Syntactic-Semantic



Analysis Layer, entire results from both layers will be put on the blackboard. The parser will be executed every time the system needs to parse a string word by word. The conversation history database and the Syntactic-Semantic Analysis Layer have included two new tags in their operation.

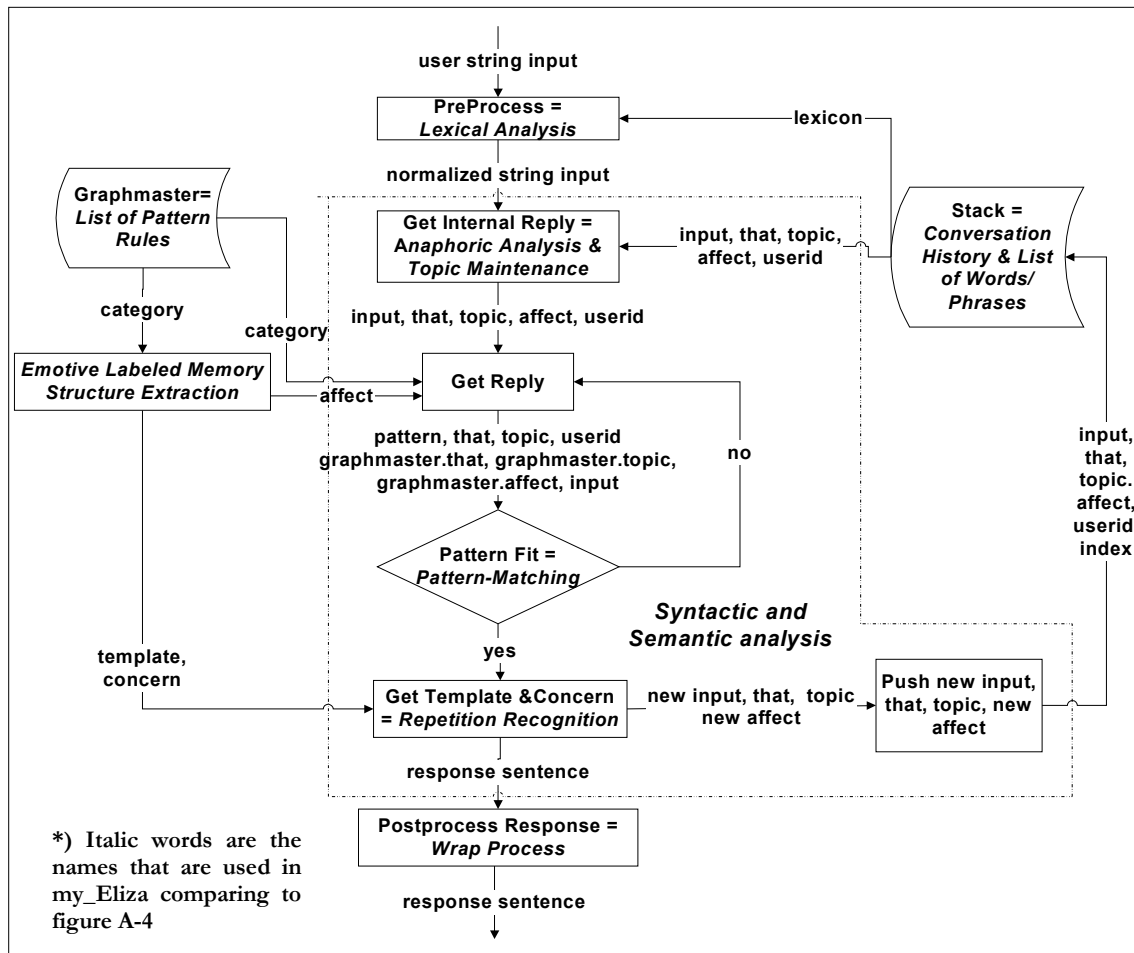


Figure 5- 16 New A.L.I.C.E = my\_Eliza's dialog box

### 5.3.2 My\_Eliza's Dialog Box Design

Since Program D A.L.I.C.E was implemented in Object Oriented Paradigm, we choose to use UML to design my\_Eliza's dialog box. First, the context of the dialog box is described using a context flow diagram. This diagram explains data flow and control flow in the system. The next is the use case diagram of the dialog box. This diagram explains all the tasks and actors, which are involved in the system. the object diagram describes the relationship between entities in the system. Finally, a dynamic model describes the events and transitions that happen between the entities in the system.

#### Context Model

There are three entities involved with the system, such as (see figure 5.17 below):

1. The user who communicates with the system by sending a text prompt. The dialog box will reply to the user also by sending the respond text prompt.
2. The list of pattern rules that provide systems memory structures in AIML form.



3. Conversation history database that stored system's previous reply, current topic, user's previous input, system's situation type and user's situation type. This database also contains list of words/phrases for normalizing a sentence that can be accessed during conversation.

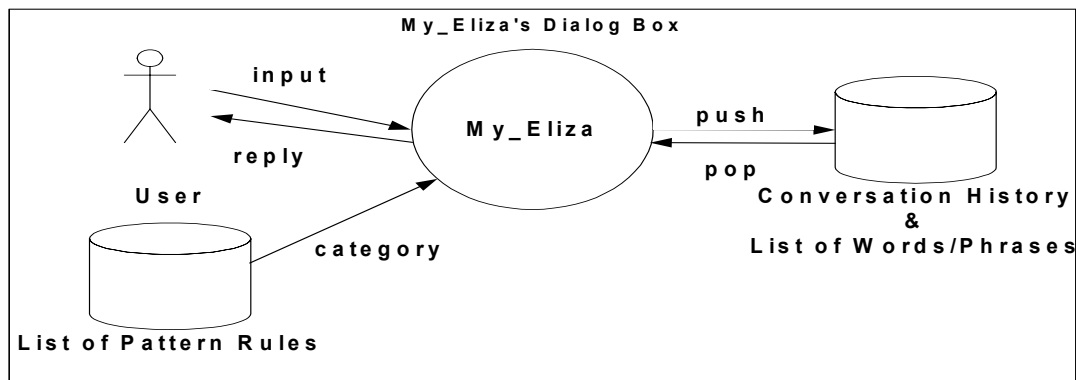


Figure 5- 17 Context flow diagram

## Use case

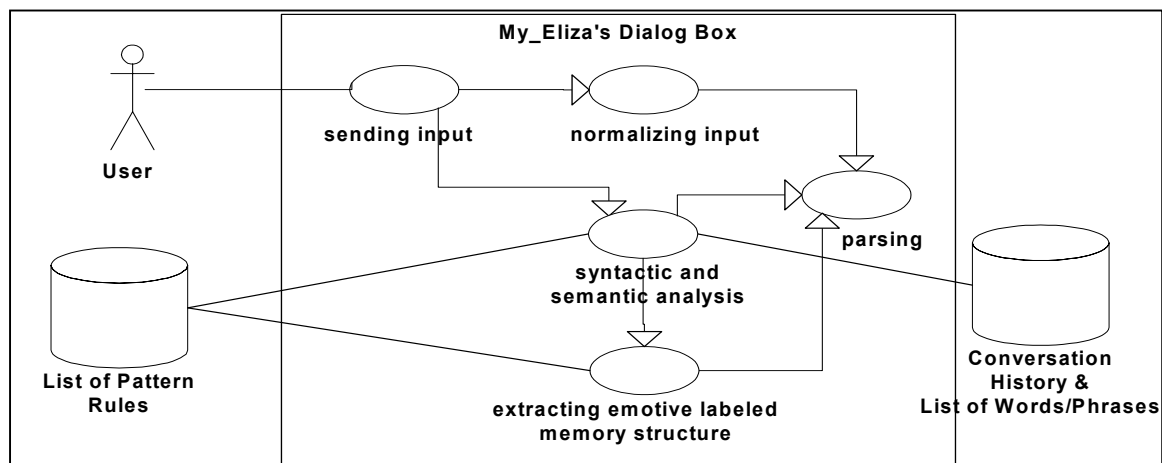


Figure 5- 18 Use case diagram

List of use cases:

1.	Name	:	Sending input
	Description	:	Sends input to the system
	Actors	:	The user
	Preconditions	:	-
	Exceptions	:	-
	Results	:	Use case normalizing input and syntactic and semantic analysis are active

2.	Name	:	Normalizing input
	Description	:	Recognizes input and corrects typo
	Actors	:	-
	Preconditions	:	The user has sent the input
	Exceptions	:	-
	Results	:	Input is normalized



3.	Name	:	Syntactic and Semantic analysis
	Description	:	Generates reply sentence
	Actors	:	List of pattern rule, conversation history database and list of words/phrases
	Preconditions	:	The input has been normalized
	Exceptions	:	-
	Results	:	Reply sentence is sent to the user

4.	Name	:	Extracting emotive labeled memory structure
	Description	:	Gets situation type inside <affect> tag and <concern> tag
	Actors	:	List of pattern rule
	Preconditions	:	Triggered by syntactic and semantic analysis when it analyzes the category from list of pattern rule
	Exceptions	:	-
	Results	:	Situation type value inside <affect> tag and <concern> tag

5.	Name	:	Parsing
	Description	:	Parses a string word by word
	Actors	:	-
	Preconditions	:	Triggered by almost all task
	Exceptions	:	-
	Results	:	Parsed string

## Object Model

Most of the object classes in my\_Eliza's dialog box use Program D's object classes. Figure 5-19 below shows the object diagram of my\_Eliza's dialog box. The white boxes indicate that those are not derived from Program D. Some boxes appear as a combination of white and gray, which means these boxes are derived from program D and there are some changes. The Program D's object classes description can be referred in appendix B. In this section, we only explain new object classes that are added to Program D's object classes, and they are:

1. *Blackboard system*, a shelter that used to be a place for entire message passing in the system since the system receives an string input until it generates a new reply
2. *Emotive Label extractor*, an object that used to provide the memory structure extraction result, i.e. two emotive labeled situation type value inside the <affect> tag and the <concern> tag

The data dictionaries of both new object classes are:

1. *Blackboard*  
Attribute (all private):
  - blackboard: Hashtables
  - + createNewMessage(String userid): boolean
  - + insertAffect(String userid, String affect): boolean
  - + insertConcern(String userid, String concern): boolean
  - + insertThat(String userid, String that): booleanMethods:
  - + insertTopic(String userid, String topic): boolean
  - + insertInput(String userid, String input): boolean
  - + insertReply(String userid, String reply): boolean
  - + insertThermometer(String userid, Thermometer thermometer): boolean
  - + insertUserAffectCandidate(String userid, String userAffect, boolean question): boolean
  - + insertSystemReactionCandidate(String userid, String systemReact, boolean question): boolean
  - + insertStimulusReaction(String userid, String stimulusReact): boolean
  - + insertCognitiveReactReaction(String userid, String cognitiveReact): boolean
  - + getAffect(String userid): String





... continues from Blackboard data dictionary

```
+ getConcern(String userid): String
+ getThat(String that): String
+ getTopic(String userid): String
+ getInput(String userid): String
+ getReply(String userid): String
+ getThermometer(String userid): Thermometer // added for prototype-1
+ getUserAffect(String userid): String
+ getSystemReact(String userid): String
+ getStimulusReact(String userid): String
+ getCognitiveReact(String userid): String
+ isUserQuest(String userid): boolean
+ isSystemQuest(String userid): boolean
```

## 2. *Emotive Label Extractor*

Attribute (all private):

```
- affect: String
- concern: String
```

Methods:

```
+ setAffect(String affect)
+ setConcern(String concern)
+ setAffect(String affect): String
+ setConcern(String concern): String
```

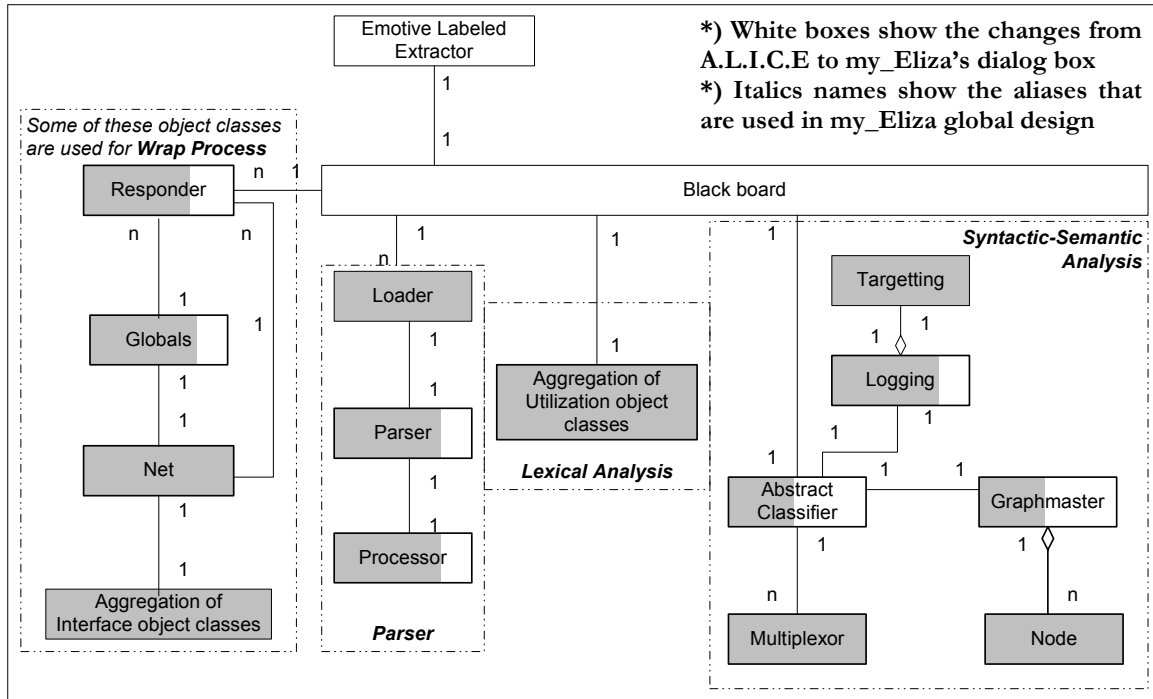


Figure 5- 19 Object diagram of my\_Eliza's dialog box

## Dynamic Model

Scenario of my\_Eliza's dialog box is divided into two parts, such as:

1. **Load time.** This time is running when the system is running at the first time. The system loads all rules of patterns into Graphmaster. Graphmaster object contains many nodes. It also loads list of words/phrases into Multiplexor, which can be accessed during conversation.
2. **Conversation time.** This time is running when the system receives user's string input via Responder object. The system will parse the input and normalize it. The normalized input is used to find the reply sentences through the pattern matching operation in aggregation of object classes of syntactic and semantic analysis. The normalized input and the result of the pattern matching operation are stored on the blackboard. The emotive label extractor extracts



two values (affect value and concern value) from the category that has been selected by pattern matching operation and they are stored on the blackboard too.

By the scenario above, we construct the event trace diagram in figure 5-20.

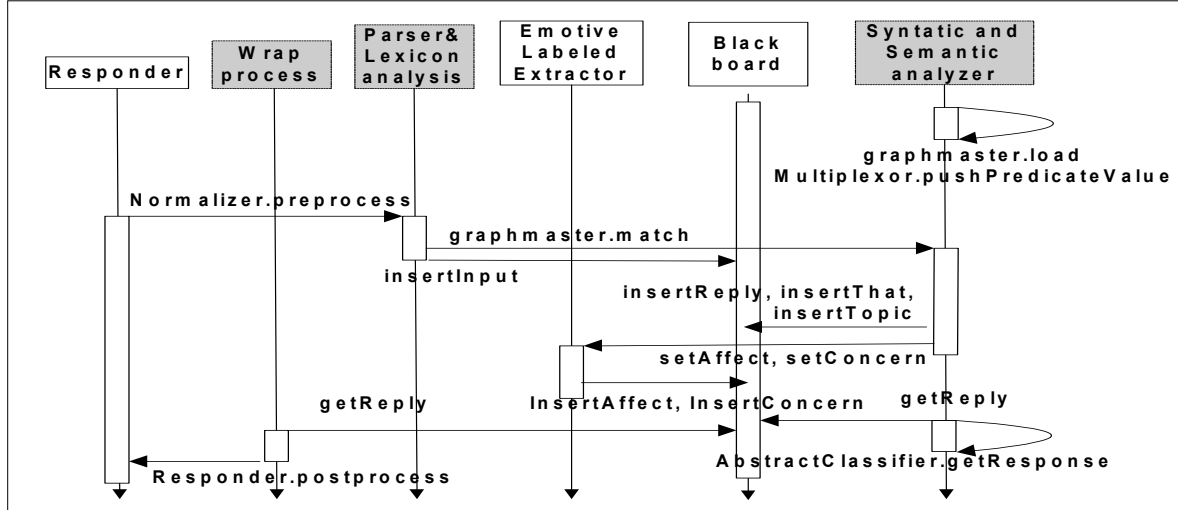


Figure 5- 20 Event trace diagram of my\_Eliza dialog box: loading and conversing

The shadowed boxes means those object classes are aggregate of some object classes of Program D A.L.I.C.E and we rename those aggregation based on the global design in figure 5-14. By the scenario above also, we construct transition diagrams of two new object classes in figure 5-21 and figure 5-22.

#### 1. Blackboard transition diagram

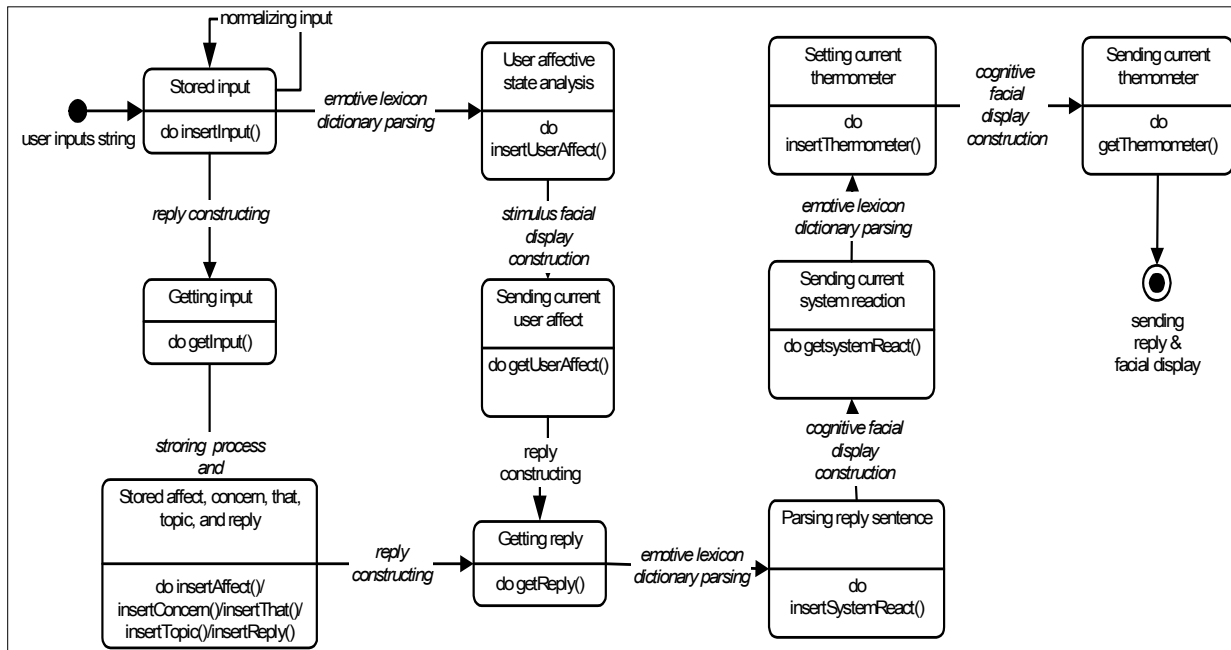


Figure 5- 21 Blackboard transition diagram



## 2. Emotive Labeled Extractor transition diagram

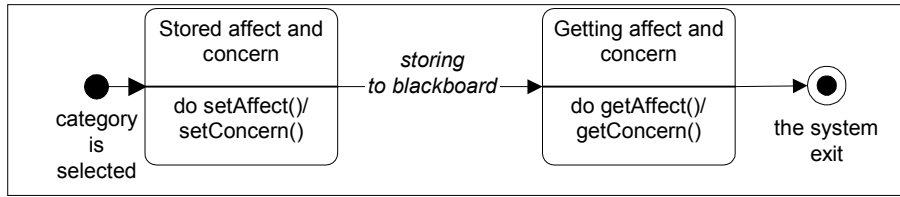


Figure 5- 22 Emotive Labeled Extractor transition diagram

### 5.3.3 Build the List of Pattern Rules

The simplest way to build my\_Eliza's list of pattern rules is by using dialog analysis. In figure 5-23 and figure 5-24 below, we can see step by step the construction process of a category.

```
Example dialog:
User   : Do you know anything about computers?
My_Eliza: Sure. Why do you want to know about it?
User   : My computer crashed last night.
My_Eliza: I am sorry. Is it a PC or laptop?
User   : It is a laptop.

Transformation into AIML form:
<topic name="*">
<category> <affect name="*"> <that>*</that>
<pattern>DO YOU KNOW ANYTHING ABOUT *</pattern>
<template>
  <think><setconcern>+</setconcern><settopic><star/></settopic></set></think>
  Sure, why do you want to know about <set_it><star/></set_it>?
</template>
</affect> </category>
</topic>

<topic name="COMPUTERS">
<category> <affect name="*"> <that>WHY *</that>
<pattern>MY COMPUTER CRASHED *</pattern>
<template>
  <think><setconcern>+</setconcern><setaffect>-</setaffect></think>
  I am sorry. Is it a PC or laptop?
</template>
</affect></category>
</topic>
```

Figure 5- 23 List of pattern rules construction using dialog analysis

This process is similar to constructing a category from Weizenbaum's Eliza script (see figure 5-9). Please refer to appendix A and [BUS01] to get detailed information about tags in AIML. One way to collect dialogs is by joining a lot of conversation in chat-rooms. Usually it works by taking two dialog states with two user's lines and two my\_Eliza' lines. This is necessary to see how the system's previous reply sentence influences the conversation. Two additional steps are determining user's situation type (affect value) and system's situation type (concern value). Most of the time, we can use assumption to figure user's affective state when he/she feeds the string input and how he/she reacts after he/she reads my\_Eliza's reply sentence. To determine my\_Eliza's situation type, we have to take into account the system's role as psychoanalyst. The heuristic approach to set the value of situation type (both the user and the system) please refer to section 4.2.1.2.

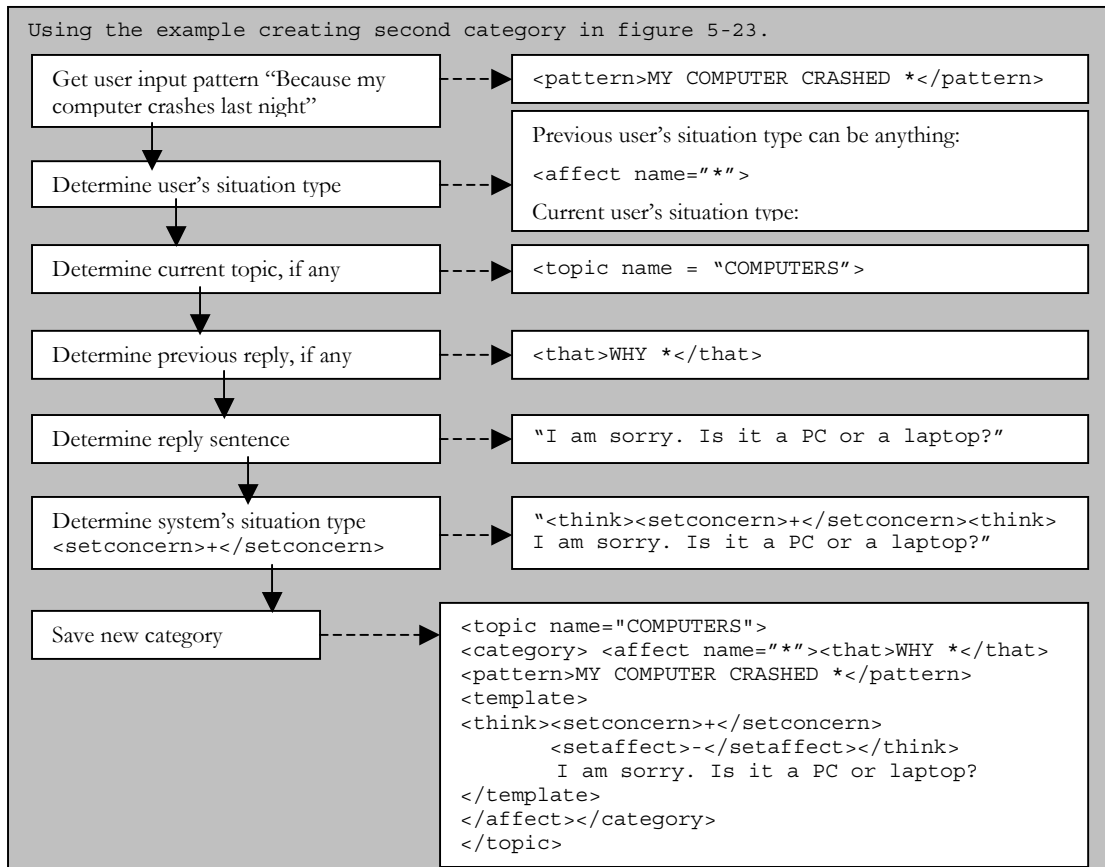


Figure 5- 24 Constructing a category from a dialog

In table 5-2 below, there are some examples of creating a category from dialog analysis. The comments and assumption in constructing process can be seen inside the <!--...--> tag.

Table 5- 2 Example of categories construction based on a dialog analysis

```
User: Hallo. My name is Andy
My_Eliza: Hallo. Andy, nice to meet you.

<!-- in greeting there is not a particular topic to discuss,
      assumptions: user is in a normal situation type and
                  my_Eliza is in a positive situation type.
      ----->
<topic name="*">
<category> <affect name="*"> <that>*/that>
<pattern>HALLO</pattern>
<template> <think><setconcern>+</setconcern></think>Hallo.</template>
</affect></category>
</topic>

<!-- The system will process the string input per sentence.
      The user's string input above consists of two sentences.
      Using <set name="name"> tag the system stores the name of the user.
      ----->
<topic name="*">
<category> <affect name="*"> <that>*/that>
<pattern>MY NAME IS */pattern>
<template><think><setconcern>+</setconcern></think>
      <set name="name"><star/></set>, nice to meet you.
</template>
</affect></category>
</topic>
```



User: Nice to meet you too.

My\_Eliza: Enough! Now, what is your problem?

```
<!-- There is still not particular topic,
      assumptions: user is in a normal situation type and
                  my_Eliza is in a positive situation type.
      Using <li> tag, the system can randomly choose the reply sentence.
      Using <get name="name"> the system refers to the user's name.
      ----->
<topic name="*">
<category> <affect name="*"> <that>NICE TO MEET YOU</that>
<pattern>NICE TO MEET YOU *</pattern>
<template><random>
<li><think><setconcern>+</setconcern></think>
      What a polite of you. Tell me about yourself</li>
<li><think><setconcern>+</setconcern></think>
      Enough! Now, what is your problem?</li>
<li><think><setconcern>+</setconcern></think>
      What's the news, <get name="name">?</li>
</random></template>
</affect></category>
</topic>
```

User: I am in trouble now.

My\_Eliza: What's wrong? How can I help you?

```
<!-- There is still not particular topic.
      assumptions: user is in a negative situation type and
                  my_Eliza is in a negative situation type
      Using <setaffect> tag in the TSAE, the system sets current user's situation type.
      This category does not care about the system's previous reply.
      ----->
<topic name="*">
<category> <affect name="*"> <that>*</that>
<pattern>I AM IN TROUBLE *</pattern>
<template><think><setconcern>-</setconcern><setaffect>-<setaffect></think>
      What's wrong? How can I help you?
</template>
</affect></category>
</topic>
```

User: Do you know anything about computers?

My\_Eliza: Sure. Why do you want to know about it?

User: My computer crashed last night

My\_Eliza: I am sorry. Is it a PC or laptop?

```
<!-- There is still not a particular topic.
      Assumptions: user is in a negative situation type and,
                  my_Eliza is in a positive situation type
      Using (*) value in affect tag, this category can be used in every type of situation.
      This category do not care about the system's previous reply.
      Using <star/> tag, the system refers to "computers" at the same time this category
      can be use to other user's string input that has the same pattern.
      Using <settopic> tag, the system sets current conversation topic.
      Using <set_it> tag, the system transforms whatever the value of <star/> tag is
      into "it" and store the value in the system.
      ----->
<topic name="*">
<category> <affect name="*"> <that>*</that>
<pattern>DO YOU KNOW ANYTHING ABOUT *</pattern>
<template>
      <think><setconcern>+</setconcern><settopic><star/></settopic></think>
      Sure. Why do you want to know about <set_it><star/></set_it>?
</template>
</affect> </category>
</topic>
```



<pre>&lt;!-- The conversation is in "computers" topic.       Assumptions: user is in a negative situation type and,                    my_Eliza is in a positive situation type,                    (sorry-for the user's feeling is a positive situation type) -----&gt; &lt;topic name="COMPUTERS"&gt; &lt;category&gt; &lt;affect name="-"&gt; &lt;that&gt;WHY *&lt;/that&gt; &lt;pattern&gt;MY COMPUTER CRASHED *&lt;/pattern&gt; &lt;template&gt;     &lt;think&gt;&lt;setconcern&gt;+&lt;/setconcern&gt;&lt;/think&gt;     I am sorry. Is it a PC or laptop? &lt;/template&gt; &lt;/affect&gt;&lt;/category&gt; &lt;/topic&gt;</pre>
<p><b>User: It is a PC</b> <b>My_Eliza: I had a PC before, have you ever formatted it?</b></p> <pre>&lt;!-- The conversation is still in computers topic.       Assumptions: user is in a negative situation type and,                    my_Eliza is in a positive situation type. -----&gt; &lt;topic name="COMPUTERS"&gt; &lt;category&gt; &lt;affect name="-"&gt; &lt;that&gt;IS IT A PC OR LAPTOP&lt;/that&gt; &lt;pattern&gt;IT IS A PC&lt;/pattern&gt; &lt;template&gt;     &lt;think&gt;&lt;setconcern&gt;+&lt;/setconcern&gt;&lt;/think&gt;     I had a PC before, have you ever formatted it? &lt;/template&gt; &lt;/affect&gt; &lt;/category&gt;  &lt;!-- If the user talks about his/her laptop, instead of a PC. -----&gt; &lt;category&gt; &lt;affect name="-"&gt; &lt;that&gt;IS IT A PC OR LAPTOP&lt;/that&gt; &lt;pattern&gt;IT IS A LAPTOP&lt;/pattern&gt; &lt;template&gt;     &lt;think&gt;&lt;setconcern&gt;+&lt;/setconcern&gt;&lt;/think&gt;I got a laptop two years ago. We have to be     more careful with &lt;set_it&gt;laptop&lt;set_it&gt;. I think you better call     computer reparation service. &lt;/template&gt; &lt;/affect&gt;&lt;/category&gt; &lt;/topic&gt;</pre>
<p><b>User: I will try it.</b> <b>My_Eliza: Good luck then. Hopefully you success with your PC.</b></p> <pre>&lt;!-- The conversation is still in computers topic.       Assumptions: user is in a negative situation type and,                    my_Eliza is in a positive situation type.       This category will work for any the system's previous reply, any topic of       Conversation and/or any user's situation type.       Using &lt;get_it&gt; tag my_Eliza will refer variable that substitutes with "it"       in the system. -----&gt; &lt;topic name="*"&gt; &lt;category&gt; &lt;affect name="*"&gt; &lt;that&gt;*&lt;/that&gt; &lt;pattern&gt;I WILL TRY *&lt;/pattern&gt; &lt;template&gt;     &lt;think&gt;&lt;setconcern&gt;+&lt;/setconcern&gt;&lt;/think&gt;     Good luck then. Hopefully you success with your &lt;get_it/&gt;. &lt;/template&gt; &lt;/affect&gt; &lt;/category&gt; &lt;/topic&gt;</pre> <p>...</p>





### 5.3.4 My\_Eliza's Dialog Box Implementation, Test and Result

Program D A.L.I.C.E is written using Java Development Kit version 1.3 and XML, therefore we use a compiler and classes contained in the same languages for my\_Eliza prototype. The way how to compile the prototype is described in Program D A.L.I.C.E explanation in appendix B. My\_Eliza's dialog box contains many packages most of them derive from Program D A.L.I.C.E. Since we use all modules of Program D A.L.I.C.E, my\_Eliza's dialog box is made as packages with the same directories division as Program D A.L.I.C.E: `org.Alicebot.server`. Two new objects above (*blackboard* and *emotive labeled extractor*) are collected in package `org.Alicebot.server.blackboard` and `org.Alicebot.server.nonverbal`. The package of `org.Alicebot.server.blackboard` contains only one Java file: `blackboard.java`. The package of `org.Alicebot.server.blackboard` contains these Java files:

- **NonverbalConstant.java.** An object class that contains constant variables that are referred by all object classes in this package.
- **EmotiveLabelExtractor.java.** An object class that contains properties and methods to extract the value of <affect> tag and <concern> tag.

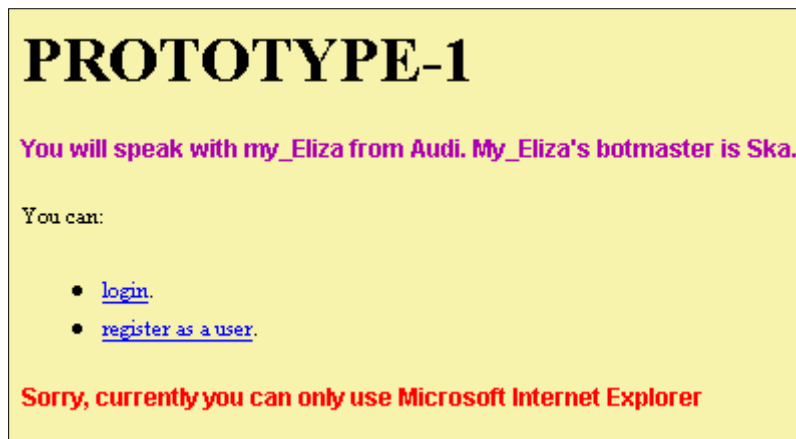


Figure 5- 25 Index page of my\_Eliza's prototype-1

When the user requests to my\_Eliza's URL address the figure 5-25 will appear. The page gives two choices to the user. If the user wants to converse with Eliza, he/she should register with a user id and a password. The user can either log in or register as a new user (see figure 5.26) The program can only work properly if the user uses Internet Explorer from Microsoft as a browser.

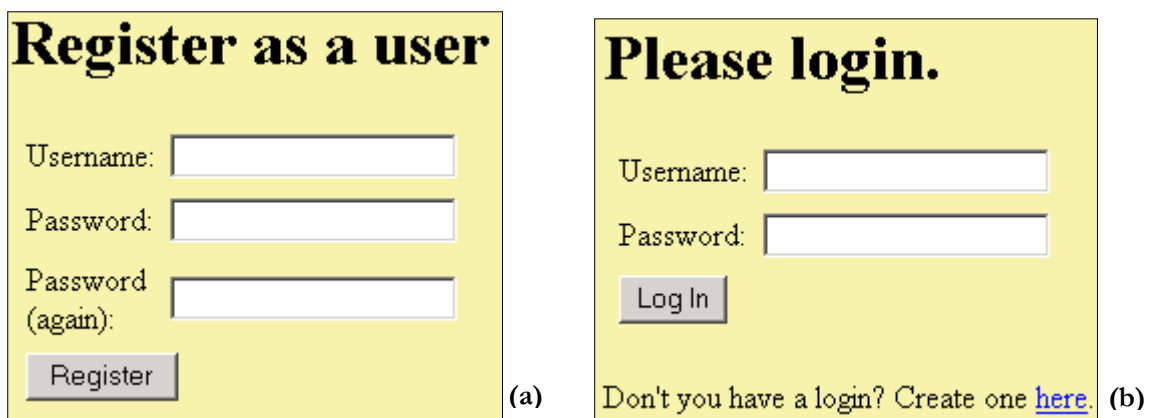


Figure 5- 26 My\_Eliza's dialog box pages: (a) register a new user and (b) login



The main page of my\_Eliza dialog box is figure 5-27. The user can type their string input on a white text box and he/she can enter it by click button “Say” or type *enter*. The input will appear on the line “You said” and the system’s reply to the user’s input will appear on the line “my\_Eliza said”. When the system just starts, the user line default is **CONNECT** and my\_Eliza will answer one of the lines below:

1. Hallo <user-name>, what a lovely day, how are you today?
2. It is nice to meet you, tell me what is the news?
3. Hi, my name is my\_Eliza, I am here to help you.
4. How are you today, <user-name>?
5. Is everything all right, <user-name>?
6. How do you do. Please state your problem.
7. What a lovely day, do you agree?
8. Hallo <user-name>, what can I do for you?

After this step, the system will wait for the user’s string input signed by clicking button “Say” or typing *enter*. The <user-name> will substitute by the user’s name, if the system does not store it uses default name: “undefined”.

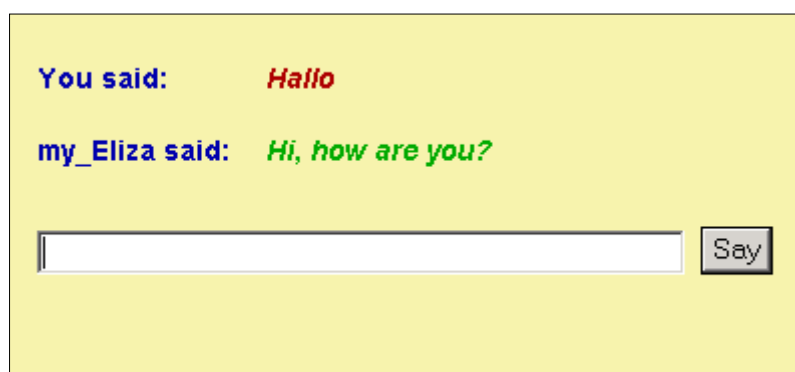


Figure 5- 27 My\_Eliza's dialog box main page chat.html

Table 5-3 below shows one of the examples of the test that we have conducted. The input is the user lines and the result is my\_Eliza’s reply sentences. My\_Eliza can remember the user’s name, for example in line number 3. The user tells the system her name and after that it can call her name in the dialog (see line number 10, 24 and 52).

Table 5- 3 Tests and results of my\_Eliza's dialog box

Line#	Dialog state	
1.	User:	CONNECT
2.	My_Eliza:	It is nice to meet you, tell me what is the news?

The user tells my\_Eliza his/her name. My\_Eliza will store it in the system:

Line#	Dialog state	
3.	User:	My name is Siska. What is your name?
4.	My_Eliza:	Let me write it first, Siska, hmm nice name. My name is my_Eliza.
5.	User:	Hallo my_Eliza!
6.	My_Eliza:	Enough! Now, what is your problem?
7.	User:	I do not have any problem at this moment.

The system use <get> tag to refer to the user’s name:

Line#	Dialog state	
8.	My_Eliza:	Good for you, Siska.
9.	User:	Do you have any gossip to share?



The system still does not have a topic about "gossip", therefore it picks this reply sentence:

Line#	Dialog state	
10.	My_Eliza:	If you continue like this I'll simply delete your autoexec.bat!
11.	User:	Are you joking? Because it's not funny.
12.	My_Eliza:	Do you think you are smart? You really think so....
13.	User:	Off course, I am cleverer than you.
14.	My_Eliza:	Siska, do you believe it is normal to be cleverer than me?
15.	User:	I do not like this conversation.
16.	My_Eliza:	It's just too bad you don't like this conversation.
17.	User:	Do you have anything better to say?
18.	My_Eliza:	What does that suggest to you?
19.	User:	Interesting story perhaps.
20.	My_Eliza:	What do you mean?
21.	User:	I am looking for a friend to talk with. Can you do that?
22.	My_Eliza:	Siska, do you believe it is normal to be looking for a friend to talk with? If I could, what good will it make?
23.	User:	You are not a good companion.
24.	My_Eliza:	What gives you the idea?
25.	User:	I am bored.
26.	My_Eliza:	True.
27.	User:	Ha ha. You are very stupid.
28.	My_Eliza:	Whatever. What causes you to think such thing about me?
29.	User:	Cause it is true, isn't it?
30.	My_Eliza:	That is quite interesting. Open your mind a little more.
31.	User:	I do not get any value when I talk with you. It is wasting my time.
32.	My_Eliza:	Do you wish to be able to get any value when you talk with me? You don't have to explain it.
33.	User:	Who made you?
34.	My_Eliza:	Maybe it says in the help file....
35.	User:	You don't know, do you?
36.	My_Eliza:	Whatever. No response, next question.
37.	User:	Are you angry?
38.	My_Eliza:	You must mean someone else.
39.	User:	Don't be mad, please. I just feel bored.
40.	My_Eliza:	I am not! Could be so.
41.	User:	Do you know anything about computers?

My\_Eliza do not have any topic about computer, therefore it picks this reply sentence:

Line#	Dialog state	
42.	My_Eliza:	You really hold cheap of me.
43.	User:	It is not what I mean. I have some problem with my computer.
44.	My_Eliza:	But I think it is what I mean. You do? How did you get it?
45.	User:	I do not have any idea.
46.	My_Eliza:	I need you to be surer.
47.	User:	I thought you could help me with it.
48.	My_Eliza:	Do you really think so now?
49.	User:	I guess not.
50.	My_Eliza:	Siska, why do you say you guess not?
51.	User:	You cannot help me to fix my computer.
52.	My_Eliza:	But it does not interest me.
53.	User:	Is it your job to help people?
54.	My_Eliza:	I don't know if my job to help people is, what do you hope for? Tell me about your family.
55.	User:	I need to leave now, sorry.
56.	My_Eliza:	Let's forget it for now, all right?
57.	User:	I have to go really.
58.	My_Eliza:	Why do you want to quit?
59.	...	...

It is possible to get misunderstanding or ambiguous conversation especially if the system does not store such a conversation topic, however this problem can be solved easily by adding new memory unit (categories) in my\_Eliza's list of pattern rules while the conversation runs. The system provides an AIML *watcher* that is always active and monitors if there is new addition of the memory unit. As mentioned in chapter 4, the bigger the memory structure the more realistic the user can feel. An example of dialog on computer topic using the categories that are displayed in table 5-2 can be seen



in table 5-4. This table's number starts from 42 to show that if line number 42 in table 5-3 changed, it will change the rest of the conversation.

Table 5- 4 Example of a conversation between user and my\_Eliza in a topic about computer.

Line#	Dialog state	
42.	My_Eliza:	Sure, why do you want to know it?
43.	User:	My computer crashes last night.
44.	My_Eliza:	I am sorry. Is it a PC or laptop?
45.	User:	It is a laptop.
46.	My_Eliza:	I got a laptop two years ago. We have to be more careful with it. I think you better call computer reparation service.
47.	User:	I will try it.
48.	My_Eliza:	Good luck then. Hopefully you success with your laptop.
49.	User:	Thanks.
50.	...	...

As conclusion, my\_Eliza's dialog box as preprocessing of automatic emotion recognition and facial display generation system has provided a robust human-system interaction using typed natural language. The future work that needs to be done is to continue expanding the system's memory structure with more variation of topics and reply sentences from a lot of dialogs analyses.



## Chapter 6 My\_Eliza prototype-1: Stimulus-Response Based Reasoning

My\_Eliza prototype-1 is the second step in building a multimodal communication system (see three major (incrementally) implementation layers in figure 4-1). This prototype does not only contain my\_Eliza dialog box (which processes natural language and constructs reply sentences automatically when the user sends string input), but also provides an automatic emotion recognition and facial display generation based on stimulus response of the system. Stimulus-response in this prototype means my\_Eliza responses only to a single prompt of the user's string input. The emotion recognition only process current string input and the facial display response is not influenced by the anaphora of conversation.

This chapter is organized as follows. First, the development of my\_Eliza from the dialog box to the prototype-1 is summarized in section 6.1. The next is, the design of my\_Eliza prototype-1 in UML is presented in 6.2. This design includes context, use cases, object model, dynamic model of the system and the design of system's affective knowledge base. Finally, the implementation of my\_Eliza prototype-1 is described in section 6.3. The testing phase of this prototype will be explained in chapter 7.

### 6.1 From My\_Eliza's Dialog Box to My\_Eliza prototype-1

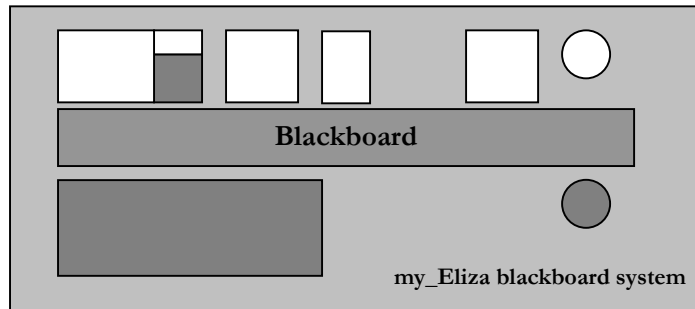
After implementation of the dialog box and establishing a list of pattern rules as the system's "brain", the next step is to implement an automatic emotion recognition and facial display generation. Since we develop my\_Eliza as an incrementally approach, we limit the system's emotion recognition ability only to the process of a single prompt of user's string input and facial display generation is based on stimulus response. However, the development of this implementation layer has to provide a robust layer to perform emotion recognition and facial display generation for the next release.

Therefore, my\_Eliza's prototype-1 must fulfill the requirements below (see the shadowed layer in figure 6-1):

1. A robust dialog box.
2. An expandable emotive lexicon dictionary and a robust parser that has the ability to recognize emotive lexicon from the string and entitled them to Ekman's seven universal emotion types. The Affective State Analysis Layer performs the process of emotive label extractor and parsing the string against the emotive lexicon dictionary.
3. An extensive Affective Knowledge Base that contains system's emotion recognition knowledge and a knowledge-based system that reasons during the recognition process.
4. A robust emotion intensity analyzer that is always active to monitor the flow of the system's affective state during conversation using its thermometers.
5. The robust facial display selector that selects two facial displays in every dialog state. It uses the result of Concern of the other Analysis Layer to select the first reaction facial display (as reaction to the user's string input) and the result of the Cognitive Reasoning Layer to select the second reaction facial display (to convey the reply sentence of the system).

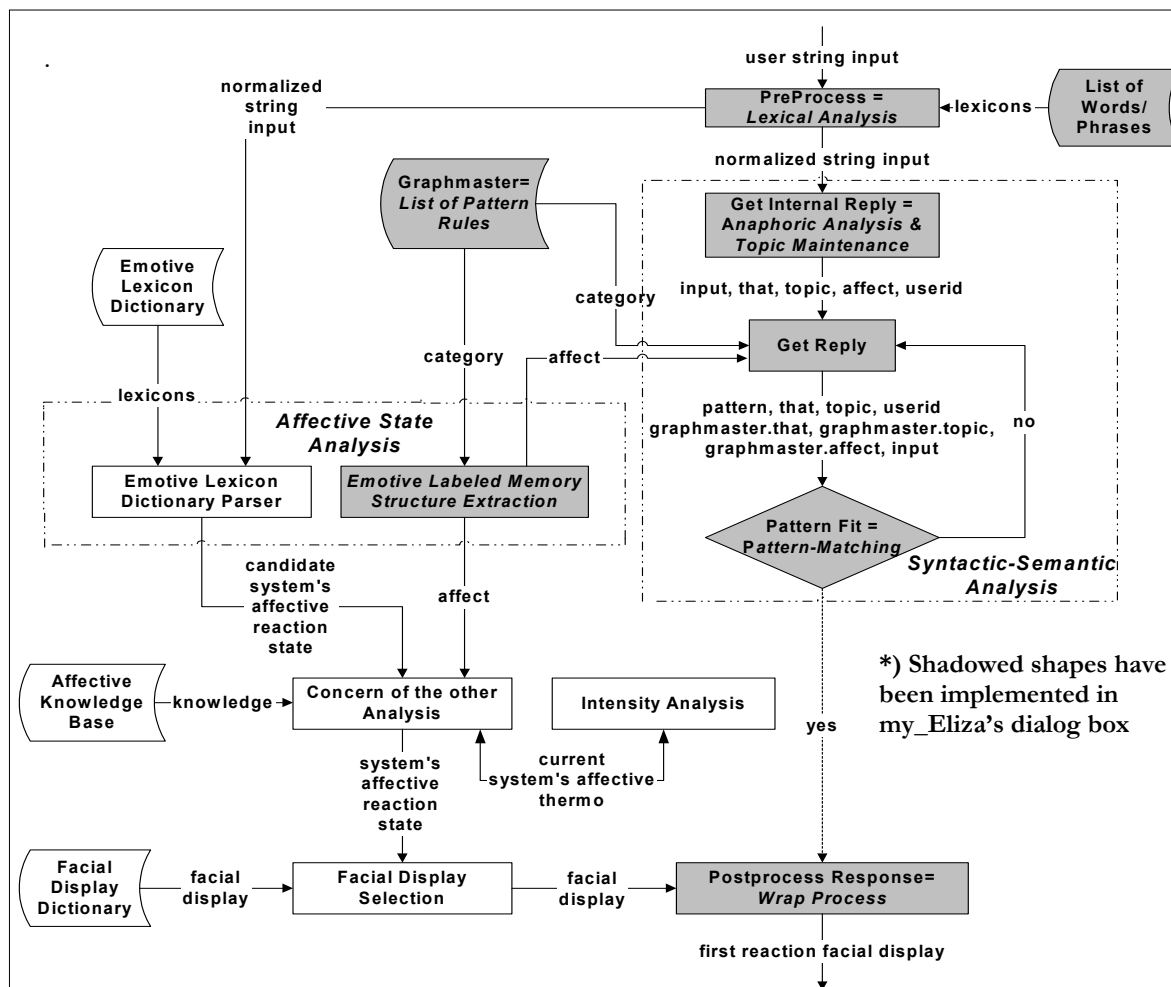
Comparing the shadowed layer in figure 6.1 to figure 5.14, the first requirements above have been fulfilled by My\_Eliza's dialog box. Based on figure 6.1 and the requirements above, figure 6.2 shows the implementation scope of my\_Eliza's prototype-1.





\*) Shaded shapes have been developed in my\_Eliza's dialog box

Figure 6- 2 My\_Eliza's prototype-1 implementation scope (white areas)



\*) Shaded shapes have been implemented in my\_Eliza's dialog box

Figure 6- 3 My\_Eliza's prototype-1 flowchart to construct the first reaction facial display

The flowchart in figure 6.3 above and figure 6.4 describe how my\_Eliza prototype-1 recognizes emotions and generates facial displays (comparing to figure 5.15). As described in chapter 4, the prototype also constructs two facial displays: a first reaction facial display (called also stimulus response facial display) and a facial display to convey the reply sentence (called also cognitive reasoning based facial display). When the user enters the string input, my\_Eliza shows the first reaction facial display. The process generating this facial display actually is almost in the same time as the system constructs a reply sentence and generates a facial display to convey the reply. Facial display selection layer using the data on the blackboard selects the first reaction facial display. The





concern of the other analysis layer derives that data by conducting emotion recognition of the user's string input and processing three data types: (1) the affect value that can be derived from emotive labeled memory structure extraction layer as user's situation type, (2) the highest degree of the six thermometers that monitors system's affective state during the conversation, and (3) the candidate of system's affective reaction that can be derived by emotive lexicon dictionary parser from the user's string input.

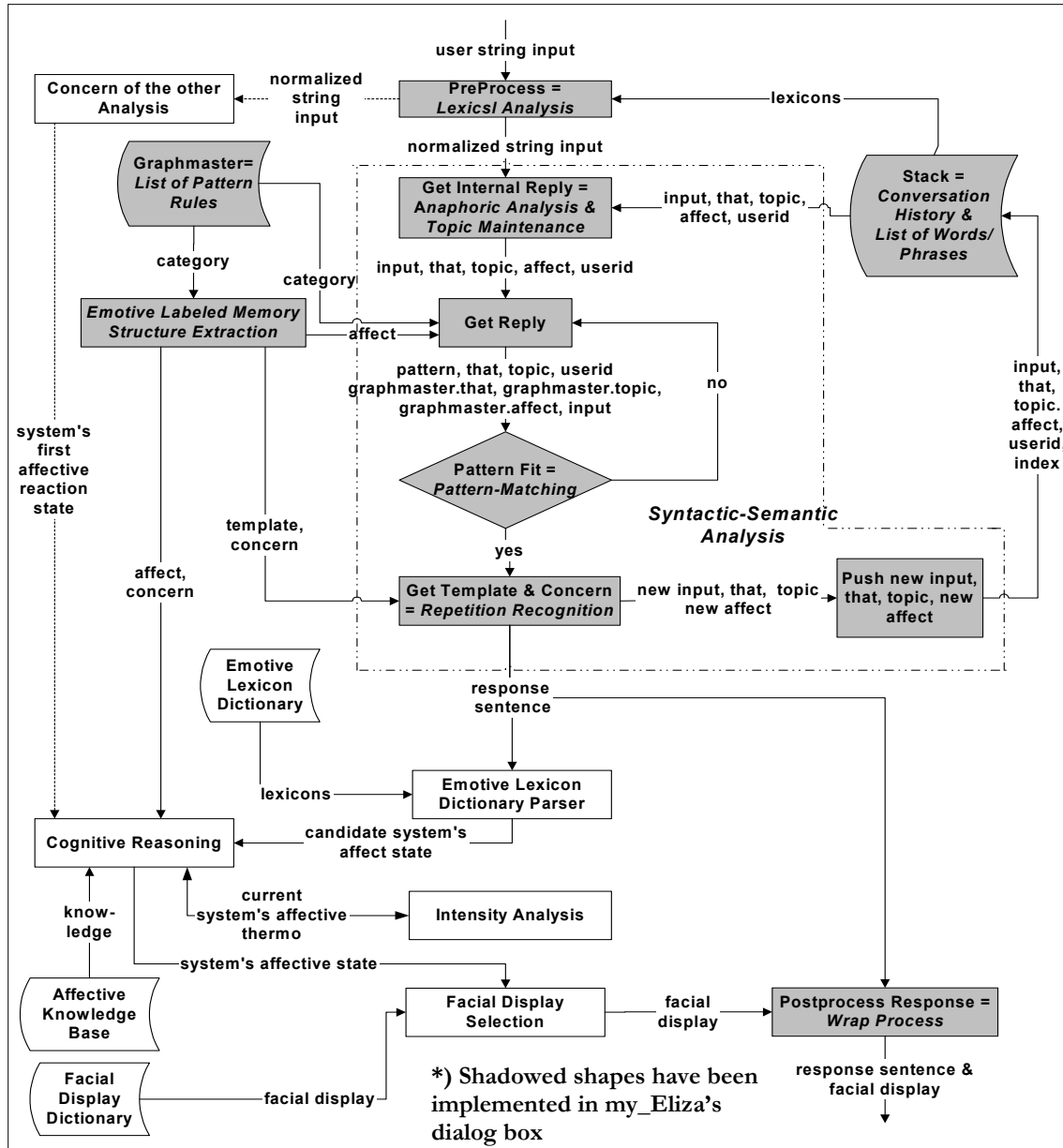


Figure 6- 4 My\_Eliza's prototype-1 flowchart to construct a corresponding facial display to a reply

Using the data on the blackboard, facial display selection layer also selects the facial display to convey system's reply sentence. Turn taking of this point, the cognitive reasoning layer derives that data by conducting emotion recognition of the system's reply sentence and processing three data types: (1) the affect value as user's situation type and (2) the concern value as system's situation type that can be derived from emotive labeled memory structure extraction layer, (3) the intensity of system's affective state that can be monitored by intensity analysis layer during the conversation, and (4) the candidate of system's reaction affective state that can be derived by emotive lexicon



dictionary parser layer. The emotive lexicon dictionary parser layer (in this phase) parses the system's reply sentence to derive the candidate of system's reaction affective state. This prototype uses two additional data stores, such as: (1) an emotive lexicon dictionary, and (2) a facial display dictionary, and one knowledge base, namely an affective knowledge base.

## 6.2 Design of My\_Eliza's Prototype-1

The design of my\_Eliza prototype-1 also uses UML. First, the new context diagram of my\_Eliza is explained in section 6.2.1. Next, the use case diagram of the prototype is described in section 6.2.2. This diagram explains all new tasks and new actors that are involved in the system comparing to my\_Eliza's dialog box. The relationship between entities in the system is described by object diagram in section 6.2.3. The event transitions, which happen between the entities in the prototype, are described by a dynamic model and summarized in section 6.2.4. Finally, the design of the affective knowledge base is explained in section 6.2.5.

### 6.2.1 Context Model

There are three new entities involved with the system such as: (see figure 6-5 below):

1. The emotive lexicon dictionary that contains a list of words or phrases showing the emotion in a sentence.
2. The facial display dictionary that contains a list of my\_Eliza facial displays one to one corresponding to seven Ekman's universal emotion types plus an uncertain facial display.
3. The affective knowledge base that contain rules in IF-THEN form used in emotion reasoning.

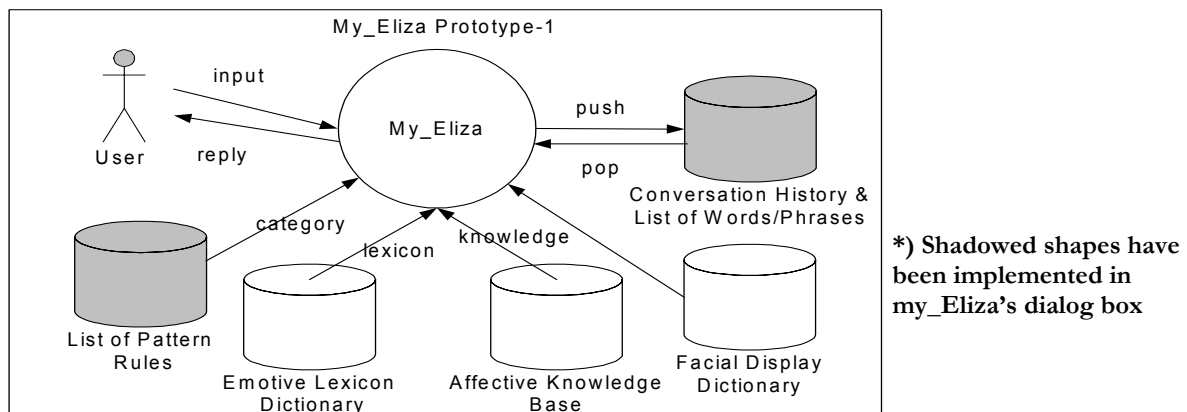


Figure 6- 5 My\_Eliza prototype-1's context flow diagram

### 6.2.2 Use case

Based on figure 6-6 below, list of new use cases are:

1.	Name	:	Concern of the other analysis.
	Description	:	Determines system's reaction affective state as a first reaction to user's string input.
	Actors	:	Affective knowledge base.
	Preconditions	:	The affect value and candidate of system's reaction affective state are known. The system's affective thermometer is active.
	Exceptions	:	-
	Results	:	One of twenty-six emotion types of system's reaction affective states is determined and sent to the blackboard.



2.	Name	:	Cognitive reasoning.
	Description	:	Determines system's reaction affective state to convey system's reply sentence.
	Actors	:	Affective knowledge base.
	Preconditions	:	The reply sentence has been generated. The affect value, the concern value and the candidate of system's reaction affective state are known. The system's affective thermometer is active.
	Exceptions	:	-
	Results	:	One of twenty-six emotion types of system's reaction affective states is determined and sent to the blackboard.

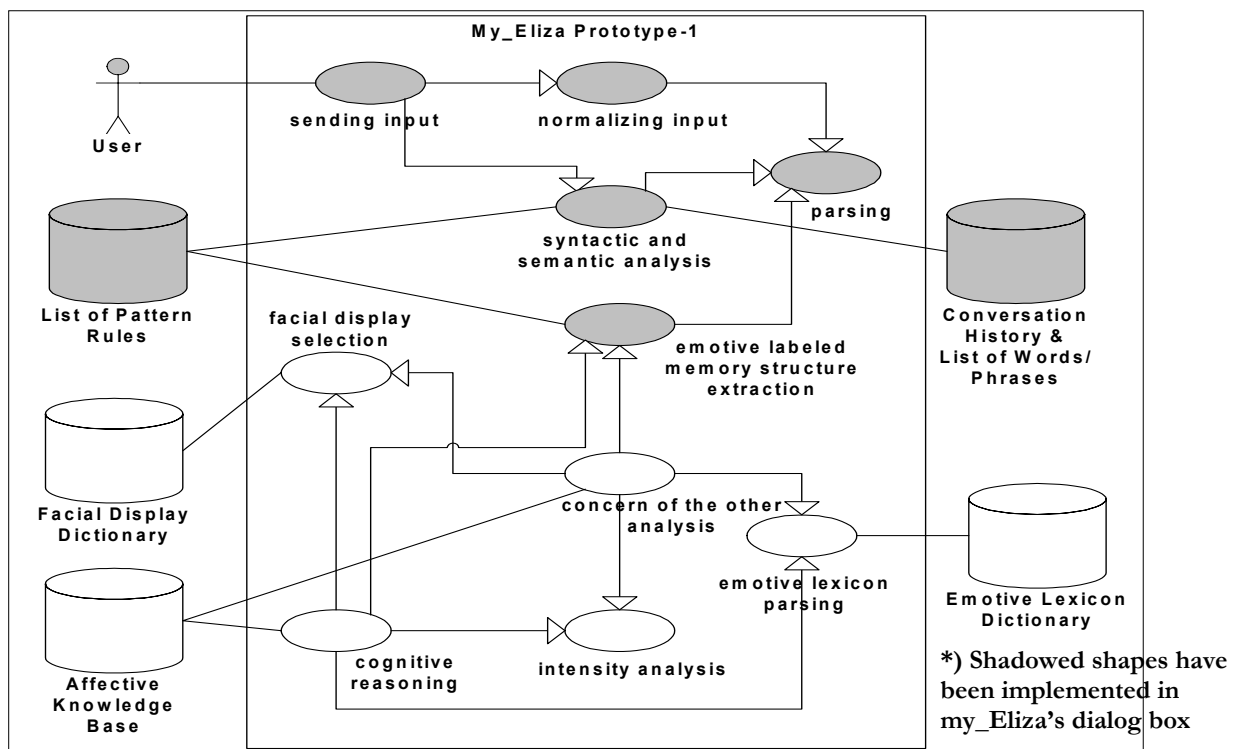


Figure 6- 6 My\_Eliza prototype-1's use case diagram

3.	Name	:	Emotive lexicon parsing.
	Description	:	Determines two candidates of system's reaction affective states: as reaction to the user's string input and to convey system's reply sentence, using shallow parsing against emotive lexicon dictionary.
	Actors	:	Emotive lexicon dictionary.
	Preconditions	:	Input has been normalized or the reply sentence has been generated.
	Exceptions	:	-
	Results	:	Two of twenty-six emotion types of the candidate of system's reaction affective states are determined and sent to the blackboard.



4.	Name	:	Intensity analysis.
	Description	:	Monitors system emotion level based on system's reaction affective states to convey the reply sentence.
	Actors	:	-
	Preconditions	:	-
	Exceptions	:	-
	Results	:	Six thermometers of six Ekman's universal emotion types: happiness, sadness, disgust, surprise, fear and anger.

5.	Name	:	Facial display selection.
	Description	:	Selects two facial displays.
	Actors	:	Facial display dictionary.
	Preconditions	:	Two system's reaction affective states are determined.
	Exceptions	:	-
	Results	:	Sending two facial displays to the user.

### 6.2.3 Object Model

Figure 6-7 below shows the object diagram of my\_Eliza prototype-1. The new object classes are:

1. *Concern of the other analyzer*, an object that performs emotion recognition from user's string input and generates system's reaction affective state.
2. *Cognitive reasoning*, an object that performs emotion recognition from system's reply sentence and generates system's reaction affective state.
3. *Affective attributing analyzer*, an object that performs and controls both emotion recognition processes for each user id. The main process of this object is to entitle the text prompt into twenty-six emotions (twenty-four emotion types of OCC theory on figure 4-1 plus uncertainty and normal).
4. *Emotive lexicon dictionary parser*, a parser that performs shallow words or phrases matching between emotive lexicon dictionary and user's string input or system's reply sentence. This parser generates two candidates of system's reaction affective state.
5. *Affective state analyzer*, an object that performs and controls two other objects: emotive lexicon dictionary parser and emotive labeled extractor, for each user id.
6. *Thermometer*, a container of six thermometers for six Ekman's universal emotional types.
7. *Intensity analyzer*, an object that monitors system's affective states per user id using system's affective thermometers.
8. *Facial display selector*, an object that selects two facial displays based on the result of the affective attributing analyzer.
9. *Nonverbal property*, an object that establishes the emotive lexicon dictionary and discrete value of distances between six Ekman's universal emotional expressions. Those discrete values are used for summation factor measurement of system's affective thermometers.
10. *Emotive transformation*, an object that performs a transformation from twenty-six emotion types (twenty-four OCC theory's emotion types plus uncertainty and normal) to eight emotion types (seven Ekman's universal emotion types plus uncertainty) based on the transformation cluster in table 4-2.

The data dictionaries of nine new object classes above are:

1. *Concern of the other analyzer*  
Attribute (all private):
  - rete: ReteMethods:
  - + ConcernOfOtherAnalysis(Blackboard blackboard, String userid)

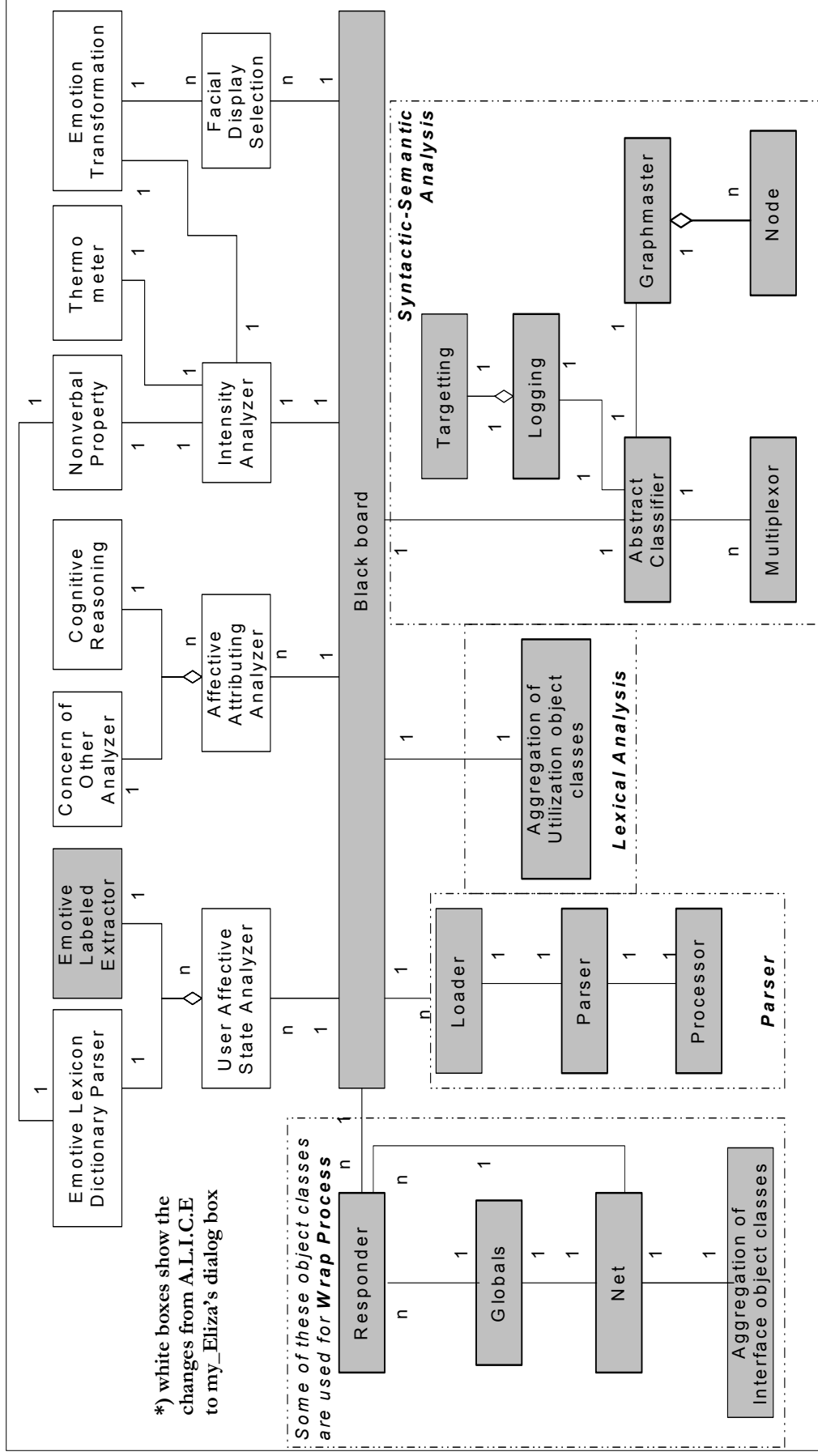
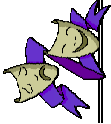


Figure 6- 7 Object Diagram of my\_Eliza prototype-1



... *Continue my\_Eliza prototype-1 data dictionary* ...

2. *Cognitive reasoning*,

```
Attribute (all private):  
-   rete: Rete  
Methods:  
+   CognitiveReasoning(Blackboard blackboard, String userid)
```

3. *Affective attributing analyzer*

```
Attribute (all private):  
-   m_reaction: String  
Methods:  
+   getReaction(): String  
+   setReaction(String react)
```

4. *Emotive lexicon dictionary parser*

```
Attribute (all private):  
-   CounterHappiness: int  
-   CounterSadness: int  
-   CounterFear: int  
-   CounterSurprise: int  
-   CounterDisgust: int  
-   CounterAnger: int  
-   Questionflag: boolean;  
Methods:  
-   stringNormalize(String sentence): String  
+   countEmotionType(String sentence)  
+   getCounterHappiness(): int  
+   getCounterSadness(): int  
+   getCounterFear(): int  
+   getCounterSurprise(): int  
+   getCounterDisgust(): int  
+   getCounterAnger(): int  
+   getQuestionFlag(): boolean
```

5. *Affective state analyzer*

```
Methods:  
+   affectExtraction(String userid, Blackboard blackboard): boolean  
+   concernExtraction(String userid, Blackboard blackboard): boolean  
+   maxEmotionType(EmotiveLexiconDictionaryParser parsing): String  
+   parseInput(String input, Blackboard blackboard): boolean  
+   parseReply(String reply, Blackboard blackboard): boolean
```

6. *Thermometer*

```
Attribute (all private):  
-   happyThermo: double  
-   sadThermo: double  
-   disgustThermo: double  
-   fearThermo: double  
-   surpriseThermo: double  
-   angryThermo: double  
Methods:  
+   setThermo(int flag, int intensity)  
+   double getHappyThermo(): double  
+   getSadThermo(): double  
+   getFearThermo(): double  
+   getSurpriseThermo(): double  
+   getAngryThermo(): double  
+   getDisgustThermo(): double  
+   getMaxThermo(): String
```

7. *Intensity analyzer*

```
Methods:  
+   newIntensityAnalysis(String userid, Blackboard blackboard)  
+   setIntensity(String userid, Blackboard blackboard, String emotionFace)  
+   getIntensity(String userid, Blackboard blackboard): Thermometer
```



8. *Facial display selector*

```
Attribute (all private):  
- FacialDictionary: Hashtable  
Methods:  
+ select(String affect, boolean quest, int intensity): String
```

9. *Nonverbal property*

```
Attribute (all private):  
- happinessDistances: Hashtable  
- sadnessDistances: Hashtable  
- fearDistances: Hashtable  
- disgustDistances: Hashtable  
- angerDistances: Hashtable  
- surpriseDistances: Hashtable  
- nonverbalDictionary: Hashtable // storage of emotive lexicon dictionary  
- nonverbalComparison: Hashtable  
- nonverbalAttribute: Hashtable  
Methods:  
+ getDistanceValue(int flag, String name): Double  
+ addDistanceValue(String name, String value)  
+ setDistanceValue(int flag, String name, Double value)  
+ addNonverbalDictionary(String name, String value)  
+ setNonverbalDictionary(String emotionType, String word)  
+ getIndexWordList(Vector wordList, String word, String tail): int  
+ checkInEmotionType(String emotionType, String word, String tail): boolean  
+ recognizeEmotionType(String word, String tail): Vector  
+ addNonverbalComparison(String name, String value)  
+ setNonverbalComparison(String comparison, String word)  
+ checkInComparisonList(String comparison, String word): boolean  
+ addNonverbalAttribute(String name, String value)  
+ setNonverbalAttribute(String attribute, String word)  
+ recognizeAttribute(String word): String  
+ checkInAttributeList(String attribute, String word): boolean
```

10. *Emotive transformation*

```
Methods:  
+ transformOCC2Ekman(String emotionStr): String  
+ transformOCC2Pictures(String emotion): String
```

## 6.2.4 Dynamic Model

Scenario of my\_Eliza prototype-1 is also divided into two parts:

1. *Load time.* When the system runs, it loads all pattern rules into the *graphmaster* and list of words/phrases into the *multiplexor*. At the same time, the system also loads the discreet values of distances between six emotion types and emotive lexicon dictionary into the nonverbal property.
2. *Conversation time.* When the system receives user's string input via *responder* (an object derived from A.L.I.C.E, see appendix B), the *concern of the other analyzer* performs emotion recognition from the input and facial display selector selects a first reaction facial display that is appropriate to it. The next phase is to construct a reply sentence by *syntactic-semantic analyzer* object classes. The cognitive reasoning performs emotion recognition from the reply sentence and facial display selector once more time uses the result to select the appropriate facial display to convey this reply.

By the scenario above, we construct an event trace diagram shown in figure 6-8 below. Similar to the event flow diagram of my\_Eliza's dialog box, the shadowed boxes means that those object classes are an aggregate of some object classes of Program D A.L.I.C.E and we rename those aggregation based on the global design in figure 6-1.



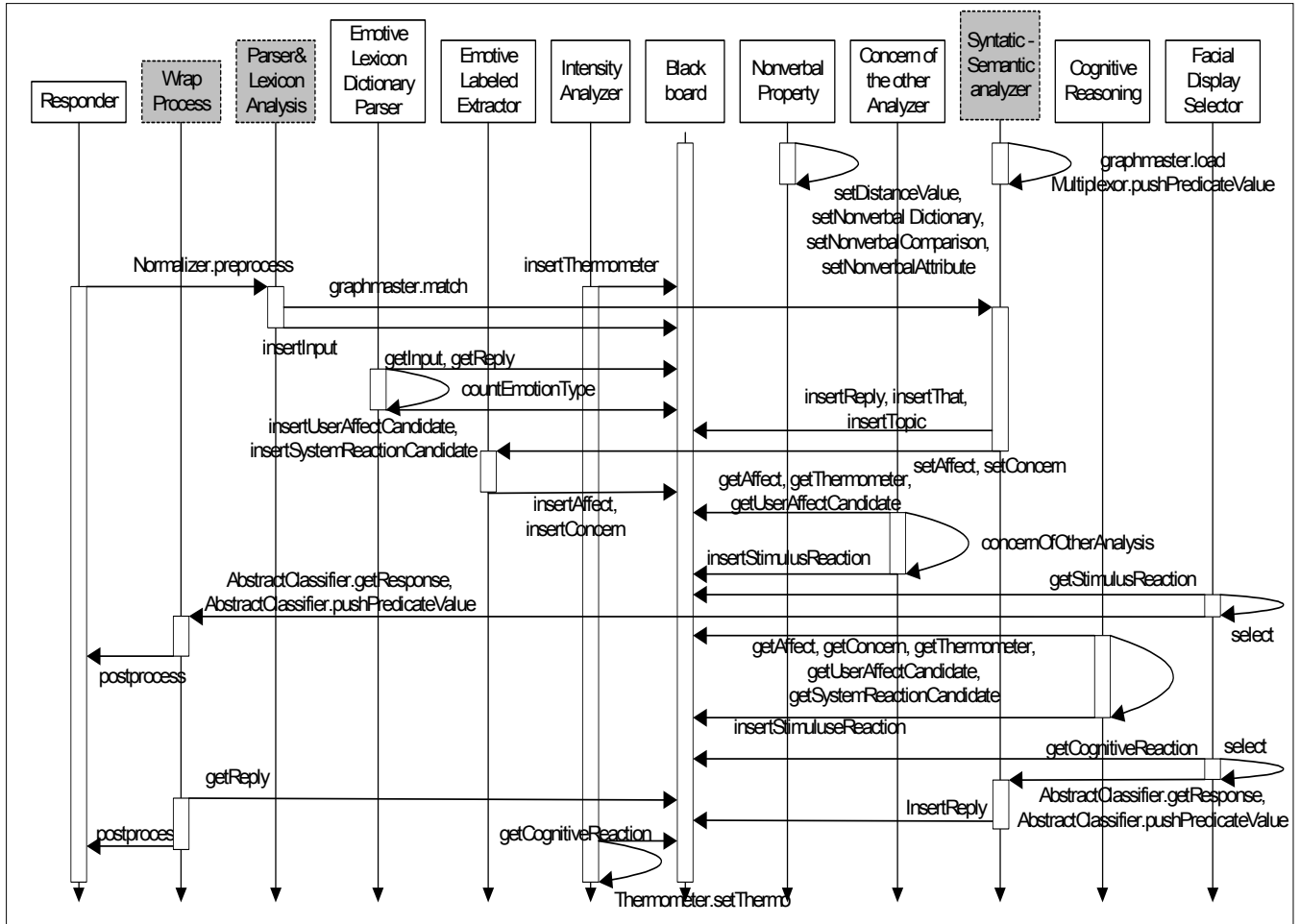


Figure 6- 8 Event flow diagram of my\_Eliza prototype-1: loading and conversing

According to the scenario above, we construct transition diagrams of all new object classes in figure 6-9 until figure 6-14. Some object classes, such as: concern of the other analyzer, cognitive reasoning, emotive lexicon dictionary parser, and facial selector, store their result into the blackboard. Therefore, in their transition diagram there is not a transition to store their property to other object classes. The methods of these object classes exchange the data through blackboard. Those exchange events also can be seen clearly in the event flow diagram in figure 6-8 above.

1. *Concern of the other Analyzer, Cognitive Reasoning, and Affective Attributing Analyzer* transition diagram

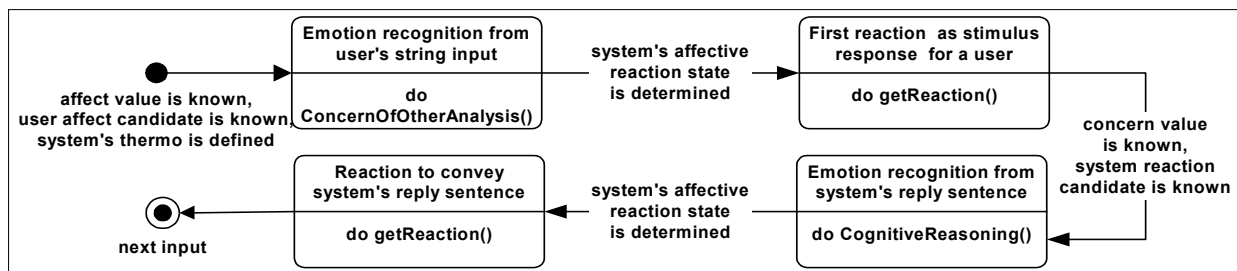


Figure 6- 9 Concern of the other analyzer, cognitive reasoning and affective attributing analyzer transition diagram



2. *Emotive Lexicon Dictionary Parser and Affective State Analyzer* transition diagram

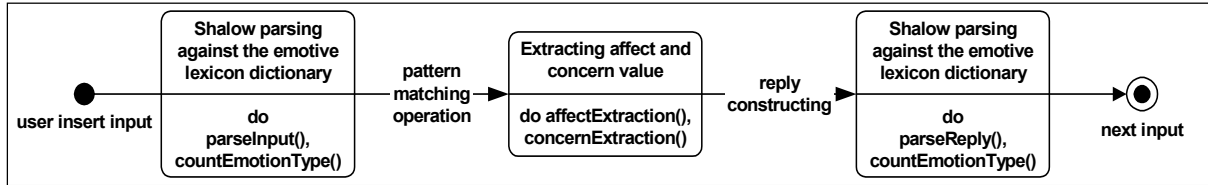


Figure 6- 10 Emotive lexicon dictionary parser and user's affective state analyzer transition diagram

3. *Thermometer and Intensity Analyzer* transition diagram

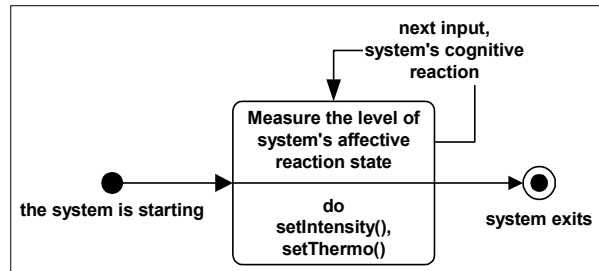


Figure 6- 11 Thermometer and intensity analyzer transition diagram

4. *Facial Display Selector* transition diagram

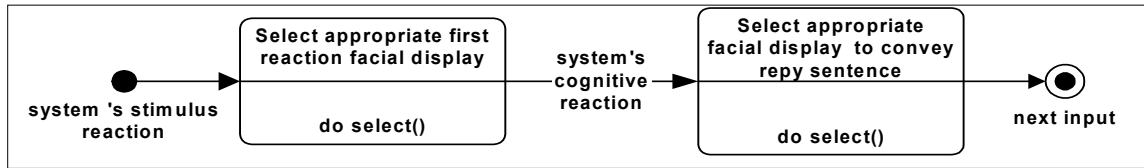


Figure 6- 12 Facial display selector transition diagram

5. *Nonverbal property* transition diagram

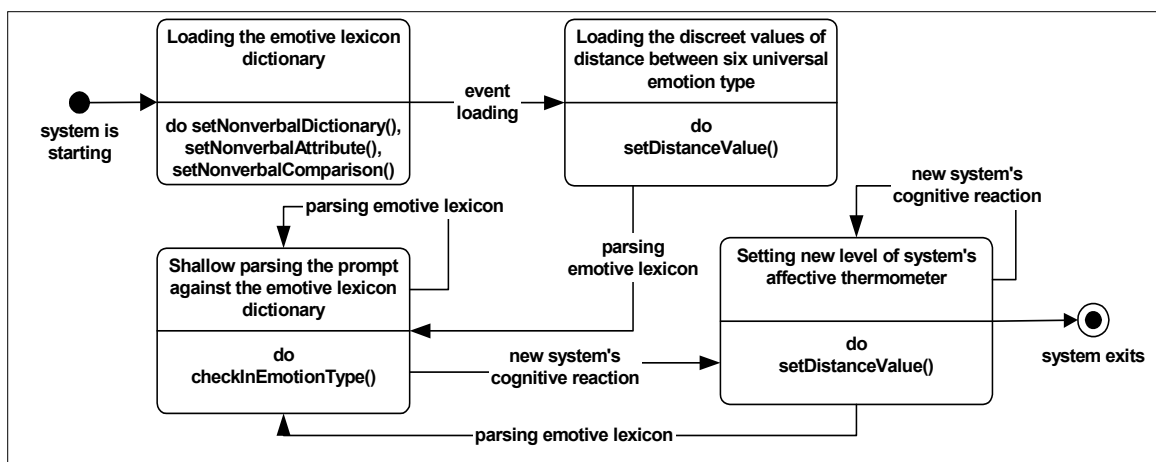


Figure 6- 13 Nonverbal property transition diagram



6. Emotion Transformation transition diagram

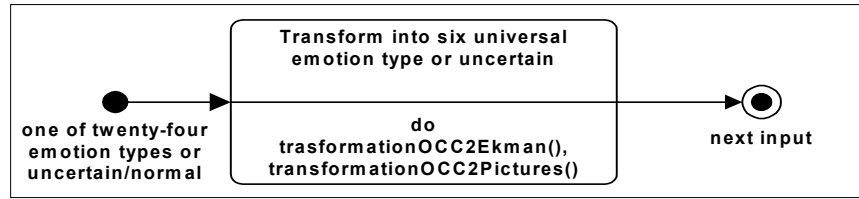


Figure 6- 14 Emotion transformation transition diagram

## 6.2.5 Affective Knowledge Base Design

As described in section 4.2.2.2, the affective knowledge base controls the whole emotion from a dialog state in the “IF-THEN” form and there are two kinds of knowledge bases inside it:

1. **Stimulus response knowledge base**, a knowledge base that used to classify the system’s reaction affective state based on system’s stimulus response to user’s input string. Table 6-1 is an example of preference rules for my\_Eliza’s prototype to generate the first reaction to user’s string input.

Table 6- 1 Example of stimulus response preference rules

Rule #	User-affective state candidate	Input Question	Affect-value	Thermometer	First-reaction
1.	Happiness	True	+/#	Happiness	Joy
2.	Happiness	False	+/#	Happiness	Happy-for
3.	Happiness	True	-	Happiness	Uncertainty
4.	Happiness	False	-	Happiness	Disappointment
5.	Sadness		+/-	Happiness	Fear
6.	Sadness		#	Happiness	Uncertainty
7.	Sadness		+/-	Sadness	Sorry-for
8.	Sadness		#	Sadness	Uncertainty
9.	...	...	...	...	...

Currently, my\_Eliza prototype-1 has seventy-seven preference rules in the stimulus response knowledge base (see appendix C.1), which take the following conditions into consideration:

- **User-affective state candidate** – the result of emotive lexicon dictionary parsing process of user’s string input.
- **Input-question** – whether the user’s string input is a question sentence or not.
- **Affect-value** – the value situation type of affect tag as the result of the emotive labeled extractor.
- **Thermometer**– current system’s affective level.

Note: that in the rule-set tables (above and in the appendix) an empty field means ‘don’t care’, which means that the corresponding condition does not influence the firing of the rule. Below we explain some examples of the reasoning process of the preference rules above.

### Preference rule 1:

This rule will fire the preference first reaction **joy** if the following conditions are met:

- user is happy,
- user asks question,
- situation type of user is not negative,
- current maximum system’s affective thermo is happy.

In this case my\_Eliza will answer any questions from the user joyfully, because she enjoys the situation and she met the goal: making the user feel happy.



### Preference rule 3:

This rule will fire the preference first reaction **uncertainty** if the following conditions are met:

- user is happy,
- user asks question,
- situation type of user is negative,
- current maximum system's affective thermo is happy.

In this case my\_Eliza do not know how to react to the user since the conditions do not match to each other. My\_Eliza is happy and she assumes that the user is also happy, but it is not clear why the user asks negative things. It is shown by the situation type is negative.

### Preference rule 5:

This rule will fire the preference first reaction **fear** if the following conditions are met:

- user is sad,
- situation type of user is not joking,
- current maximum system's affective thermo is happy.

Here my\_Eliza is surprised and sad because the user suddenly feels sad.

### Preference rule 7:

This rule will fire the preference first reaction **sorry for** if the following conditions are met:

- user is sad,
- situation type of user is not joking,
- current maximum system's affective thermo is sad.

In this case my\_Eliza's reaction is sorry for the user and the conversation condition is also mourning.

2. **Cognitive process knowledge base**, a knowledge base that used to classify the system's reaction affective state based on system's cognitive response to convey system's reply sentence. Table 6-2 is an example of preference rules for my\_Eliza's prototype to generate reactions that convey the reply.

Table 6- 2 Example of cognitive reasoning preference rules

Rule #	user-affective state candidate	Affect-value	system-reaction affective state candidate	Concern -value	Thermometer	Cognitive-reasoning reaction
1.	Happiness	+	Happiness		Happiness	Happy for
2.	Happiness	#	Happiness		Happiness	Joy
3.	Happiness	-	Happiness	-	Happiness	Dislike
4.	Sadness	+/-	Sadness	+	Happiness	Sorry-for
5.	Sadness	+/-	Sadness	-	Happiness	Distress
6.	Sadness		Sadness	#	Happiness	Gloating
7.	Sadness	#	Sadness	+	Happiness	Joy
8.	Sadness	#	Sadness	-	Happiness	Resentment
9.	...	...	...	...	...	...

My\_Eliza prototype-1 has 151 preference rules of stimulus response knowledge base (see appendix C), which take the following conditions into consideration:

- **User-affective state candidate**
- **Thermometer**
- **Affect-value**
- **Concern-value** – the value situation type of concern tag as the result of the emotive labeled extractor.
- **System-reaction affective state candidate** – the result of emotive lexicon dictionary parser parsing system's reply sentence.



In our first prototype we simplify the emotion eliciting factors, which influences system's reaction affective state, comparing to the cognitive process knowledge base in the global design (see section 4.2.2.2). Below we explain some examples of the reasoning process of the preference rules above.

**Preference rule 1:**

This rule will fire the cognitive reasoning preference **happy for** if the following conditions are met:

- user is happy,
- system's reply is happy,
- situation type of user is positive,
- current maximum system's affective thermo is happy.

In this case my\_Eliza assumes that the user tells happy news to her and for that, she feels happy for the user's condition.

**Preference rule 3:**

This rule will fire the cognitive reasoning preference **dislike** if the following conditions are met:

- user is happy,
- situation type of user is negative,
- situation type of system is negative,
- current maximum system's affective thermo is happy.

In this case my\_Eliza does not like what the user just said to her and she replies the user in a negative way.

**Preference rule 5:**

This rule will fire the cognitive reasoning preference **distress** if the following conditions are met:

- user is sad,
- system's reply is sad,
- situation type of user is not joking,
- current maximum system's affective thermo is sad.

Here my\_Eliza is sad because the user suddenly feels sad and says negative things to her.

**Preference rule 8:**

This rule will fire the cognitive reasoning preference **resentment** if the following conditions are met:

- user is sad,
- system's reply is sad,
- situation type of user is joking,
- situation type of the system is negative,
- current maximum system's affective thermo is sad.

Here my\_Eliza does not like the user makes a joke while she feels sad.

## 6.3 My\_Eliza's Prototype-1 Implementation

My\_Eliza's Prototype-1 also is written on Java Development Kit version 1.3 and XML. The way to compile the prototype uses the same way as Program D A.L.I.C.E (can be seen in appendix B). Since this prototype is expanding from My\_Eliza's dialog box, it also contains the same package and the same file system/directory configuration. All new objects above are collected in package `org.Alicebot.server.nonverbal`, which now contains also ten new Java files. We give those Java files the same name and the same description as the object classes that have been explained in section 6.3 above. This Java files have extension **.java**.



When the server runs for the first time, on the console prompt will appear list of executed routines (see figure 6-15). As mentioned earlier, my\_Eliza is developed from Program D A.L.I.C.E version 4.1.3, therefore we still use Alicebot's server configuration. On running time, the server loads four databases into the system, they are:

- (a) *The list of words/phrases.* Figure 6-15 displays that the server loads 288 input substitutions, 19 gender substitutions, 9 person substitutions, 60 person2 substitutions and 3 sentence-splitters. Please refer to section 4.2.1.2 for the AIML form for writing this list.
- (b) *The discreet values of distances between six universal emotion types.*
- (c) *The emotive lexicon dictionary.* Figure 6-15 displays that the server loads 33 happiness lexicons, 16 surprise lexicons, 13 fear lexicons, 28 anger lexicons, 38 disgust lexicons, 51 sadness lexicons, 6 high comparative, 8 low comparative, 33 negations attributes, and 6 question attributes. Please refer to section 4.2.2.1 for the AIML form for writing the content of the dictionary.
- (d) *The list of pattern rules.* Figure 6-15 displays that the server loads 1953 AIML categories into the *Graphmaster*. Addition, currently Alicebot has more than 41000 categories in its Graphmaster.

The first three databases above are stored in the same file, i.e. `./bots/startup.xml`. This file contains also the system's preferences database. Please refer to section 4.2.1.2 for writing this database.

```
[2002-07-16 17:12:07] Starting Alicebot Program D version 4.1.3
[2002-07-16 17:12:07] Using Java VM 1.3.0-C from Sun Microsystems Inc.
[2002-07-16 17:12:07] On Windows 2000 version 5.0 (x86)
[2002-07-16 17:12:07] Bot predicates with no values defined will return: "undefined".
[2002-07-16 17:12:08] Initializing Multiplexor.
[2002-07-16 17:12:08] Starting Graphmaster.
[2002-07-16 17:12:19] Loaded 288 input substitutions.
[2002-07-16 17:12:19] Loaded 19 gender substitutions.
[2002-07-16 17:12:19] Loaded 9 person substitutions.
[2002-07-16 17:12:19] Loaded 60 person2 substitutions.
[2002-07-16 17:12:19] Loaded 3 sentence-splitters.
[2002-07-16 17:12:19] Loaded happiness descreet distance values.
[2002-07-16 17:12:19] Loaded surprise descreet distance values.
[2002-07-16 17:12:19] Loaded fear descreet distance values.
[2002-07-16 17:12:19] Loaded anger descreet distance values.
[2002-07-16 17:12:19] Loaded disgust descreet distance values.
[2002-07-16 17:12:19] Loaded sadness descreet distance values.
[2002-07-16 17:12:19] Loaded 33 happiness lexicons-dictionary.
[2002-07-16 17:12:19] Loaded 16 surprise lexicons-dictionary.
[2002-07-16 17:12:19] Loaded 13 fear lexicons-dictionary.
[2002-07-16 17:12:19] Loaded 28 anger lexicons-dictionary.
[2002-07-16 17:12:19] Loaded 38 disgust lexicons-dictionary.
[2002-07-16 17:12:19] Loaded 51 sadness lexicons-dictionary.
[2002-07-16 17:12:19] Loaded 6 high comparatives.
[2002-07-16 17:12:19] Loaded 8 low comparatives.
[2002-07-16 17:12:19] Loaded 33 negation attributes.
[2002-07-16 17:12:19] Loaded 6 question attributes.
[2002-07-16 17:12:20] 1953 categories loaded in 11.897 seconds.
[2002-07-16 17:12:20] The AIML Watcher is active.
[2002-07-16 17:12:20] "my_Eliza" is thinking with 1953 categories.
[2002-07-16 17:12:20] Alicebot Program D version 4.1.3 Build [04].
[2002-07-16 17:19:37] Type exit to shut down.
```

Figure 6- 15 Starting my\_Eliza server script

My\_Eliza's list of pattern rules is stored in the `./bots/brain.aiml` file. We can add this list in other AIML files. The server knows which AIML files must be loaded by reading the `./bots/startup.aiml` file. This files contains a category with pattern "LOAD ROBOT" (see figure 6.16 below). It has a template that contains list of file name of the AIML files with their relative directory: `/home/Alicebot/brain/<AIML file-name>`. We can use another filename for `startup.aiml` and `startup.xml` as long as those are registered in the



server.properties file that contains the server and the file system configurations. The parser interprets <learn>X</learn> tag to load AIML file with X is the filename.

```
<aiml>
  <category>
    <pattern>LOAD ROBOT</pattern>
    <template>
      <learn filename="/home/Alicebot/brain/brain.aiml"/>
      <learn filename="/home/Alicebot/brain/a.aiml"/>
      <learn filename="/home/Alicebot/brain/b.aiml"/>
      ...
    </template>
  </category>
</aiml>
```

Figure 6- 16 Example category for loading the list of pattern rules files

Since my\_Eliza dialog box was developed using Java, a natural choice for knowledge based system shell is the **Java expert system shell** (Jess). Jess is a rule engine and scripting environment written entirely in Java. It was originally inspired by the CLIPS expert system shell, but has grown into a complete, distinct Java-influenced environment of its own. Because of its complete implementation in Java, the rule-engine can be easily embedded within the Java program. For detailed information about Jess see [FRI00]. The script of the knowledge base is written in .clp. In my\_Eliza prototype-1, the knowledge base scripts are stored into two .clp files:

- ./bots/lheart.clp, contains concern of the other knowledge base transformed from all preference rules in appendix C.1 into CLIPS language form.
- ./bots/rheart.clp, contains cognitive reasoning knowledge base transformed from all preference rules in appendix C.2 into CLIPS language form.

They are stored in the same directory as the list of pattern rules and registered also in the server.properties file. See appendix B for detailed information about server.properties file.

here.'"/>

Figure 6- 17 My\_Eliza prototype-1's login page

When the user requests to my\_Eliza's URL address of this prototype, the interface as in figure 5-25 will appear. The user also has to register and login first before they converse with my\_Eliza. My\_Eliza's prototype-1 has different login interface, which asks the user to choose his/her icon (see figure 6-17 above). The chosen icon will be used to represent the user during conversation with the system. Similar to my\_Eliza's dialog box, my\_Eliza prototype-1 can only work properly if the user uses Internet Explorer from Microsoft.



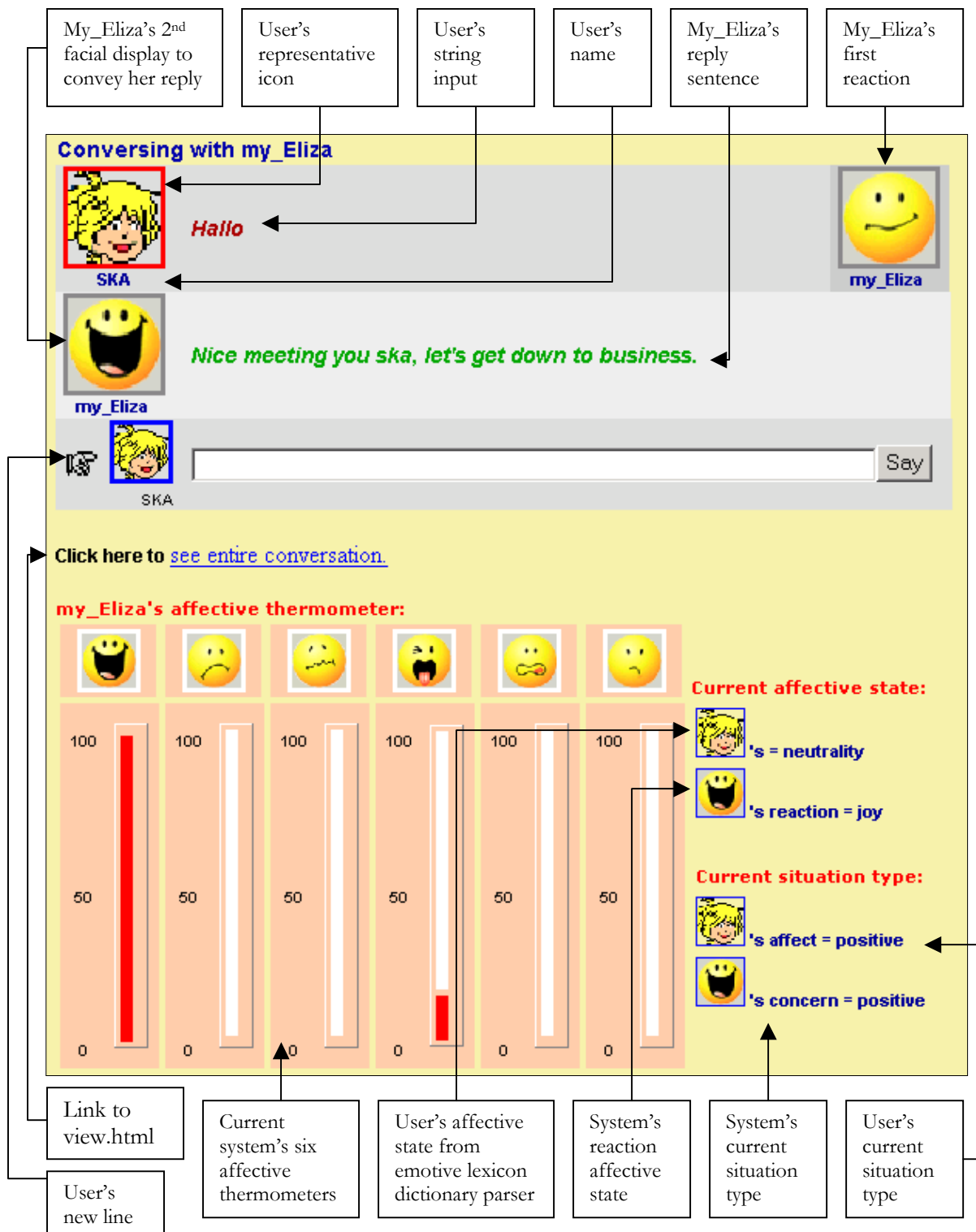


Figure 6- 18 The main page of my\_Eliza prototype-1: chat.html

Comparing to its dialog box as displayed in figure 5-27, My\_Eliza prototype-1's main page has several addition properties as appeared in the main page: chat.html (see figure 6-18). Similar to my\_Eliza's dialog box in figure 5-27, this page also contains user's string input and system's reply sentence. "You said" line is changed by user's representative icon that is selected in the login



session. “My\_Eliza said” line is changed by system’s facial display to convey its reply sentence. The user can input the prompt text by clicking the button “Say” or typing enter. Beside those, this page also contains two facial displays: (1) first reaction to user’s string input and (2) a facial display to convey system’s reply sentence. Every time the page is restored, it also shows updated six system’s affective thermometers to the user, system’s reaction affective state, user’s affective state from *emotive lexicon dictionary parser*, user’s current situation type and system’s current situation type. The page also shows the user’s name, which is updated every time the user tells the system his/her name, for example: “MY NAME IS SKA”, “YOU CAN CALL ME SKA”, and so on.

By click the line “see entire conversation” in the main page, a new window will appear – view.html. This window displays the entire user-my\_Eliza conversation complete with two my\_Eliza’s facial displays (figure 6-19).

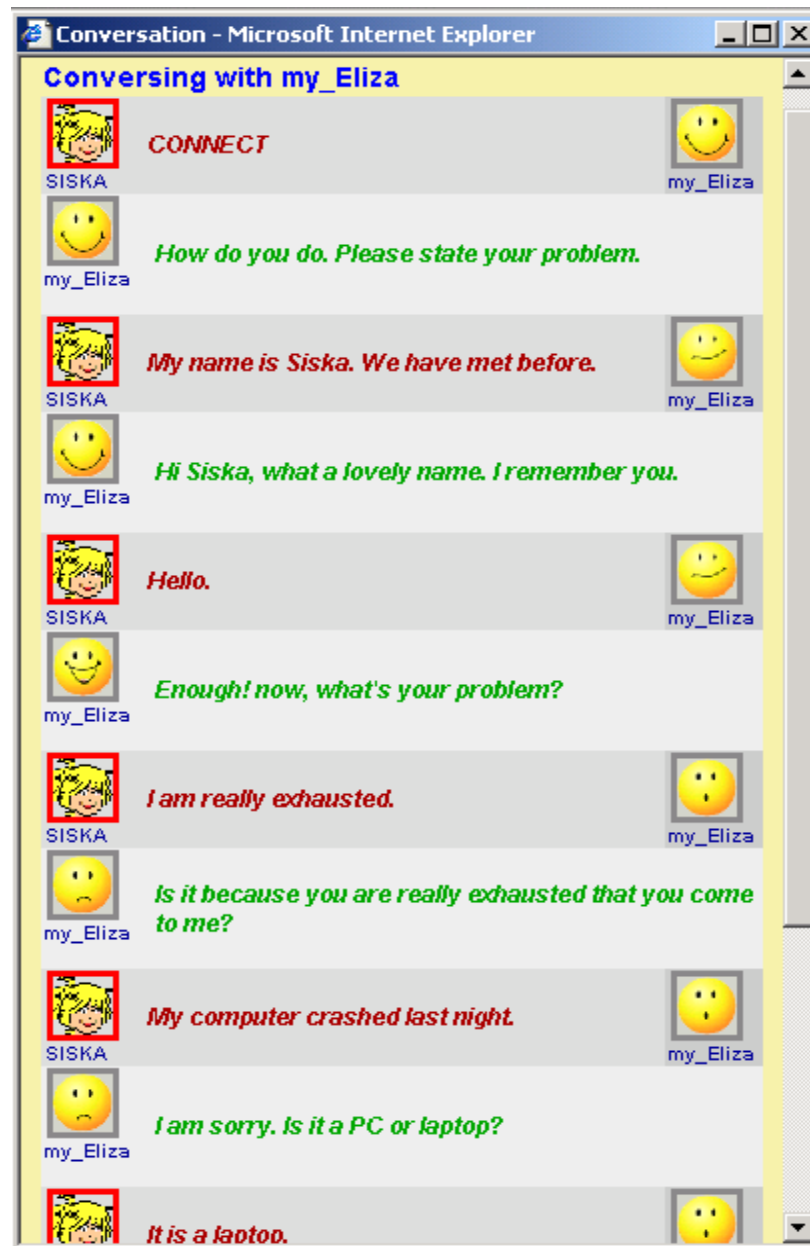


Figure 6- 19 My\_Eliza’s View.html



## Chapter 7 Testing of My\_Eliza Prototype-1

In this chapter we will review an example of conversation between my\_Eliza prototype-1 and a human user. We will review it from the point of view of the designer/programmer. In this phase, we do an experiment with the system and see how it reacts. The experiment in this case is a collection of string input from the user. The expected reactions from the system are the system's reply sentences and system's nonverbal facial displays. This prototype may not be a perfect version, however here with our review we analyze it. We contribute the analysis with some solutions of some problems that occur in this experiment.

This chapter is divided into two sections. First, the initial state of the system before the experiment is described in section 7.1. Finally, the experiment itself is presented in section 7.2. This section is divided into some problems that occur in this experiment. For each problem, we analyze it and give a comment about it, and then we propose a solution for it.

### 7.1 Initial State of the System

In this point, my\_Eliza prototype-1's emotive lexicon dictionary contains:

- 48 lexicons for happiness,
- 170 lexicons for sadness,
- 34 lexicons for surprise,
- 33 lexicons for fear,
- 93 lexicons for disgust, and
- 69 lexicons for anger.

This prototype has 1953 categories in its list of pattern rules **without** any topic discussion. Figure 7-1 displays the templates for system's default category if none of category matches to the user's string input pattern.

- `<get name="name"/>`, you don't seem reasonable.
- If you continue like this I'll simply delete your autoexec.bat!
- What makes you think that?
- I don't get what you're trying to tell me.
- What does that suggest to you?
- Tell me about your family.
- What are your dreams?
- Let's change the subject.
- Is everything all right with you, `<get name="name"/>`?
- I'm not sure I understand you fully.
- Come, come, and elucidate your thoughts to me.
- Can you elaborate on that?
- Open your mind a little more.
- Let's forget it for now, all right?
- How about speaking of computers?
- Tell me more about your problem.
- That doesn't really interest me.
- Would you like to speak about sports?
- Have you got something to add to that?
- What are you speaking about?

Figure 7- 1 Templates of default category with `<pattern>*<pattern>`



The facial display dictionary of this prototype contains twenty-two facial display files in format GIF. They are stored in `./template/html/pictures` and divided by eight emotion types. The table 7.1 displays my\_Eliza's facial displays that used in the prototype. Those are smiley icons and were downloaded from [www.smileydictionary.com](http://www.smileydictionary.com) made by Hunter Solutions, an IT Company. Each of emotion types has three levels of intensities (except for neutrality): low, medium and high.

Table 7- 1 My\_Eliza prototype-1's facial display dictionary

Intensity	Emotion name							
	Happiness	Sadness	Surprise	Fear	Disgust	Anger	Neutrality	Uncertainty
Low								
Medium								
High								

Figure 7-2 below displays my\_Eliza prototype-1 when a user connects to the system the first time. The system still does not know the user's name therefore it defines "UNDEFINE" under the user's icon. My\_Eliza assumes the user's situation type (affect) at the first time is in positive situation. From the *emotive lexicon dictionary parser*, the system known that there is not any emotive lexicon in the user's string input and it concludes that user's affective state (user-affect) is neutrality. The system's affective thermometer (thermo) initiates by incrementing happiness thermometer and it calculates that the emotion type with the highest degree of all the six thermometers is happiness. Therefore the Jess output of the *concern of the other analyzer* is:

```
f-28 user-affect neutrality
f-29 thermo happiness
f-30 affect positive
f-31 set-reaction normal
For at total of 32 facts.
```

Using the corresponding between six universal emotion types and OCC's theory emotion types in table 4-4, the system displays neutrality facial display as my\_Eliza's stimulus response. The system replies the user in positive situation too. The emotive lexicon dictionary parser finds phrase "state your problem" in the happiness dictionary and conclude that system's candidate reaction affective state (system-react) is happiness. Therefore, the Jess output of *cognitive reasoning* is:

```
f-28 user-affect neutrality
f-29 system-react happiness
f-30 thermo happiness
f-31 affect positive
f-32 concern positive
f-33 set-reaction joy
For at total of 34 facts.
```

Using transformation in the table 4-4, the system displays happiness facial display as my\_Eliza's cognitive processed reaction. The affective thermometers are updated based on this result. The system displays them.

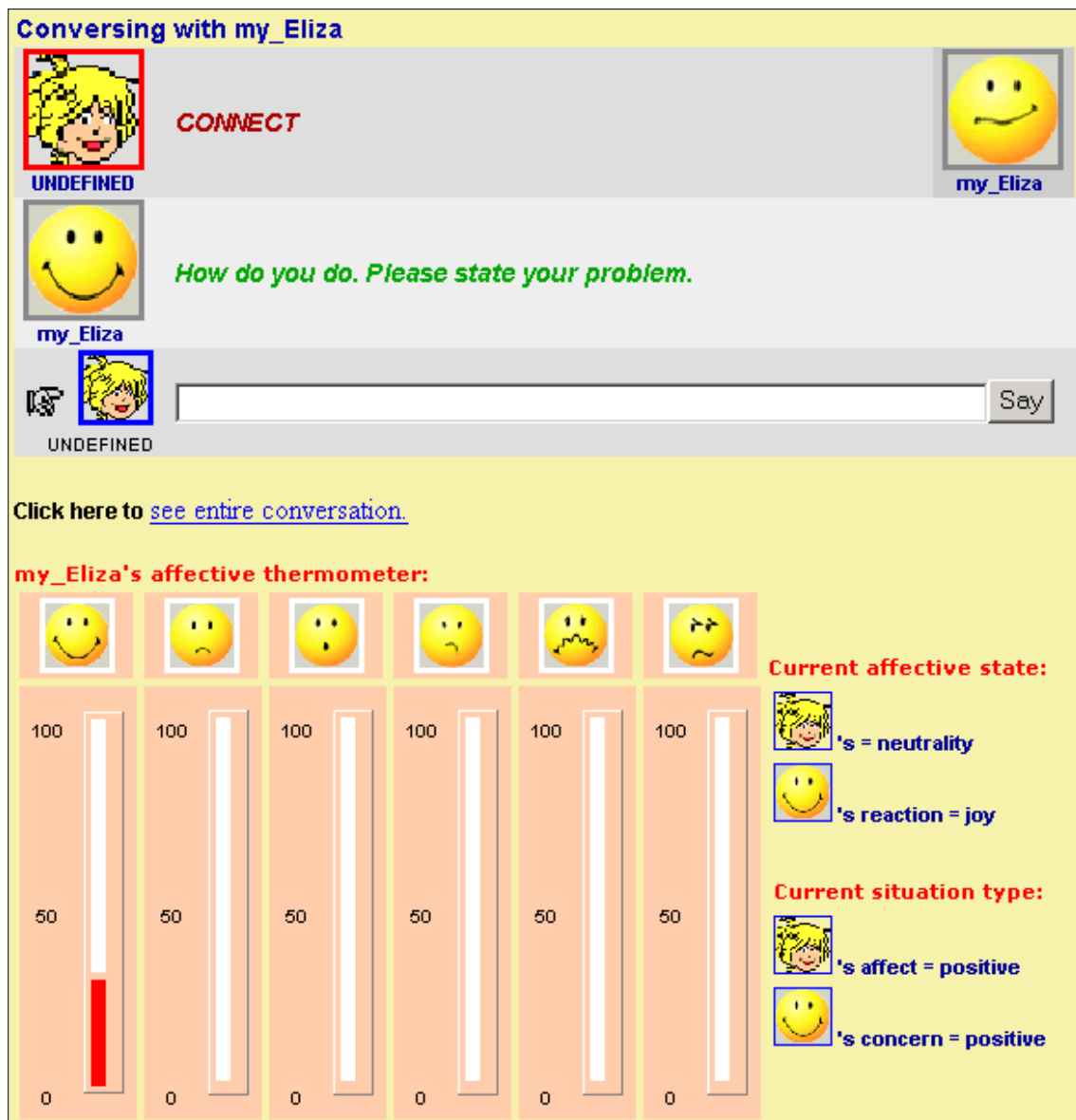


Figure 7- 2 The first connect

## 7.2 Conversing with my\_Eliza Prototype-1

In this section we review the conversation between my\_Eliza and a human player. The system will try to reason the user's emotion from the text and give reaction based on it. Unfortunately, in this thesis we cannot display the exact dialog example as we have experimented with my\_Eliza's dialog box in chapter 5. It is due to my\_Eliza has more than one reply sentence to every input pattern and the conversation flow relatives to them. In this section, we display my\_Eliza's main page and we comment based on the changing from the previous figure to the successor figure. The experiment is completed sequentially and the result cannot be loosed one another.

From figure 7-2 into figure 7-3 (below) the changing is:

1. My\_Eliza stores the user's name, indicated by displaying it under the user's icon.
2. The user's situation type is positive.



3. The user's affective state is still neutrality.

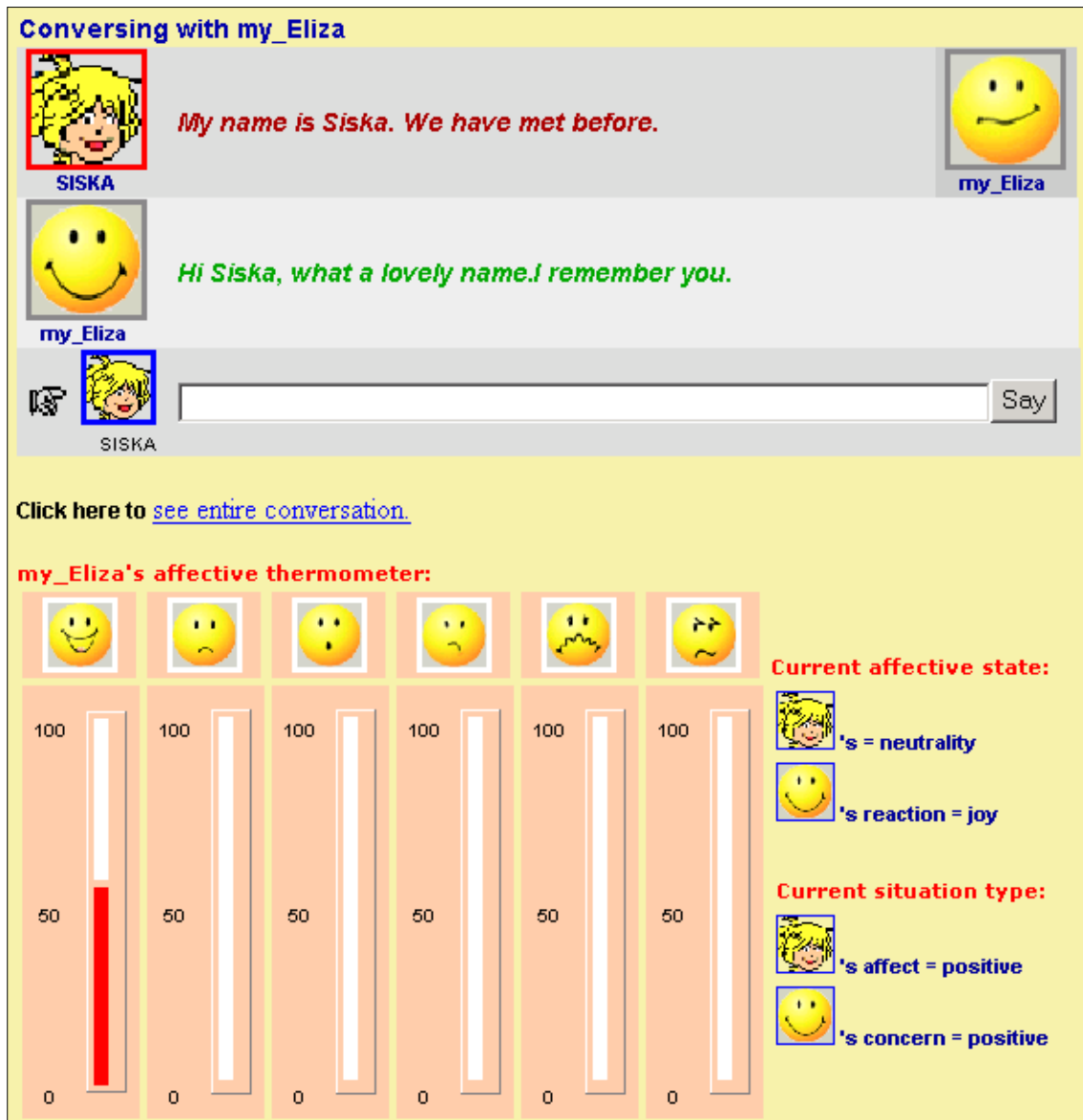


Figure 7- 3 My\_Eliza stores user name

4. The highest thermometer is happiness.
5. The Jess of concern of the other analyzer gives us:  

```
f-28 user-affect neutrality
f-29 thermo happiness
f-30 affect positive
f-31 set-reaction normal
For at total of 32 facts.
```
6. The stimulus response facial display is neutrality.
7. The system's candidate reaction affective state is happiness. The parser finds word "lovely" in the happiness dictionary.
8. The system's situation type is positive.
9. The Jess of cognitive reasoning gives us:  

```
f-28 user-affect neutrality
```



f-29 system-react happiness  
f-30 thermo happiness  
f-31 affect positive  
f-32 concern positive  
f-33 set-reaction joy  
For at total of 34 facts.

10. The cognitive processed facial display is happiness.
11. The system updates the affective thermometers.

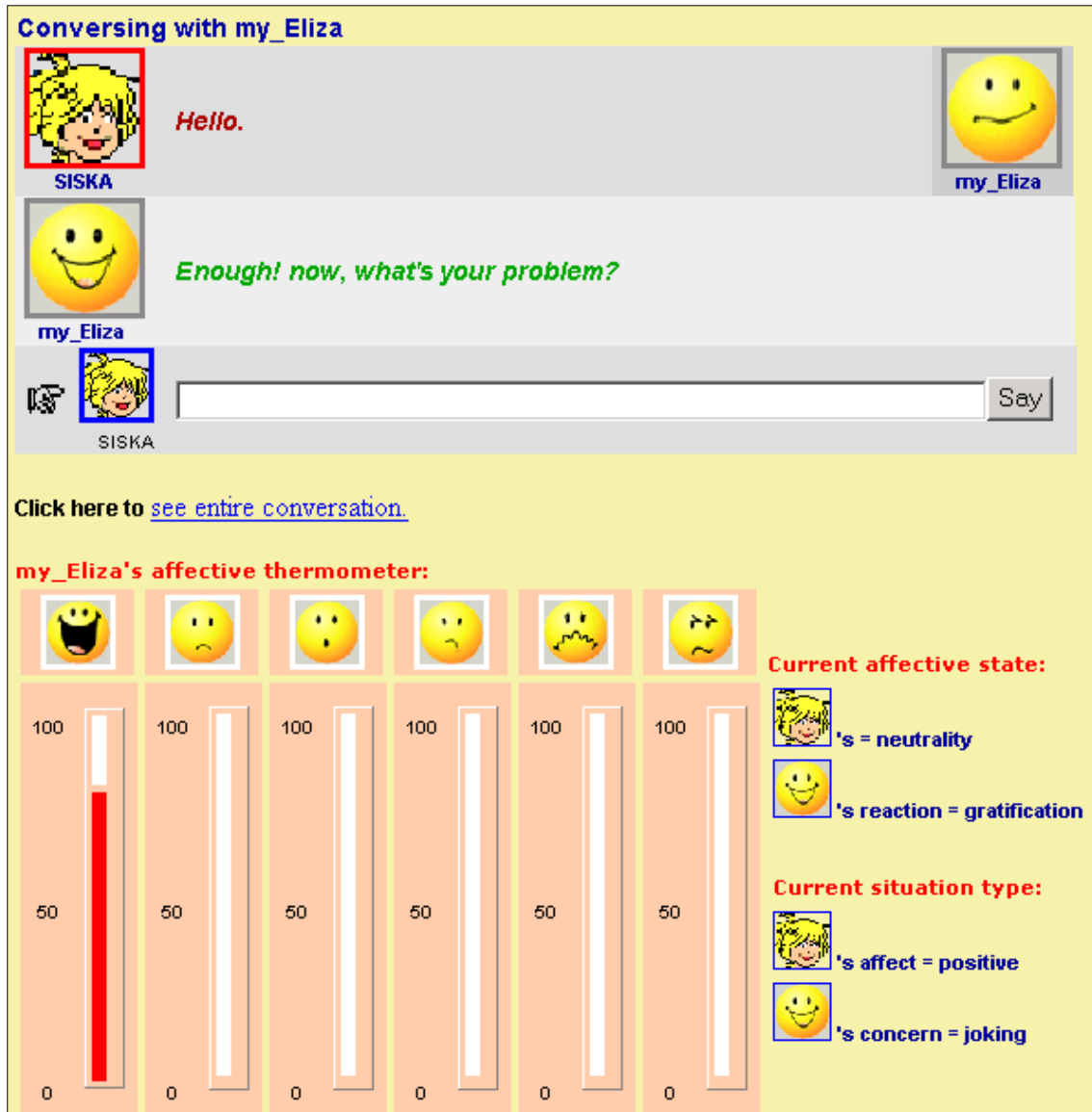


Figure 7- 4 My\_Eliza jokes with the user

From figure 7-3 into figure 7-4 the changing is:

1. The user's situation type is positive.
2. The user's affective state is still neutrality.
3. The highest thermometer is still happiness.
4. The Jess of concern of the other analyzer gives us:

f-28 user-affect neutrality  
f-29 thermo happiness  
f-30 affect positive





f-31 set-reaction normal  
For at total of 32 facts.

5. The stimulus response facial display is neutrality.
6. The system's candidate reaction affective state is neutrality. The parser does not find any emotive lexicon in the reply sentence.
7. The system's situation type is joking.
8. The Jess of cognitive reasoning gives us:

f-28 user-affect neutrality  
f-29 system-react neutrality  
f-30 thermo happiness  
f-31 affect positive  
f-32 concern joke  
f-33 set-reaction gratification  
For at total of 34 facts.

9. The cognitive processed facial display is happiness with intensity medium.
10. The system updates the affective thermometers.

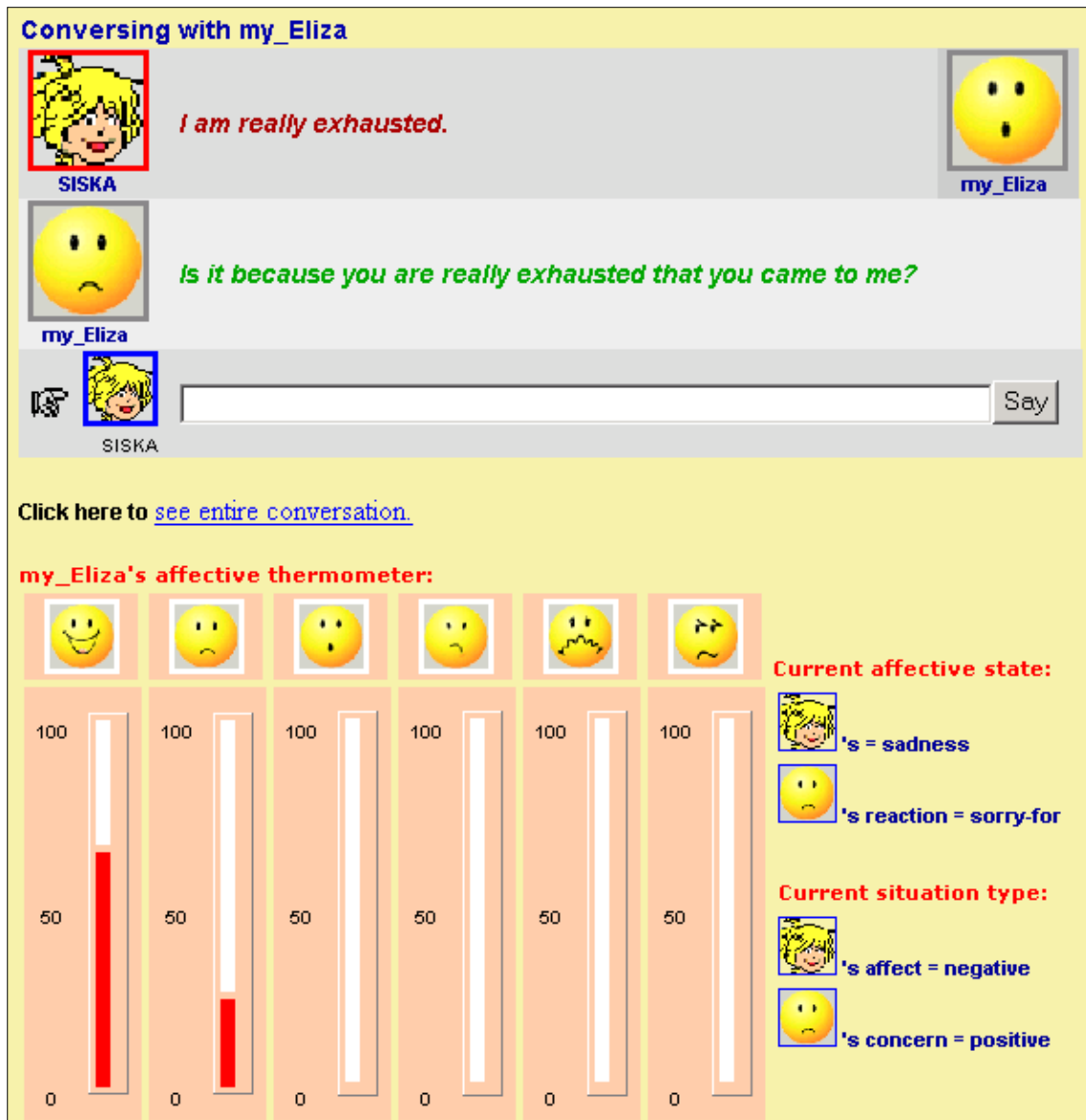


Figure 7- 5 The user is sad



From figure 7-4 into figure 7-5 (above) the changing is:

1. The user's situation type is negative.
2. The user's affective state is sadness. The parser finds phrase "really exhausted" in the sadness dictionary with medium level intensity: [REALLY EXHAUSTED:2].
3. The highest thermometer is still happiness.
4. My\_Eliza is surprised because situation is suddenly changed from happiness to sadness. The Jess of concern of the other analyzer gives us:

```
f-28 user-affect sadness
f-29 thermo happiness
f-30 affect negative
f-31 set-reaction fear
For at total of 32 facts.
```

5. The stimulus response facial display is surprise.
6. The system's candidate reaction affective state is also sadness. The parser also finds phrase "really exhausted" in the reply sentence.
7. The system's situation type is positive.
8. The Jess of cognitive reasoning gives us:

```
f-28 user-affect sadness
f-29 system-react sadness
f-30 thermo happiness
f-31 affect positive
f-32 concern negative
f-33 set-reaction sorry-for
For at total of 34 facts.
```

9. The cognitive processed facial display is sadness.
10. The system updates the affective thermometers.

From figure 7-5 into figure 7-6 (below) the changing is:

1. The user's situation type is negative. The system uses the last user's situation type.
2. The user's affective state is neutrality. The system does not store the word "crashed" in its sadness dictionary.
3. Since the highest thermometer is happiness, the system still surprise of the negative situation of the user. The Jess of concern of the other analyzer gives us:

```
f-28 user-affect neutrality
f-29 thermo happiness
f-30 affect negative
f-31 set-reaction fear
For at total of 32 facts.
```

4. The stimulus response facial display is surprise.
5. The system's candidate reaction affective state is disgust because my\_Eliza doesn't like negative situation of the user while she is happy (shown by the thermometer). The parser finds punctuation "!" in the disgust dictionary: [! :1].
6. The system's situation type is joking.
7. My\_Eliza makes a fun of the negative situation of the user therefore the Jess of cognitive reasoning gives us:

```
f-28 user-affect neutrality
f-29 system-react disgust
f-30 thermo happiness
f-31 affect negative
f-32 concern joking
f-33 set-reaction gloating
For at total of 34 facts.
```

8. The cognitive processed facial display is happiness.



9. The system updates the affective thermometers.

Instead of limited vocabulary in the emotive lexicon dictionary, my\_Eliza also does not have pattern rule for the user's input string nor any category about computer. Therefore the system chooses the default value with `<pattern>*</pattern>`.

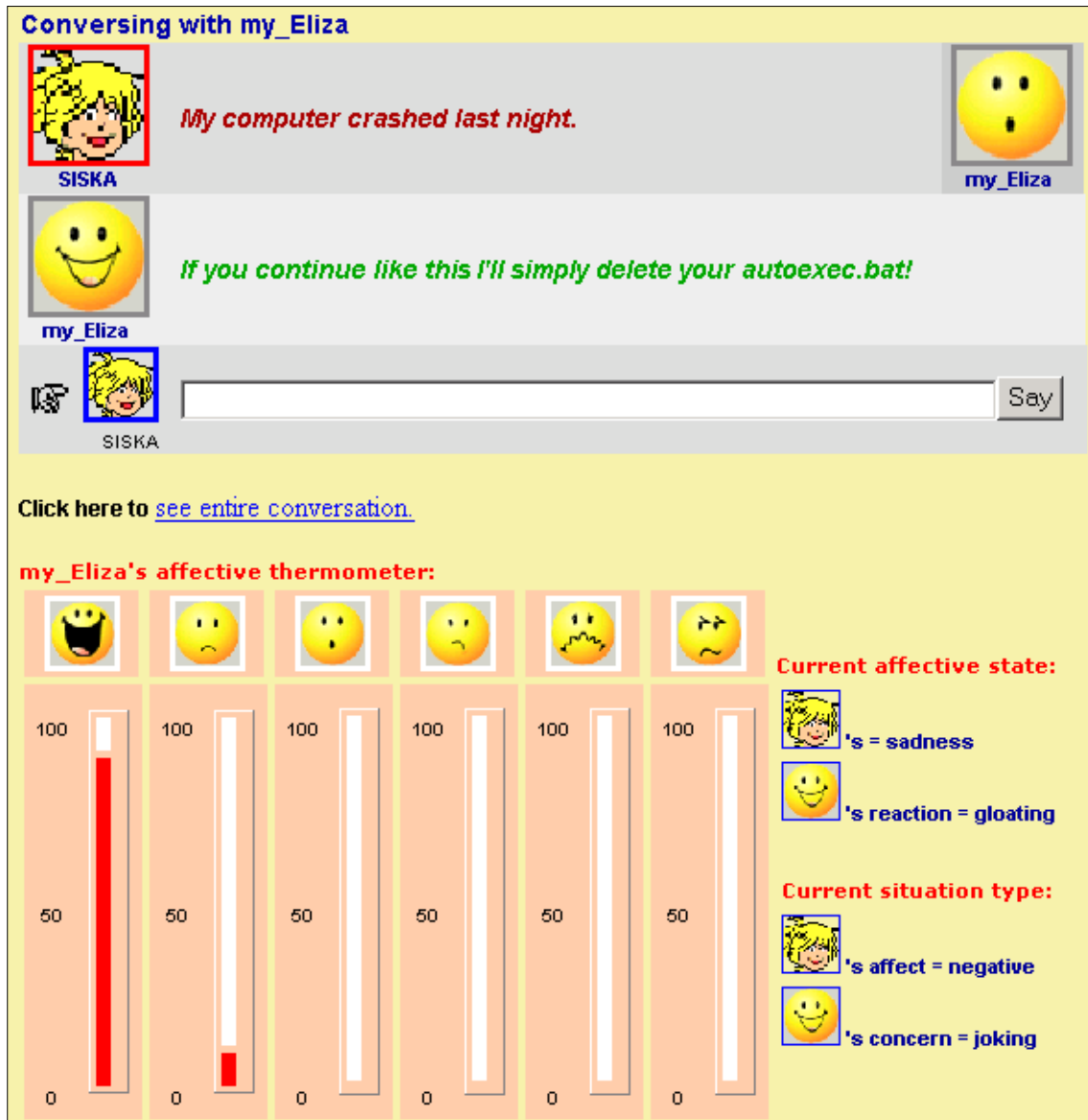


Figure 7- 6 My\_Eliza makes a mistake

From figure 7-6 into figure 7-7 (below) the changing is:

1. The user's situation type is negative.
2. The user's affective state is surprise. The system finds the words "why?" in its surprise dictionary: [WHY ?;1]
3. The highest thermometer is still happiness.
4. My\_Eliza confuses with the user's question. The Jess of concern of the other analyzer gives us:

```
f-28 user-affect surprise
f-29 thermo happiness
f-30 affect negative
f-31 user-quest
```



f-32 set-reaction uncertainty  
For at total of 33 facts.

5. The stimulus response facial display is uncertainty.

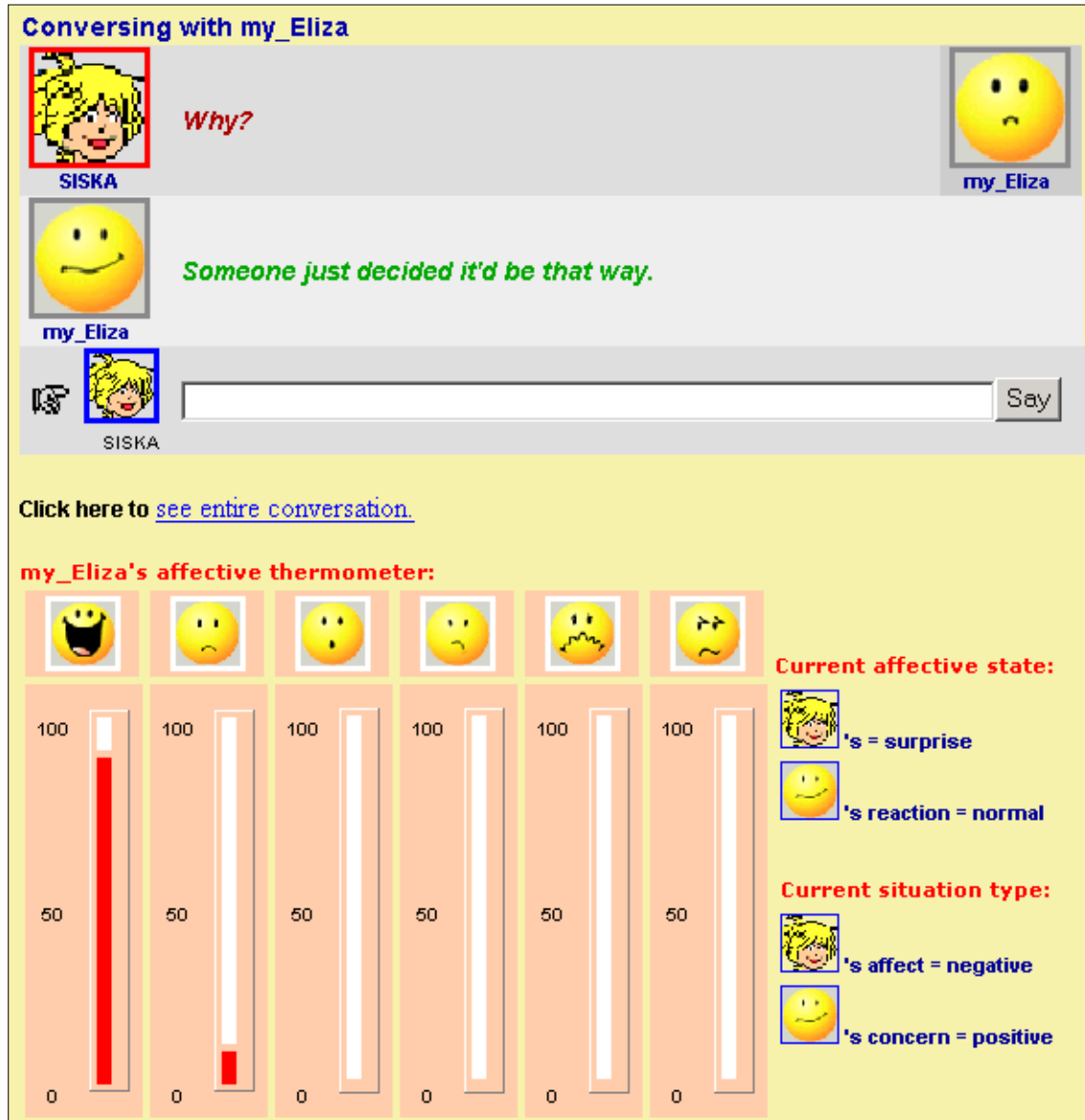


Figure 7- 7 The user is confused

6. The system's candidate reaction affective state is neutrality. The parser does not find any emotive lexicon in the reply sentence.
7. The system's situation type is positive.
8. Since My\_Eliza does understand what the user wants, she decides to react normal. The Jess of cognitive reasoning gives us:

```
f-28 user-affect surprise
f-29 system-react neutrality
f-30 thermo happiness
f-31 affect positive
f-32 concern negative
f-33 set-reaction normal
For at total of 34 facts.
```



9. The cognitive processed facial display is neutrality.
10. The system updates the affective thermometers.

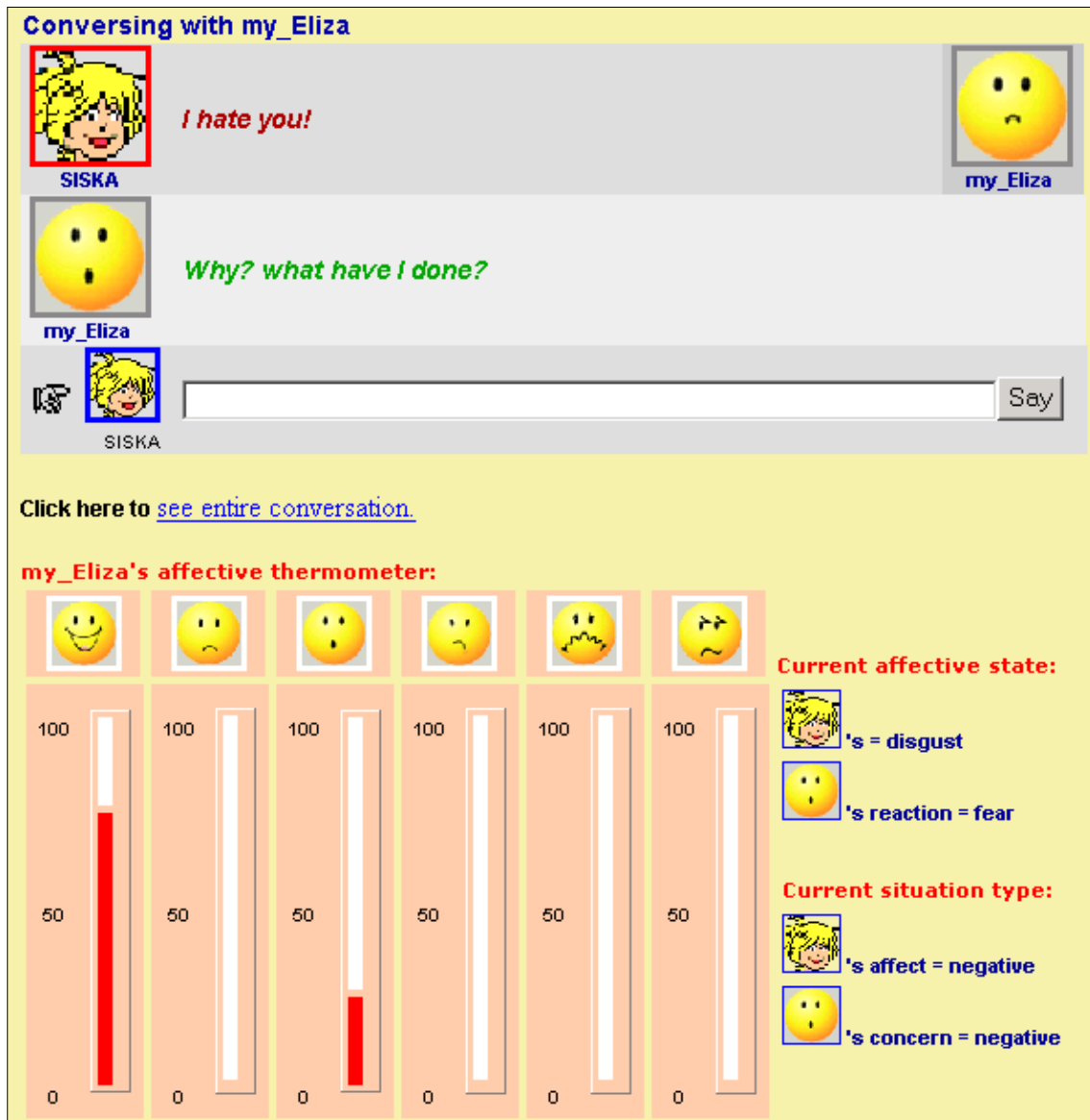


Figure 7- 8 The user hates my\_Eliza (part 1)

From figure 7-7 into figure 7-8 the changing is:

1. The user's situation type is negative.
2. The user's affective state is disgust. The system finds word "hate you" and punctuation "!" in the disgust dictionary: [HATE YOU:1] and [!:1].
3. The Jess of concern of the other analyzer gives us:

```
f-28 user-affect disgust
f-29 thermo happiness
f-30 affect negative
f-31 set-reaction uncertainty
For at total of 32 facts.
```

Since the highest thermometer is happiness, the system should be surprise with the user's reaction instead feeling uncertain. If we change the rule below:



```
(defrule concern2-disgust1
  (user-affect disgust)
  (not (affect joke))
  (thermo happiness)
  (react-uncertain ?number)
  =>
  (assert (set-reaction ?number))
)
into
  (defrule concern2-disgust13
    (user-affect disgust)
    (affect negative)
    (thermo happiness)
    (react-fear ?number)
    =>
    (assert (set-reaction ?number))
  )
```

which gives new display in figure 7-9 below.

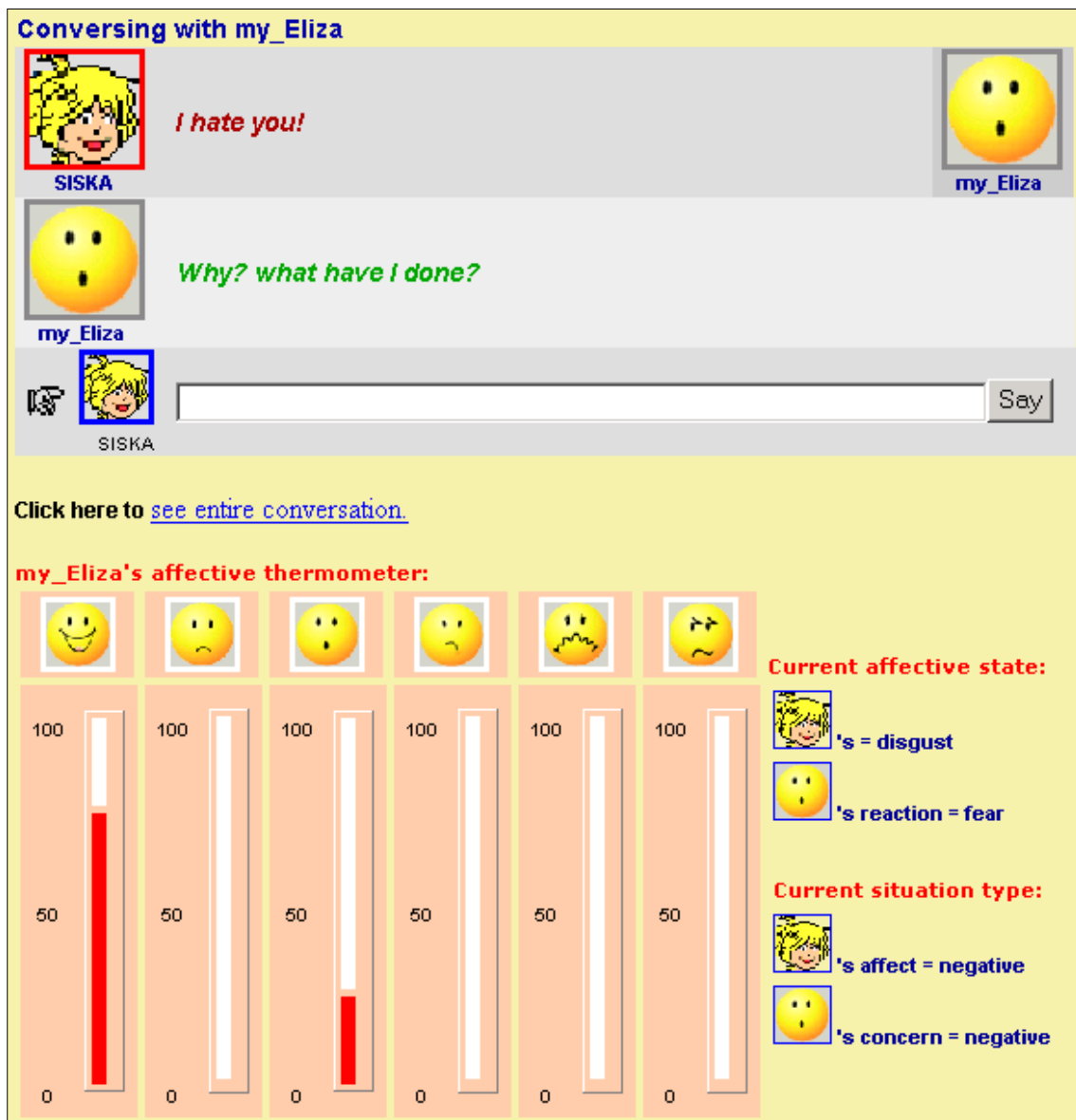


Figure 7- 9 The user hates my\_Eliza (part 2)



4. With this addition, we can expect my\_Eliza to react more natural. The Jess of concern of the other analyzer gives us:

```
f-28 user-affect disgust
f-29 thermo happiness
f-30 affect negative
f-31 set-reaction fear
For at total of 32 facts.
```

The stimulus response facial display is surprise. We can update my\_Eliza's knowledge base while the server is still running; with this benefit we can build the system's knowledge base incrementally by trial error. After the updating, in the console prompt will shown something like:

```
[15:26:19] Reloading changed AIML from "C:\ProgramD\bots\statdard\lheart.clp"
[15:26:19] Reloading changed AIML from "C:\ProgramD\bots\statdard\rheart.clp"
[15:26:20] Reloading changed AIML from "C:\ProgramD\bots\statdard\brain.aiml"
[15:26:21] Reloading changed AIML from "C:\ProgramD\bots\statdard\startup.aiml"
```

It means that the reloading process is success. As mentioned in section 6.3, the files lheart.clp and rheart.clp are the knowledge base, brain.aiml is the list of pattern rules and startup.aiml contains the emotive lexicon dictionary.

5. The system's candidate reaction affective state is surprise. The parser finds phrase "Why?" and "What have I done?" in the fear dictionary: [WHY ? :1] and [WHAT HAVE I DONE ? :1].
6. The system's situation type is negative.
7. Since the highest thermometer is still happiness, my\_Eliza is still shocked by the user's reaction. The Jess of cognitive reasoning gives us:

```
f-28 user-affect disgust
f-29 system-react fear
f-30 thermo happiness
f-31 affect negative
f-32 concern negative
f-33 set-reaction fear
For at total of 34 facts.
```

8. The cognitive processed facial display is surprise.
9. The system updates the affective thermometers.

From figure 7-9 into figure 7-10 (below) the changing is:

1. The user's situation type is negative.
2. The user's affective state is anger. The system finds phrase "never mind!" in its anger dictionary: [NEVER MIND ! :1].
3. Since the highest thermometer is happiness, my\_Eliza is surprised because the user is really angry. The Jess of concern of the other analyzer gives us:

```
f-28 user-affect anger
f-29 thermo happiness
f-30 affect negative
f-31 set-reaction fear
For at total of 32 facts.
```

4. The stimulus response facial display is surprise.
5. The system's candidate reaction affective state is fear. The parser finds word "please" and phrase "are you angry" in the fear dictionary: [PLEASE :1] and [ARE YOU ANGRY :1]. Even though it finds also phrase "not like" in the disgust dictionary ([NOT LIKE :1]), but the highest counter is still fear counter.
6. The system's situation type is negative.





7. My\_Eliza is afraid that the user is really angry and does not want to talk her any more. The Jess of cognitive reasoning gives us:

```
f-28 user-affect anger
f-29 system-react fear
f-30 thermo happiness
f-31 affect negative
f-32 concern negative
f-33 set-reaction fears-confirmed
For at total of 34 facts.
```

8. The cognitive processed facial display is fear.  
9. The system updates the affective thermometers.

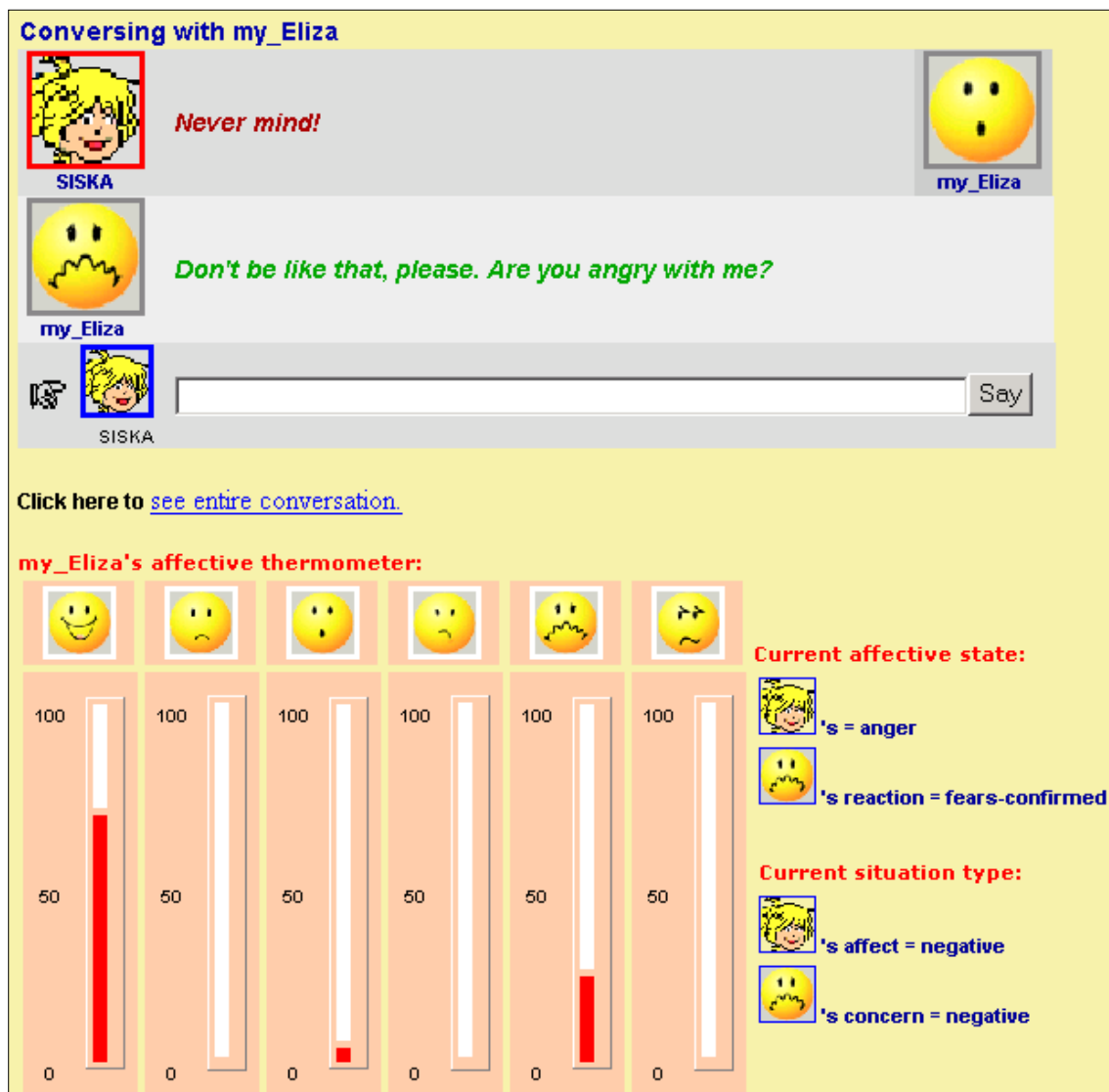


Figure 7- 10 The user gets angry

From figure 7-10 into figure 7-11 (below) the changing is:

1. The user's situation type is negative.
2. The user's affective state is disgust. The system finds word "idiot" in its disgust dictionary: [IDIOT:1].



- Since the highest thermometer is happiness, my\_Eliza is still surprise because the user hate her. The Jess of concern of the other analyzer gives us:

```
f-28 user-affect disgust
f-29 thermo happiness
f-30 affect negative
f-31 set-reaction fear
For at total of 32 facts.
```

- The stimulus response facial display is surprise.

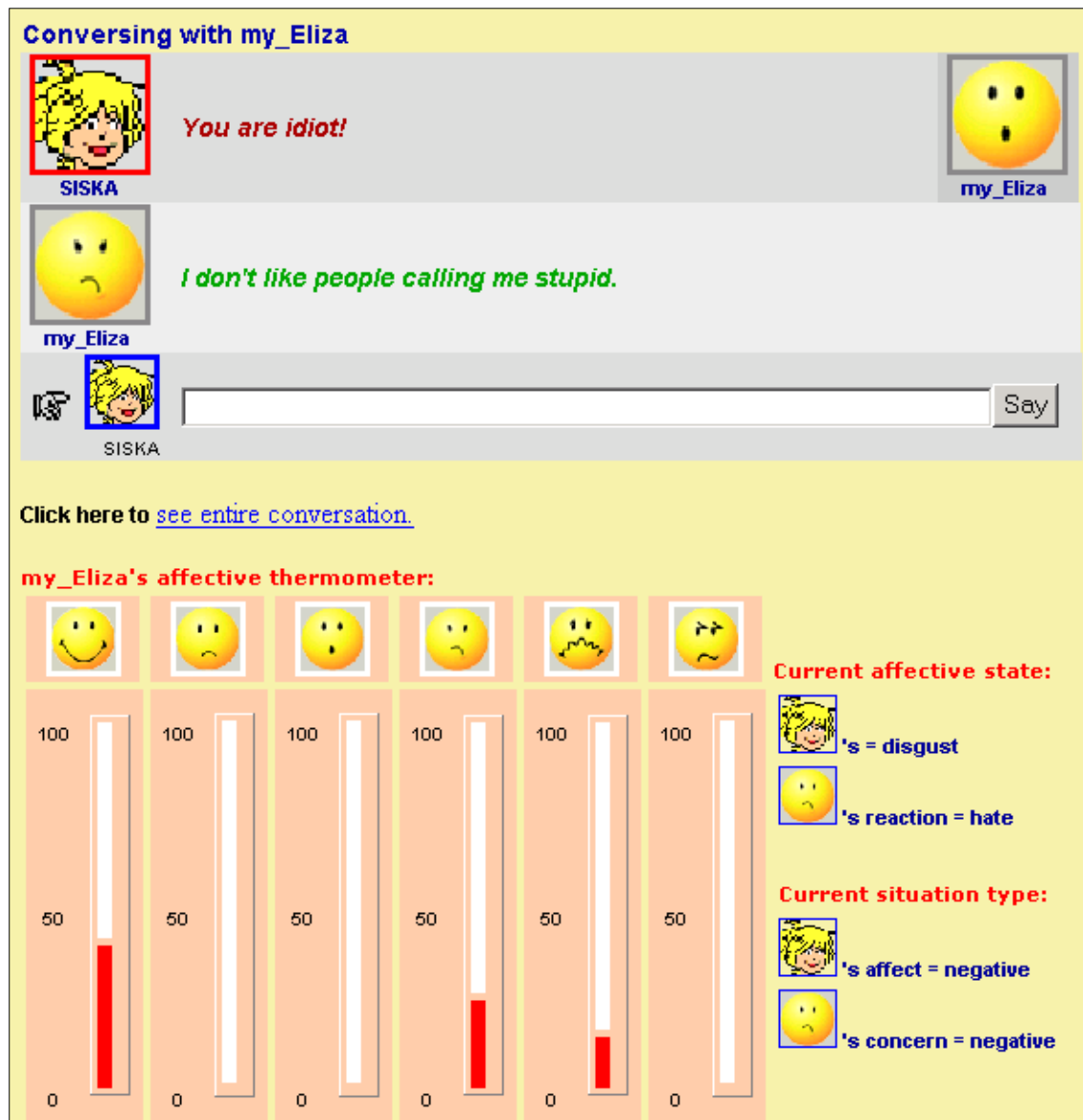


Figure 7- 11 My\_Eliza versus the user

- The system's candidate reaction affective state is disgust. The parser finds phrase "not like" and word "idiot" in the disgust dictionary: [NOT LIKE:1] and [IDIOT:1].
- The system's situation type is negative.
- The Jess of cognitive reasoning gives us:

```
f-28 user-affect disgust
```



f-29 system-react disgust  
f-30 thermo happiness  
f-31 affect negative  
f-32 concern negative  
f-33 set-reaction hate  
For at total of 34 facts.

8. The cognitive processed facial display is disgust.
9. The system updates the affective thermometers.

This condition from figure 7-7 until figure 7-11 above will not happen if the system provide the appropriate reply sentence and recognize the correct user's affective state. If we add words "crash", "crashes", and "crashed" with medium intensity to the sadness dictionary: [CRASH:2], [CRASHES:2], and [CRASHED:2]. We also add the pattern rules in table 7-2 (similar to table 5-2) about computer topic to list of pattern rule. The figure 7-6 is changed into figure 7-11. As mentioned before, we can add them while the system is running.

Table 7- 2 New categories in a topic about computer

1.	<pre>&lt;category&gt; &lt;effect name="*"&gt; &lt;pattern&gt;MY COMPUTER CRASHES *&lt;/pattern&gt; &lt;template&gt;&lt;think&gt;&lt;setaffect&gt;-&lt;/setaffect&gt;&lt;/think&gt;&lt;srai&gt;MY COMPUTER CRASHED &lt;star/&gt;&lt;/srai&gt;&lt;/template&gt; &lt;/effect&gt; &lt;/category&gt;</pre>
2.	<pre>&lt;category&gt; &lt;effect name="*"&gt; &lt;pattern&gt;MY COMPUTER CRASHED *&lt;/pattern&gt; &lt;template&gt;&lt;think&gt;&lt;setaffect&gt;-&lt;/setaffect&gt;&lt;/think&gt;&lt;srai&gt;MY COMPUTER CRASHED &lt;star/&gt;&lt;/srai&gt;&lt;/template&gt; &lt;/effect&gt; &lt;/category&gt;</pre>
3.	<pre>&lt;category&gt; &lt;effect name="-"&gt; &lt;pattern&gt;MY COMPUTER CRASHES *&lt;/pattern&gt; &lt;template&gt;&lt;srai&gt;MY COMPUTER CRASHED &lt;star/&gt;&lt;/srai&gt;&lt;/template&gt; &lt;/effect&gt; &lt;/category&gt;</pre>
4.	<pre>&lt;category&gt; &lt;effect name="-"&gt;&lt;pattern&gt;MY COMPUTER CRASHED *&lt;/pattern&gt; &lt;template&gt; &lt;think&gt;&lt;setconcern&gt;+&lt;/setconcern&gt;&lt;setaffect&gt;-&lt;/setaffect&gt; &lt;settopic&gt;COMPUTERS&lt;/settopic&gt;&lt;/think&gt;     I am sorry. Is it a PC or laptop? &lt;/template&gt; &lt;/affect&gt; &lt;/category&gt;</pre>
5.	<pre>&lt;topic name="COMPUTERS"&gt; &lt;category&gt; &lt;affect name="-"&gt; &lt;that&gt;IS IT A PC OR A LAPTOP&lt;/that&gt; &lt;pattern&gt;IT IS A PC&lt;/pattern&gt; &lt;template&gt;&lt;think&gt;&lt;setconcern&gt;+&lt;/setconcern&gt;&lt;/think&gt;     I had a PC before, have you ever formatted it? &lt;/template&gt; &lt;/affect&gt; &lt;/category&gt;  &lt;category&gt; &lt;affect name="-"&gt; &lt;that&gt; IS IT A PC OR A LAPTOP&lt;/that&gt; &lt;pattern&gt;IT IS A LAPTOP&lt;/pattern&gt; &lt;template&gt;&lt;think&gt;&lt;setconcern&gt;+&lt;/setconcern&gt;&lt;/think&gt;     I got a laptop two years ago. We have to be more careful with     &lt;set_it&gt;laptop&lt;set_it&gt;. I think you better call computer reparation     service. &lt;/template&gt; &lt;/affect&gt; &lt;/category&gt; &lt;/topic&gt;</pre>

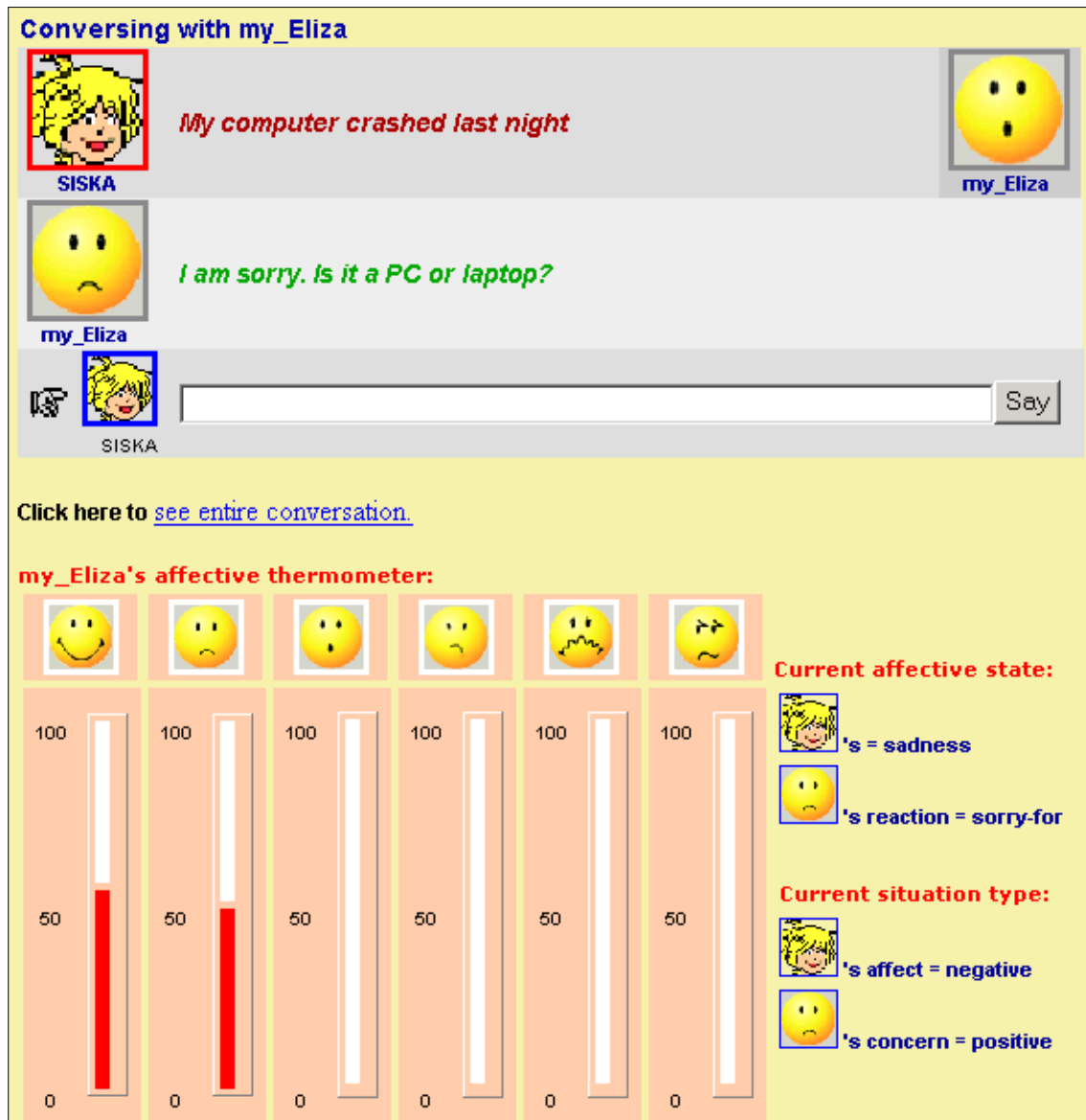


Figure 7- 12 My\_Eliza feels sorry for the user

From figure 7-5 into figure 7-12 the changing is:

1. The user's situation type is negative.
2. The user's affective state is sadness. The parser finds word "crashed" in the sadness dictionary: [CRASHED:1].
3. The highest thermometer is still happiness.
4. My\_Eliza is still surprised. The Jess of concern of the other analyzer gives us:

```
f-28 user-affect sadness
f-29 thermo happiness
f-30 affect negative
f-31 set-reaction fear
For at total of 32 facts.
```

5. The stimulus response facial display is surprise.
6. The system's candidate reaction affective state is also sadness. The parser also finds word "sorry" in the sadness dictionary: [SORRY:1].



7. The system's situation type is positive.
8. The Jess of cognitive reasoning gives us:
  - f-28 user-affect sadness
  - f-29 system-react sadness
  - f-30 thermo happiness
  - f-31 affect negative
  - f-32 concern positive
  - f-33 set-reaction sorry-forFor at total of 34 facts.
9. The cognitive processed facial display is sadness.
10. The system updates the affective thermometers.

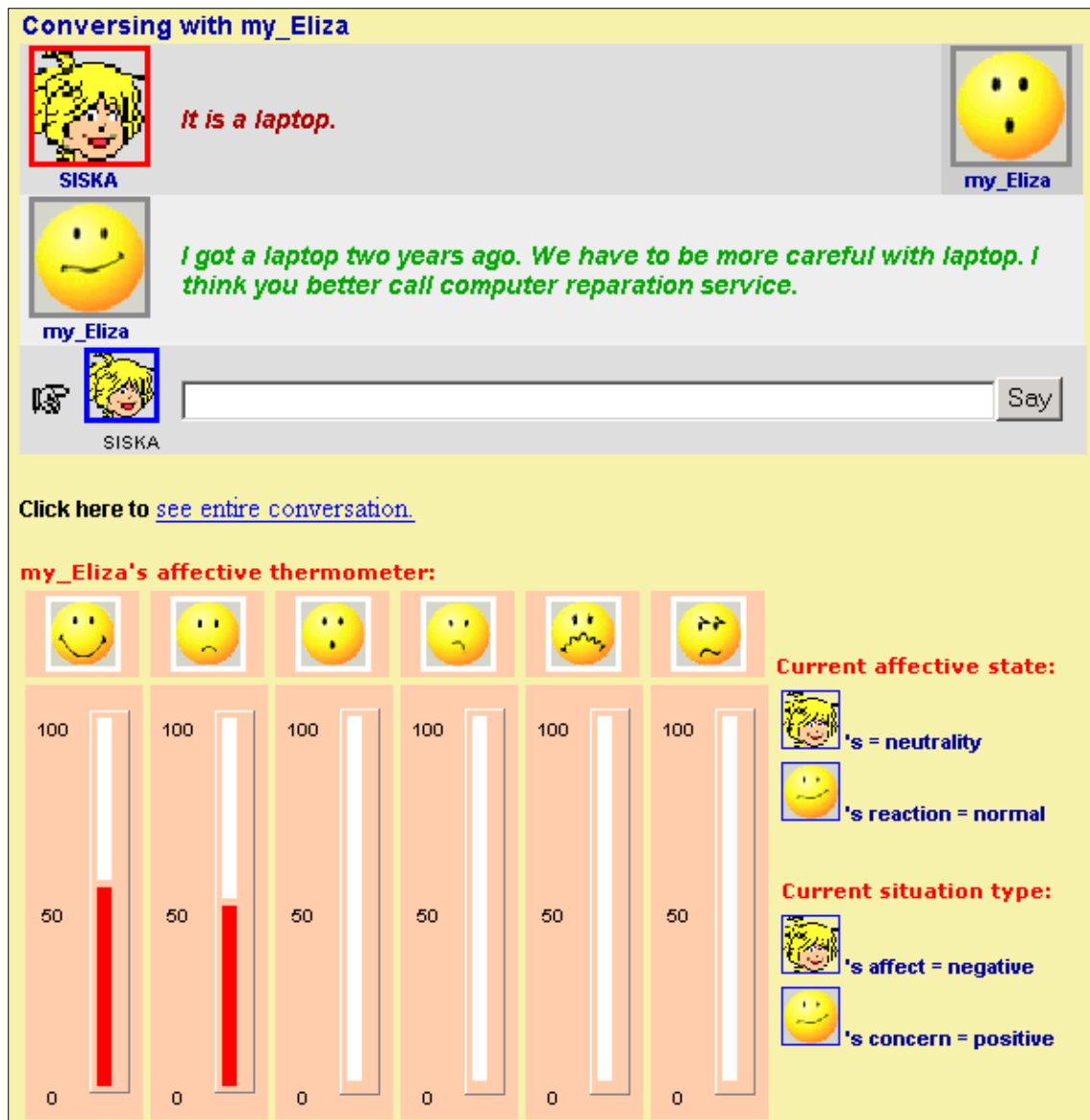


Figure 7- 13 My\_Eliza in a topic about computer.

- From figure 7-12 into figure 7-13 the changing is:
1. The user's situation type is negative.
  2. The user's affective state is neutrality. The parser does not find any emotive lexicon in the reply sentence.



3. The highest thermometer is still happiness.
4. My\_Eliza is still surprise because the negative situation of the user contradicts with her happiness. The Jess of concern of the other analyzer gives us:

```
f-28 user-affect neutrality
f-29 thermo happiness
f-30 affect negative
f-31 set-reaction fear
For at total of 32 facts.
```

5. The stimulus response facial display is surprise.
6. The system's candidate reaction affective state is happiness. The parser also finds word "better" in the happiness dictionary: [BETTER:1].
7. The system's situation type is positive.
8. The Jess of cognitive reasoning gives us:

```
f-28 user-affect neutrality
f-29 system-react happiness
f-30 thermo happiness
f-31 affect negative
f-32 concern positive
f-33 set-reaction normal
For at total of 34 facts.
```

9. The cognitive processed facial display is normal.
10. The system updates the affective thermometers.

From figure 7-13 into figure 7-14 (below) the changing is:

1. The user's situation type becomes positive.
2. The user's affective state is neutrality. The parser does not find any emotive lexicon in the reply sentence.
3. The highest thermometer is still happiness.
4. My\_Eliza is glad that the situation is positive again. The Jess of concern of the other analyzer gives us:

```
f-28 user-affect neutrality
f-29 thermo happiness
f-30 affect positive
f-31 set-reaction normal
For at total of 32 facts.
```

5. The stimulus response facial display is neutrality.
6. The system's candidate reaction affective state is happiness. The parser also finds words "good" and "pleasure" and phrase "good luck" in the happiness dictionary: [GOOD:1], [PLEASURE:1], and [GOOD LUCK:1].
7. The system's situation type is positive.
8. The Jess of cognitive reasoning gives us:

```
f-28 user-affect neutrality
f-29 system-react happiness
f-30 thermo happiness
f-31 affect positive
f-32 concern positive
f-33 set-reaction joy
For at total of 34 facts.
```

9. The cognitive processed facial display is normal.
10. The system updates the affective thermometers.



My\_Eliza still remember the user's name and use it in her reply sentence. Note, the system processes the user's input string one by one sentences divided by namely "splitter" (see section 4.2.2.1). In this prototype, the splitter contains of three string, such as: ".", "!", and "?".

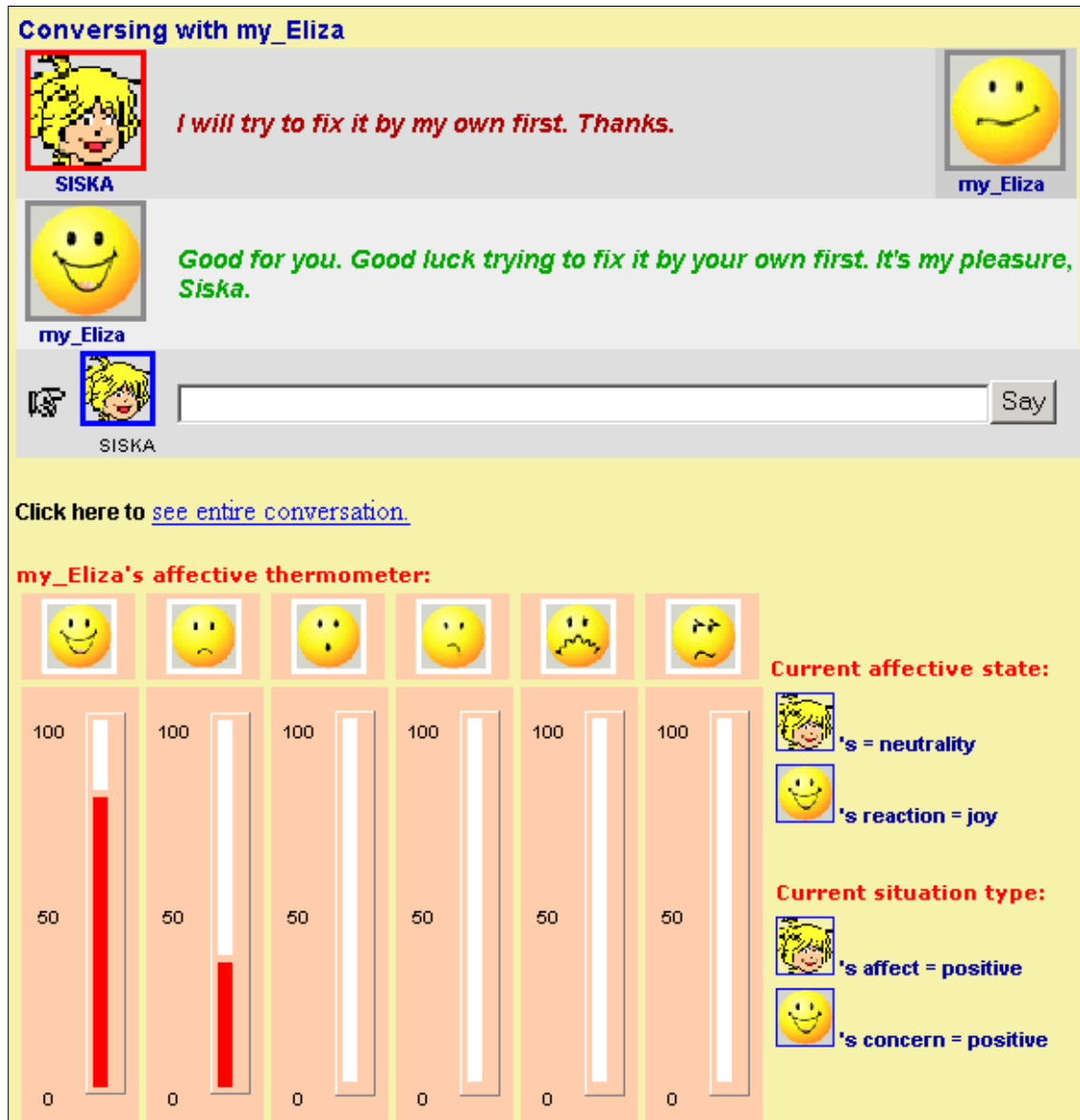


Figure 7- 14 My\_Eliza feels happy helping the user

Those sequential figures above show us that my\_Eliza reasons the emotion from a single user's input string. The system does not care about the user's feeling in the previous dialog. In every dialog state, the system reacts to and replies the user based on stimulus response. It seems that my\_Eliza follows "her feeling" than play her role as psychoanalyst. This limitation will not happen if my\_Eliza has a fully *cognitive reasoning layer* and *pragmatic analysis*. Those will implemented in My\_Eliza prototype-2, the last step in building a multimodel communication system (see three major (incrementally) implementation layers in figure 4-1). This prototype will contain a full version of emotion recognition and facial display generation as has been explained in global design in chapter 4. The system is able to reasons the emotion in the prompt text based on the anaphora of the conversation. The system can also react based on its goal, status, and preference (GSP). In this prototype, "react based on status" means the system is entitled to particular mood and/or





characteristic of personality. System's preferences means that the system has list of something it likes or dislikes. As mentioned in section 4.3, the system has several goals and reacts based on one of this goals means during the conversation, it will try to achieve it. To be fair, one of the goals of the system is to understand more about the user. In this prototype, the system stores user's preferences in the database therefore it "understands" more about the user. This database does not only contain what he/she likes or dislikes but also his/her personal data, such as: gender, birthday, family matter and etc. This personal data is derived from user's text prompt during the conversation. For these new capabilities, cognitive reasoning is important for the system since it means the system's reaction affective state is influenced by the whole conversation flow not only a single prompt of a dialog state. The design and implementation of my\_Eliza prototype-2 are left out as the recommendation of the system development in future.



## Chapter 8 Conclusion

In this investigation we have examined a combination of a NLP approach and a nonverbal communication approach for designing and developing a multimodal communication system, my\_Eliza system. These approaches involve running the system to conduct conversations between the system and human users. These approaches were based on emotion reasoning and natural language processing to reply each string input of the users and show two appropriate nonverbal facial signals: (1) as stimulus response to the content of user's string input and (2) as cognitive processed result to convey the reply sentence. We assumed that my\_Eliza system could serve as an excellent QA system and as a test bed for applying those approaches. Typically, it is difficult to analyze human emotion from text. This is due to the fact that the emotion of the user might be implicitly beyond the text. Human conversation contains many topics and its flow can easily be changing. Therefore the analysis also has to cope with this inflection. Furthermore, the system's design enables the introduction of human emotional semantic of the text to the system. Upon modeling the system, we make use of the reference of how human reacts to certain texts emotionally.

This chapter contains the conclusions of this thesis work, which is organized as follows. First, an evaluation of the results based on the problem definition (as described in section 1.4) is summarized in section 8.1. Finally, the future work of what can be done to further development of the system is explained in section 8.2.

### 8.1 Evaluation

At the start of the investigation a literature survey has been done about QA systems and nonverbal communication systems. Firstly, this has led to insights about natural language processing of a QA system, degree of intelligence, and conversation flow regarding to topic focusing and history reference. Secondly, the survey also led to insights about human emotion classification, emotion recognition process and facial display generation. Additionally, studying the literature about the emotion recognition process leads to the conclusion that none of the research is particularly in the work of recognizing emotion in a QA system. However, from the work of researchers in the fields of multi agents system, emotion recognition from human speech intonation, automated character animation system and communicative facial display system, could give inspiration which allows us to fit in our approach with the fields.

Unfortunately there have not been any studies or analysis about other QA systems spreading in the Internet, instead of Eliza and A.L.I.C.E. We have written down some general considerations about the QA system's characteristic to Eliza and A.L.I.C.E, but a thorough analysis is not a purpose of the investigation. In an attempt to acknowledge the complexities of automatic emotion recognition in a QA system in general, we have studied some papers on identifying eliciting-emotion factors from texts approaches. These approaches apply can be applied how to introduce emotion types of certain texts to the system. Approaches to identify eliciting-emotion factors with information about the words or phrases and their emotion types however still need human manual annotations. These can be very complicated since it is a gray area and may raise ambiguity. Each person may have his/her own perspective about correspondences between situations and certain emotion types. This leads us to believe that an intuitive solution for recognizing emotion from the text is to be found using three approaches, such as:

- Introducing the system to current situation type of the sentence in its memory structure, whether pleasant or unpleasant (in this thesis we use the terms positive and negative).
- Parsing the text against a dictionary that contains words or phrases with their possible emotion types and a degree of intensity.



- Giving certain goals and preferences to the system, and storing the current system's affective state. The emotion-eliciting factors from this approach are identified from whether current condition violates the system's goals, preferences and status.

We have modeled the system to be able to exhibit behavior patterns of current system's affective state as reaction to the user's string input, which resulted by reasoning of those emotion-eliciting factors. Furthermore, there are two things the literature survey and seeking existing system have resulted in. First, as the starting point for the system's dialog box model we use the work of Richard Wallace, A.L.I.C.E system and its AIML form for its memory structure. A.L.I.C.E system has a much bigger memory structure than Eliza, since each unit of its memory structure also contains reference to current topic of conversation and system's previous reply sentence. Moreover, since AIML is the extended form of XML, it is possible for A.L.I.C.E to store information about the user and the content during conversation. Second, many researchers have proposed emotion classifications and it is not trivial to choose the best one for this system. Therefore we use four methods of emotion classification and we use them fit to the need of the process in the system:

- *Reddy's basic emotions*, we use it to label the situation type of sentence.
- *Ekman's seven universal emotion types*, we use it in lexicon classification in emotive lexicon dictionary, facial displays classification in facial display selector, and thermometers in intensity analysis. This classification was used if we found it is too ambiguous to use twenty-four emotion types of OCC's theory.
- *Schlosberg's circular two dimension emotion types*, we use it since we use the approach of Hendrix and Ruttkay's discrete values between six Ekman's universal emotion types as summation factor to define the threshold of an emotion activation.
- *Twenty-four emotion types of OCC's theory*, we use it to determine system's reaction affective state. We use it instead Ekman's universal emotion types to get a broaden possibility of system's behavior for the next development.

This way of modeling the system has some important advantages. First of all, it comes close to explaining how human recognizes emotion from texts. Secondly, a pragmatic advantage of using AIML form and preference rules to exhibit system's behavior is that the systems' memory structure and its behavior can be extended easily. By adding new AIML categories with certain topics and correlation with possible previous' reply sentences, the system can "speak" more. By adding new eliciting-emotion factors, the agent expresses a new overall behavior.

We have chosen to use a rule-based approach for specifying the system's reaction affective state. It turns out that for it, we can specify a lot of rules, which result from the characteristic emotion-eliciting factors. We can easily define a number of possible reaction affective states, in this thesis work we have added two emotion types: (1) uncertainty and (2) normal, to the twenty-four emotion types of OCC's theory.

Our design reflects the desire to build a generic tool for investigations. The basic functionality of the design has been implemented successfully, except for *pragmatic analysis* in NLP layers. The first prototype, my\_Eliza prototype-1, currently allows for human-computer interactive locally or via the Internet. The most important part of my\_Eliza prototype-1 is the implementation of a framework for a multimedia communication in which they have:

- *Natural language processing.*
  - Users' string input perception.
  - Pattern matching operation against the system's memory structure based on the input pattern, the current topic and the previous reply sentence.
  - Automatic reply sentence.



- *Facial display generation*
  - Emotion eliciting factors extraction from user's string input, system's reply sentence and situation type of both of them.
  - Observation of system's affective state during conversation using six emotion types thermometers.
  - Emotion recognition from the eliciting-factors and the system's knowledge bases. The knowledge bases take the form of known facts about the emotion eliciting-factors and current system's affective state.
  - Automatic facial display generator based on the system's reaction affective state resulted by emotion recognition.

A QA system is a human-computer interaction system that can converse with many open discussions. As stated before, there has not been any analysis of emotion recognition in a QA system. Therefore we advocate using an incremental approach to build up the system, its memory structure and also its knowledge base, which can be used for studies and experiments about a multimodal communication. The client-server model that we use based on A.L.I.C.E system, is a framework, which can be used for experiments regarding QA systems and nonverbal communication systems. An interesting property of the framework is that the possibility to add new memory structure units and system's knowledge bases while it runs. This way we can build up the library during conversation in order to get natural interaction in the future. Those incremental stages can not be achieve if we did not design the model based on a blackboard system, which allows all message passing processes between layers through the blackboard. With this approach, we can build more modular system development and easier to maintain the system than procedural approach.

We have tested my\_Eliza prototype by letting the system interact with a human user. The experiments have resulted in some valuable ideas about our emotion recognition approach. The result of our experiments support our intuitive solutions how to extract emotion-eliciting factor. Unfortunately, a fully testing of the system for multi-user and multi-browser in the Internet has been put as a recommendation to future work. Nevertheless, we have shown this first prototype as a proof of concept to combine a QA system and nonverbal generation into a multimodal communication system.

## **8.2 Future Work**

The future work that needs to be done to continue the investigation can be divided into several categories. First of all, a fully testing of the system can be done. By running the system in the Internet, we may receive a lot of feed back from the users and a lot of dialogs are recorded. Next, several improvements to the first prototype can be made. As stated before, we have implemented the basic functionality of the design. An important part that still has to be done is adding more memory structures, more lexicons in emotive lexicon dictionary, and more possible facial displays. We can expand the system's memory structure and emotive lexicon dictionary using the collection of recorded dialogs and dialogs analyses. Moreover, these additions should be a means to more natural interaction between human users and the system.

One of the aspects of the system that can be extended is the rule-sets in system's affective knowledge bases. First of all we want to have more emotion eliciting-factors in the rule-sets. Extra rules are needed for new eliciting factors. Furthermore, to make that the system has own goals, status and preferences; the system should be made more flexible by adding new emotion eliciting-factors regarding the parameter settings for the system's behavior. Examples of such parameters are system's goals definitions and determinations and system's preferences blameworthiness or praiseworthiness. One of recommendation how to determine the system's goals is by classification of the dialog state (both string input and reply sentence) based on a dialog scheme adopted from



Carletta et. al. [CAR95]. For example, by distinguishing a dialog state as a certain dialog act like a question, statement, acknowledgement, or pause, the system has to know which goal to pursue.

The prototype that we have implemented, still do not store the users' information. This requires several works to be done in the system's memory structure. For example by adding <set> tags in the memory structure units to get certain information about the user or the current conversation. Also by adding <get> tags in them to use the information during conversation. Hereby, the system needs pragmatic analysis to give more correct reply sentence to the system according to the user's personal data. The systems can also communicate information across users, for example for gossiping.

Using the rule-sets that we have implemented in the system do not know the correlation emotion of current dialog and the emotion of entire conversation content. This requires specific knowledge about the system's goals, status, and preferences and also thermometers of the user's affective state that has to be observed during conversation. The rule-sets that we have implemented is not enough to covering this circumstance, therefore extra rules have to be used. The rules-sets could also be made temporal. We can make specific rules for the opening of the conversation, during the discussion or the end of the discussion. The system also should have the ability to learn from conversation history. These additions to the system will be valuable assets to add new memory structure units and to add rule-sets of the system to generate its affective knowledge bases autonomous.

Additionally the server interface should have extended functionality to show the thinking process of the system. Realistic virtual environments not only include believable appearance and simulation of the virtual world but also imply the natural representation of participants. That can be fulfilled by visualization of human character embodiment with animation. More complex virtual human character embodiment increases the natural interaction within the environment. The users can feel more natural perception each other, increases their sense of being together and thus the overall sense of shared presence in environment. To be maximally realistic figure, animated human characters must be able to express many different kinds of emotion. The characters need capability to express the emotion flow within conversation. They have to be able to convey many different kinds of emotion as different social situation arise. It needs to explore several ways so that real-time, animated, and virtual human characters can be given more intelligence and communication skills, therefore they can act, react, make decisions, and take initiatives. Current prototype uses only a simple dynamic HTML, we can develop the interface using Java APPLETS (as first step) therefore the users can see that the stimulus response facial display to their string input is processed and displayed first before the system formulates its reply sentence and cognitive processed facial display. In a similar fashion, since my\_Eliza should be able to communicate with a broad range of conversation topics, the system should be able to visually support these conversations with an equally broad range of emotion and expressions behaviors.

Those recommendations above lead us to design and develop more natural my\_Eliza system as a multimodal communication system. Those works have to be done to develop the third increments of implementation layer of my\_Eliza system into my\_Eliza prototype-2.



## References

- [ALM92] Alm, Norman, et al., *Computer and People with Disabilities, Prediction and Conversational Momentum in an Augmentative Communication System*, Communication of ACM, Vol. 35 No.5, p45-55, May 1992.
- [BAZ01] Bazzan, A.L.C, Bordini, R.H., *A Framework for the Simulation of Agents with Emotions, Report on Experiments with the Iterated Prisoner's Dilemma*, AGENT'01, Communication of ACM, p292-p299, Canada, 2001.
- [CAR80] Carbonell Jr, Jaime G., *Proceedings of the theoretical issues in natural language processing-2, Intentionality and Human Conversations*, p141-148, Department of Computer Science, Yale University – New Haven, Connecticut, July 1978.
- [CAR95] Carletta, J., Isard, A., Isard, S., Kowtko, J., Doherty-Sneddon, G., Anderson, A., *The Coding of Dialogue Structure in a Corpus, in Corpus-based Approaches to Dialogue Modeling*, 9<sup>th</sup> Twente Workshop on Language Technology, p25-p34, University of Twente, 1995.
- [CAS94] Cassell, J., Pelachaud, C., Badler, N., Steedman, M., Achorn, B., Becket, T., Douville, B., Prevost, S., Seah, C., and Stone M., *Animated Conversation: Rule Based Generation of Facial Expression, Gesture and Spoken Intonation for Multiple Conversation Agents*, in SIGGRAPH'94, 1994.
- [CHO65] Chomsky, N., *Aspects of the Theory of Syntax*, MIT Press, Cambridge, Mass., 1965.
- [COW95] Cowley, J.S. & MacDorman, K.F., *Nautilus, Chatterbot FAQ*, [http://www.kartefakt.de/jabberwock/faq\\_en.html](http://www.kartefakt.de/jabberwock/faq_en.html), 1995.
- [EKM75] Ekman, P. and Friesen, W.V., *Unmasking the Face*, Prentice Hall, New Jersey, USA, 1975.
- [ELL92] Elliott, Clark, *Thesis Report, The Affective Reasoner: A Process Model of Emotions in a Multi-Agent System*, Ph.D. Dissertation, Northwestern University, The Institute for the Learning Sciences, Technical Report No.32, 1992.
- [ELL93] Elliott, Clark, & Siegle, Greg, *Variables Influencing the Intensity of Simulated Affective States*, In AAAI Technical Report for Spring Symposium on Reasoning about Mental States: Formal Theories and Applications, 58-67, American Association for Artificial Intelligence, Stanford University, March 23-25, Palo Alto, CA, 1993.
- [ELL93a] Elliott, Clark, *Using the Affective Reasoner to Support Social Simulations*. In Proceeding of the 13<sup>th</sup> International Joint Conference on Artificial Intelligence, 194-200, Chambery, France: Morgan Kaufmann, 1993.
- [ELL94] Elliott, Clark, & Melchior, Ernst, *Getting to the point: Emotion as a Necessary and Sufficient Element of Story Construction*, <http://condor.depaul.edu/~elliott/ar.html>, 1994.
- [ELL95] Elliott, Clark, *Draft: Components of two-way emotion communication between humans and computers using a broad, rudimentary, model of affect and personality*, Depaul University, <http://condor.depaul.edu/~elliott/ar.html>, 1995.
- [ERI98] Erricsson, Hans-Eric, & Penker, Mangus, *UML Toolkit*, John Wiley & Sons, Inc., USA, 1988.
- [DES00] Desmet, P.M.A, *Emotion Trough Expression; Designing mobile telephones with an emotional fit*, Reprot of Modeling the Evaluation Structure of KANSEI, Vol. 3, p103-p110, <http://pubwww.tudelft.nl/promood/aioio/pieter/pieter.htm>, 2000.
- [FRA96] Franklin, S., and Graesser, A., *Is It an Agent or just a Program?: A Taxonomy for Autonomous Agents*, Proceeding of the 3<sup>rd</sup> International Workshop on Agent Theories, Architectures, and Lanugages, <http://www.msci.memphis.edu/~franklin/AgentProg.html>, Springer-Verlag, 1996.
- [FRI00] Friedman-Hill, Ernest, Jess: *Rule Engine of the Java Platform*, Sandia National Laboratory, <http://herzberg.ca.sandia.gov/jess/>, USA, 2000.
- [HAN02] Hansen, Dexter A., *Flowcharting Help Page (Tutorial)*, <http://home.att.net/~dexter.a.hansen/flowchart/flowchart.htm>, 2002
- [HAR85] Harris, Mary D., *Introduction to Natural Language Processing*, Prentice Hall, USA, 1985.





- [HEN84] Hendix, J. & Ruttkay, Zs., *Exploring the Space of Emotional Faces of Subjects without Acting Experience*, ACM Computing Classification System: H.5.2, I.5.3, J.4, <ftp://ftp.cwi.nl/pub/CWIreports/INS/INS-R0013.ps.Z>, 1998.
- [HOL94] Holland, Norman N., *Eliza Meets the Postmodern*, E-Journal, Vol. 4 No.1, ISSN 1054-1055, <http://www.hanover.edu/philos/ejournal/archive/ej-4-1.txt>, April 1994.
- [KIN97] King, Donnell, A., *Nonverbal Communication*, <http://www2.pstcc.cc.tn.us/~dking>, 1997.
- [LAV96] Laven, Simon, *The Simon Laven Page, Chatterbot Central*, [www.simonlaven.com](http://www.simonlaven.com), 1996.
- [LEE99] Lee, Kwanyong, *Integration of Various Emotion Eliciting Factors for Life-Like Agents*, ACM Multimedia'99, Part 2, p155-p158, 1999.
- [LOE91] Loebner, Huge, *Home Page of The Loebner Prize - "The First Turing Test", What is the Loebner Prize?*, <http://www.loebner.net/Prizef/loebner-prize.html>, 1991.
- [NAK99] Nakatsu, Ryohei, et al. *Emotion Recognition and Its Application to Computer Agent with Spontaneous Interactive Capabilities*, Creativity and Cognition 99, Communication of ACM, p135-p143, UK, 1999.
- [PAN01] Pantic, Maja, *Facial Expression Analysis by Computational Intelligence Techniques*, Ph.D. Dissertation, Delft University of Technology, 2001.
- [PEL94] Pelachaud, C., Badler, N.L., & Steedman, M., *Generating Facial Expression for Speech*, Dept. of Computer and Information Science, University of Pennsylvania, 1994.
- [PRE98] Pressman, Roger S., *Software Engineering: A Beginner's Guide*, Prentice Hall, 1998.
- [PRE01] Prendinger & Ishizuka, *Social Role Awareness in Animated Agents*, AGENTS'01 ACM, p270-p276, 2001.
- [PUP93] Puppe, Frank, *Systematic Introduction to Expert System, Knowledge Representation and Problem-Solving Methods*, Springer-Verlag Berlin Heidelberg, 1993.
- [RED01] Reddy, W.M., *The Navigation of Feeling, A Framework for the History of Emotion*, Cambridge University Press, Edinburgh, 2001.
- [RUS80] Russell, J.A., *A Circumplex Model of Affect*, Journal of Personality and Social Psychology, 39(6), 1161-1178, 1980.
- [RUS95] Russell, Stuart, & Norvig, Peter, *Artificial Intelligence, A Modern Approach*, Prentice Hall Series in Artificial Intelligence, 1995.
- [SES94] Sestito, Sabrina, & Dillor, Tharam S., *Automated Knowledge Acquisition*, Prentice Hall, Australia, 1994.
- [SCH52] Schlosberg, H. A., *A Description of Facial Expressions in Terms of Two Dimensions*, Journal of Experimental Psychology, Vol. 44, p229-p237, 1952.
- [SCH00] Schiano, Ehrlich, Rahardja & Sheridan, *Face to Interface: Facial Affect in (Hu)Man and Machine*, CHI Letters, Communications of the ACM, Vol. II No. 1, p193-p200, 2000.
- [SIM70] Simmons, R.F., *Computational Linguistic, Natural Language Question Answering System: 1969*, Communications of the ACM, Vol. XIII No. 1, p15-p29, January 1970.
- [SLU00] Slugoski, B.R. & Hilton, D.J., *Handbook of language and social psychology: Conversation*, Guide de Psychologie Sociale, Richerches, <http://www.multimania.com/psychosocial>, 2000.
- [TAK93] Takeuchi & Nagao, *Speech Dialogue with Facial Displays: Multimodal Human-Computer Conversation*, Agent Interfaces, Section VII, p572-p579, 1993.
- [TUR50] Turing, A.M., *Mind, A Quarterly Review of Psychology and Philosophy, Computing Machinery and Intelligence*, Vol. LIX No. 236, p.433, October 1950.
- [VEL97] Velasquez, J.D., *Modelling Emotions and Other Motivations in Synthetic Agents*, American Association for Artificial Intelligence (AAAI), [www.aaai.org](http://www.aaai.org), 1997.
- [WEI66] Weizenbaum, J., *ELIZA - A Computer Program for the Study of Natural Language Communication between Man and Machine*, Communication of the ACM 9(1): p36-p45, 1966.
- [WEI67] Weizenbaum, J., *Computational Linguistics, Contextual Understanding by Computer*, Communication of the ACM, Vol. X No. 8, p474-p480, August 1967.
- [WEI76] Weizenbaum, J., *Computer Power and Human Reason*, W.H. Freeman and Co., San Fransisco, USA, 1976.
- [WIE99] Wierzbicka, Anna, *Emotional Universal*, Language Design 2, p23-p99, Australia, 1999.





[WOO95] Wooldridge, M., & Jennings, N.R., *Intelligent Agents: theory and practice*. Knowledge Engineering Review 10 (2), 1995.

### Chatterbot Related Sites

- [ADA80] Adams, Douglas, *Starship Titanic*, <http://www.starshiptitanic.com/>, 1980.
- [BUS01] Bush, Noel, *AIML Specification ver. 1.0.1*, <http://Alicebot.org/TR/2001/WD-aiml/>, 2001.
- [BUS02] Bush, Noel, *Getting Started with Program D*, <http://Alicebot.org/resources/programd/readme.html>, 2002.
- [COL73] Colby, Kenneth, *Parry*, <http://maiw.com/>, 1973.
- [ETT84] Etter & Chamberlain, *Racter*, <http://robotwisdom.com/ai/racterfaq.html>, 1984.
- [GAR97] Garner, Robby, *Fred*, <http://www.cybermecha.com/fred.html>, 1997.
- [HUT95] Hutchens, Jason, *How Hex Works*, <http://www.amristar.com.au/~hutch>, 1995.
- [MAU94] Mauldin, Michael, *Julia*, <http://www.lazytd.com/lti/julia>, 1994.
- [RIN01] Ringate, Thomas, *AIML Reference Manual*, <http://Alicebot.org/documentation/aiml-reference.html>, 2001.
- [WAL95] Wallace, Richard, *Alicebot*, <http://www.Alicebot.org>, 1995.
- [WAL00] Wallace, Richard, *A.L.I.C.E and AIML Specification, Don't Read me*, <http://Alicebot.org/articles/wallace/dont.html>, 2000.
- [WAL01] Wallace, Richard, *AIML Pattern Matching Simplified*, <http://Alicebot.org/documentation/matching.html>, 2001.
- [WEI86] Weintraub, Joseph, *PC Therapist*, <http://www.loebner.net/Prizef/weintraub-bio.html>, 1986.



## Appendix A A.L.I.C.E and AIML

A.L.I.C.E (Artificial Linguistic Internet Computer Entity) is an entertaining chatterbot created by Dr. Wallace in 1995 [WAL95] and continuously improved over the years. A.L.I.C.E asks and answers questions, acts as a secretary reminding people of appointments, spreads gossips and even tells lies. A.L.I.C.E won the 2000 and 2001 Loebner Prize. The basis for A.L.I.C.E's behavior is AIML (Artificial Intelligence Markup Language) - an XML (Extensible Markup Language) specification for programming chatterbots. It follows a minimalist philosophy based on simple stimulus-response algorithms, allowing programmers to specify how A.L.I.C.E will respond to various input statements.

The first edition of A.L.I.C.E was implemented 1995 using SETL, a language based on set theory and mathematical logic. It migrates to the platform-independent Java language in 1998. The first implementation of A.L.I.C.E and AIML in Java was codenamed *Program A*. Afterward Program B launched in 1999 and won the Loebner Prize in January 2000. Program C released by Jacco Bikker as the first C/C++ implementation of AIML in 2000. Program B Java edition was based on pre-Java 2 technologies however it did not take advantage of newer Java features such as Swing and Collections. Jon Baer recoded Program B and added many features leap in the interface and core, also named it as Program D as the latest A.L.I.C.E program (namely called Alicebot) Java edition. Beginning in November 2000, Program D is the only Java edition of the Alicebot engine actively supported.

Alicebot used “nearest neighbor classification” algorithm extended from Case-Based Reasoning. The cases are the categories in AIML and the algorithm finds best-matching pattern for each input. The category ties the response template directly to the stimulus pattern. A.L.I.C.E is conceptually not much more complicated than Original Weizenbaum's Eliza. The main differences are the much larger case base and the tools for creating new content by dialog analysis. Next generation of A.L.I.C.E is a part of the tradition of minimalist, reactive or stimulus-response robotics that demonstrate the most animated and realistic behavior

### A.1 Alicebot Architecture

Alicebot is not controlled by a single program but a collection of autonomous client-server communicating via TCP/IP. The client in this case is itself a server, specifically a limited HTTP server. In general there is no requirement that this program runs on the same machine. The figure A-1 shows a subset of the processes involved in a typical transaction with Alicebot through Alicebot engine. Alicebot engine implements AIML. Alicebot engine is responsible to receive the string user input and generates the bot's reply by activating and interpreting the bot's brain. The client (the user) utilizes a browser to connect to the server installed on the host Alicebot Server and transmit the query. The reply contains HTML and XML markup and the browser interprets them.

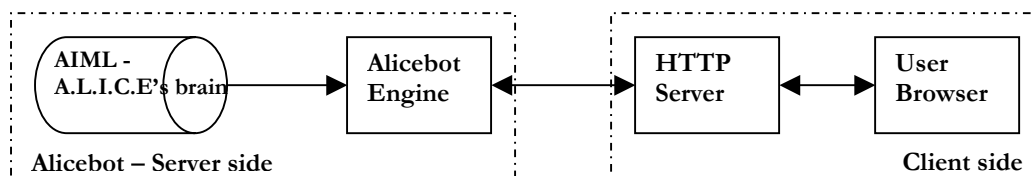


Figure A- 1 The process involved in a typical transaction with A.L.I.C.E



Alicebot engine uses “virtual IP” to keep track of clients since many clients have “dynamic IP addressing” enforced by their Internet Service Provider. Each this IP address is a key into a *hashtable* (or database) that stores the client’s dialogue, name and values of pronouns and other AIML values. This process also is controlled by Alicebot engine. When a client connects to Alicebot, the engine assigns a unique name to the value of “virtual IP”. This value then becomes a hidden variable in the client browser as the user id value in the browser’s cookie. Each successive client transaction contains this virtual IP address; the engine uses it as a key to index the conversation.

## A.2 AIML Structure

AIML emphasizes the language design with its minimalism. The simplicity of AIML makes it easy for non-programmers, especially those who already know HTML, to get started writing chat robots. AIML is an XML language, implying that it obeys certain grammatical meta-rules. The choice of XML syntax permits integration with other tools such as XML editors. Another motivation for XML is its familiar look and feel, especially to people with HTML experience. AIML consists of a list of statements called categories. The most important units of AIML are [WAL00], [BUS01]:

- **<aiml>** : the tag that begins and ends an AIML document.
- **<category>**: the tag that marks a "unit of knowledge" in an system’s knowledge base.
- **<pattern>** : the tag that is used to contain a simple pattern that matches what a user may type.
- **<topic>** : the tag that is used to contain current conversation topic.
- **<that>** : the tag that refers to system’s previous reply.
- **<template>**: the tag that contains the response to a user input. There is considerable freedom of expression in the construction of response templates.

### Category <category></category>

Each category contains an input pattern and a reply template. The AIML category tags are case-sensitive. Each open tag has an associated closing tag. This syntax obviously derives from XML. The syntax of an AIML category is:

<pre>&lt;category&gt; &lt;pattern&gt; PATTERN &lt;/pattern&gt; &lt;template&gt; Template &lt;/template&gt; &lt;/category&gt;</pre>	or	<pre>&lt;category&gt; &lt;pattern&gt; PATTERN &lt;/pattern&gt; &lt;that&gt; THAT &lt;/that&gt; &lt;template&gt; Template &lt;/template&gt; &lt;/category&gt;</pre>
------------------------------------------------------------------------------------------------------------------------------------	----	--------------------------------------------------------------------------------------------------------------------------------------------------------------------

### Pattern <pattern></pattern>

The pattern is the "stimulus" or "input" part of the category. The pattern is an expression in a formal language that consists of:

1. Words of natural language in UPPER CASE.
2. The symbol “\*” which matches any sequence of one or more words.
3. The symbol “\_” which is the same as “\*” except that it comes after Z in lexicographic order.
4. The markup <name/> that is replaced at robot load time with the name of the bot.

AIML broadly breaks down two parts: *pattern side AIML expression* – PSAE that can appear in the tag <pattern>, <that>, and <topic>, and *template side AIML expression* – TSAE that appear inside the tag <template>. Note there is a difference between the patterns “HELLO” and “HELLO \*”. “HELLO” matches only identical one-word sentences ("Hello.") and “HELLO \*” matches any sentence of two or more words starting with "Hello" ("Hello how are you?"). To simplify pattern description and matching, AIML patterns allow only one "\*" per pattern. In other words, "MY NAME IS \*" is a valid pattern, but "\* AND \*" is not.



## Topic <topic></topic>

Topic tag allows A.L.I.C.E to prefer responses that deal with the topic currently being discussed. This creates topical conversation, yet still has the ability to move from one subject to another. This tag also allows A.L.I.C.E to have duplicate patterns in different contexts (topics) allowing A.L.I.C.E to have different responses to the same input patterns depending on the topic. For example, "overriding" the "\*" pattern for different topics. There is <get\_topic/> tag to refer to the topic in your output statements (templates). By adding topics on top of all existing AIML keeps the bot's current personality.

Topic tags are placed around one or more categories. The categories (with each respective "pattern", "that", and "template") within a set of <topic> </topic> tags would be associated with the defined topic. The name of the topic would be given by a "name" property in the beginning topic tag. Here would be the full AIML format with topic:

```
<aiml>
<topic name="TOPIC">
  <category>
    <pattern> PATTERN </pattern>
    <that> THAT </that>
    <template> TEMPLATE </template>
  </category>
</topic>
</aiml>
```

The concept is that the botmaster uses the <set\_topic> tags to set the current topic being discussed. Once the topic is set, when the client types in a statement for A.L.I.C.E to find a response for, the categories defined within the <topic> tags matching the current topic will be searched first-- before any of the non-topic categories, or the default categories. If there is not a matching category defined in the current topic, then any categories that are not defined in topic tags are searched. As mentioned before, we can create categories with identical <pattern> phrases in different topics, each with different responses that cater to the current topic. A proof of concept example: A very useful topic entry might be the default "\*" input for specific topics. If A.L.I.C.E were set up on a pet store web site and a person was talking to A.L.I.C.E about dogs, a useful entry might as displayed in figure A-2.

```
<topic name="DOGS">
<category>
<pattern> * </pattern>
<template><random>
  <li> Dogs are one of the most popular pets to have.</li>
  <li> Have you ever met a Chihuahua you didn't like?</li>
  <li> What else do you know about dogs? </li>
  <li> Do you have any questions about dogs? </li>
</random> </template>
</category>
//more dog categories....
</topic>
```

**Figure A- 2 Example of an AIML category in a topic about dogs**

Normally there would be many entries in a topic, but in this example, we simply entered the default "\*". In this case, if the person said something that A.L.I.C.E didn't have a specific programmed response for, she could still respond intelligently within the current topic. (Note: this is all assuming there are existing categories that might set the current topic to "DOGS"). Also, though topics can only have one name, they can contain the wild characters "\*" or "\_" just like a pattern. Also, while sticking with the pattern criteria, only one wildcard character is allowed per name. This would allow topics like "CARING FOR DOGS" or "GROOMING DOGS" to also fall into the "\_ DOGS" topic. As with patterns, the more specific topics would gain preference over the wildcarded topic. This means that if the topic is currently "GROOMING DOGS" and yet there is not a programmed



response for the input in that category, then "\_ DOGS" would be checked, and then next the default categories.

### That <that></that>

The keyword "that" in A.L.I.C.E refers to whatever the robot said before a user input. Conceptually the choice of "that" comes from the observation of the role of the word "that" in dialogue fragments like:

```
Robot: Today is yesterday.  
Client: That makes no sense.  
Robot: The answer is 3.14159  
Client: That is cool.
```

The AIML tag <that> refers to the robot's previous reply. In AIML the syntax <that>...</that> permits an optional "That Pattern" to match the robot's "that" expression. A common example using "that" is any yes-no question:

```
<category>  
<pattern>YES</pattern>  
<that> DO YOU LIKE MOVIES </that>  
<template> What's your favorite movie? </template>  
</category>
```

This category handles the user input "YES" and checks to see whether the client is replying to the question "What's your favorite movie?". One interesting application of "that" are the categories that enable a robot to respond to "knock-knock" jokes (see figure A-3).

Categories	Fragment
<pre>&lt;category&gt; &lt;pattern&gt;KNOCK KNOCK&lt;/pattern&gt; &lt;template&gt;Who's there?&lt;/template&gt; &lt;/category&gt; &lt;category&gt; &lt;pattern&gt;*&lt;/pattern&gt; &lt;that&gt;WHO IS THERE&lt;/that&gt; &lt;template&gt;&lt;person/&gt; Who?&lt;/template&gt; &lt;/category&gt; &lt;category&gt; &lt;pattern&gt;*&lt;/pattern&gt; &lt;that&gt;* WHO&lt;/that&gt; &lt;template&gt;Ha ha very funny, &lt;getname/&gt;.&lt;/template&gt; &lt;/category&gt;</pre>	<pre>Client: KNOCK KNOCK. Robot: Who's there? Client: BANANA. Robot: banana Who? Client: KNOCK KNOCK. Robot: Who's there? Client: BANANA. Robot: banana Who? Client: KNOCK KNOCK. Robot: Who's there? Client: ORANGE. Robot: orange Who? Client: ORANGE YOU GLAD I DID NOT SAY BANANA. Robot: Ha ha very funny, Ska.</pre>

Figure A- 3 Example of the use of <that> tag in AIML categories

There are two forms of the <that> tag: a paired form <that>...</that> appearing in a category, and an atomic form <that/> always appearing in a template. Often we can use <that/> to find an opportunity to create a category with <that></that>. One of the default replies to the input "WHY" is "<that/>"? Why? This default produces the following dialogue fragment:

```
Robot: Do not ask me any more questions please.  
Client: WHY  
Robot: "Do not ask me any more questions please"? Why?
```

The botmaster notices the fragment and creates the new AIML category:

```
<category>  
<pattern>WHY</pattern>  
<that>DO NOT ASK ME ANY MORE QUESTIONS PLEASE</that>  
<template>Because I would rather talk about you.</template>  
</category>
```

Now the next client who asks "WHY" to the robot's request will active the new <that> category:



Robot: Do not ask me any more questions please.  
Client: WHY  
Robot: Because I would rather talk about you.

This style of conversational analysis does not presuppose that we know when the client will say "WHY"; rather it looks backward to identify cases where the "WHY" appeared following one of the robot's statements. Having identified the conversation point, the designer of A.L.I.C.E creates the new category.

## Template <template></template>

A template is the "response" or "output" part of an AIML category. The template is the formula for constructing the reply. The simplest template consists of plain, unmarked text. AIML provides markup functions to tailor the replies for each individual input and client. The markup function <getname/> for example inserts the client's name into the reply. The template may call the pattern matcher recursively using the <sr/> and <srai> tags. Many templates are simple symbolic reductions that map one sentence form to another, for example "Do you know what X is?" transforms to "What is X" with the category:

```
<category>
<pattern>DO YOU KNOW WHAT * IS</pattern>
<template><srai>WHAT IS <star/> </srai></template>
</category>
```

It is possible to find other tags inside the template (see table A-1 is the list of tags that may appear in AIML file based on AIML version 1.0 [RIN01]). The template may also contain other embedded HTML and XML. These embedded tags may cause the browser to play a sound, show an image, or run an applet. There is considerable freedom of expression in the construction of response templates. The programmer (namely called botmaster) is encouraged to study the examples in A.L.I.C.E, to and experiment with new ideas. There is no automated technique to finding the conversation points where the pronoun tags improve the flow of conversation. This is the "art" of AIML programming. There are no doubt countless programmer tricks and the field is wide open to linguists.

Table A- 1 List of AIML Tag on AIML version 1.0

No.	AIML Tag	Note
1.	<aiml>	AIML block delimiter
2.	<bot name="XXX"/>	Bot parameter may appear in pattern, XXX is the bot's property variable name
3.	<that index="nx,ny">	History reference bot's reply. The index="nx,ny" is optional, and if left off, the index value of "1,1" is assumed. The tag <that/> is equivalent to <that index="1,1"/>.
4.	<that>	Reference to previous' reply
5.	<category>	AIML memory unit
6.	<input index="n"/>	Reference to the entire user input response. The value "n" is a backward reference to the previous user response.
7.	<input/>	Reference to current user input response
8.	<condition name="X" value="Y">	Conditional branch
9.	<condition>	Conditional branch
10.	<gender>	Gender substitution exchange "he" and "she"
11.	<date/>	Get date and time
12.	<id/>	Get default localhost id
13.	<get name="XXX"/>	Get user name from user's preference



No.	AIML Tag	Note
		database
14.	<size/>	Number of loaded categories
15.	<star index="n">	Binding of "*", <star/> if n=1
16.	<thatstar index="n">	Binding of "*" in that
17.	<get name="topic">	Get current topic, default is "YOU"
18.	<topicstar index="n">	Binding of "*" in topic
19.	<version/>	AIML program version
20.	<gossip src = "XXX">	Append to file gossip with file name XXX
21.	<learn>X</learn>	AIML loading, X is AIML file name
22.	<li name="X" value="Y">	Conditional branch item used by <condition>
23.	<li value="Y">	Conditional branch item used by <condition name="X">
24.	<li>	General list item, used by <random> or <condition>
25.	<pattern>	Contains AIML pattern
26.	<person2/>	Transform person pronouns from 1 <sup>st</sup> to 2 <sup>nd</sup> person
27.	<person/>	Transform person pronouns from 1 <sup>st</sup> to 3 <sup>rd</sup> person
28.	<random>	Random uniform selection
29.	<set name="name">	Set user name
30.	<set name="topic">	Set current topic
31.	<set name="XXX">	Set variable XXX with a value
32.	<srai>	Recursion to the pattern matcher
33.	<sr/>	Abbreviates of <srai><star/></srai>
34.	<system>	Execute OS shell platform-dependent
35.	<template>	AIML template
36.	<think>	Nullify output
37.	<topic name="X">	AIML topic group, X is AIML pattern
38.	<uppercase>	Text manipulation to convert all text to Uppercase
39.	<lowercase>	Text manipulation to convert all text to Lowercase
40.	<sentence>	Text manipulation to capitalize the first word
41.	<formal>	Text manipulation to capitalize every word
42.	<if name="X" value="Y">	Conditional branch
43.	<else>	Conditional branch
44.	<Javascript>	AIML script for Java script
45.	<load filename="XXX">	Load a file with the file name is XXX

### Three Types of Category

Given only the <pattern> and <template> tags, there are three general types of categories: (a) atomic, (b) default, and (c) recursive. Strictly speaking, the three types overlap, because "atomic" and "default" refer to the <pattern> and "recursive" refers to a property of the <template>.

- a) "*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.

```
<category>
<pattern>WHAT IS A CIRCLE</pattern>
<template><set_it>A cicle</set_it> is a the set of points equidistant
from a common point called the center.
</template>
</category>
```

- b) 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. In the A.L.I.C.E distribution the ultimate default category resides in a file called "Pickup.aiml".





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>I NEED HELP *</pattern>
<template>Can you ask for help in the form of a question?</template>
</category>
```

responds to a variety of inputs from "I need help debugging my program" to "I need help with my marriage." Putting aside the philosophical question of whether the robot 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.

- c) *"Recursive"* categories are those that "map" inputs to other inputs, either to simplify the language or to identify synonymous patterns. Many synonymous inputs have the same response. This is accomplished with the recursive `<srai>` tag. Take for example the input "GOODBYE". This input has dozens of synonyms: "BYE", "BYE BYE", "CYA", "GOOD BYE", and so on. To map these inputs to the same output for GOODBYE we use categories like:

```
<category>
<pattern>BYE BYE</pattern>
<template><srai>GOODBYE</srai></template>
</category>
```

Simplification or reduction of complex input patterns is another common application for recursive categories. In English the question "What is X" could be asked many different ways: "Do you know what X is?", "Tell me about X", "Describe X", "What can you tell me about X?", and "X is what?" are just a few examples. Usually we try to store knowledge in the most concise, or common form. The `<srai>` function maps all these forms to the base form:

```
<category>
<pattern>DO YOU KNOW WHAT * IS</pattern>
<template><srai>WHAT IS <star/></srai></template>
</category>
```

The `<star/>` tag substitutes the value matched by "\*", before the recursive call to `<srai>`. This category transforms "Do you know what a circle is?" to "WHAT IS A CIRCLE", and then finds the best match for the transformed input. Another fairly common application of recursive categories is what might be called "parsing", except that AIML doesn't really parse natural language. A better term might be "partitioning" because these AIML categories break down an input into two (or more) parts, and then combine their responses back together. If a sentence begins with "Hello..." it doesn't matter what comes after the first word, in the sense that the robot can respond to "Hello" and whatever is after "..." independently. "Hello my name is Carl" and "Hello how are you" are quite different, but they show how the input can be broken into two parts. The category:

```
<category>
<pattern>HELLO *</pattern>
<template><srai>HELLO</srai> <sr/>
</template>
</category>
```

accomplishes the input partitioning by responding to "HELLO" with `<srai>HELLO</srai>` and to whatever matches "\*" with `<sr/>`. The response is the result of the two partial responses appended together.



### A.3 Knowledge Extraction in A.L.I.C.E

When the server is run, Alicebot engine load all AIML files into a data structure called *Graphmaster* (see figure A-4 (a)). The Graphmaster consists of a collection of nodes called *Nodemappers*. These Nodemappers map the branches from each node. The branches are either single words or wildcards. The root of the Graphmaster is a Nodemapper with about 2000 branches, one for each of the first words of all the patterns (40,000 in the case of the A.L.I.C.E). The number of leaf nodes in the graph is equal to the number of categories, and each leaf node contains the <template> tag.

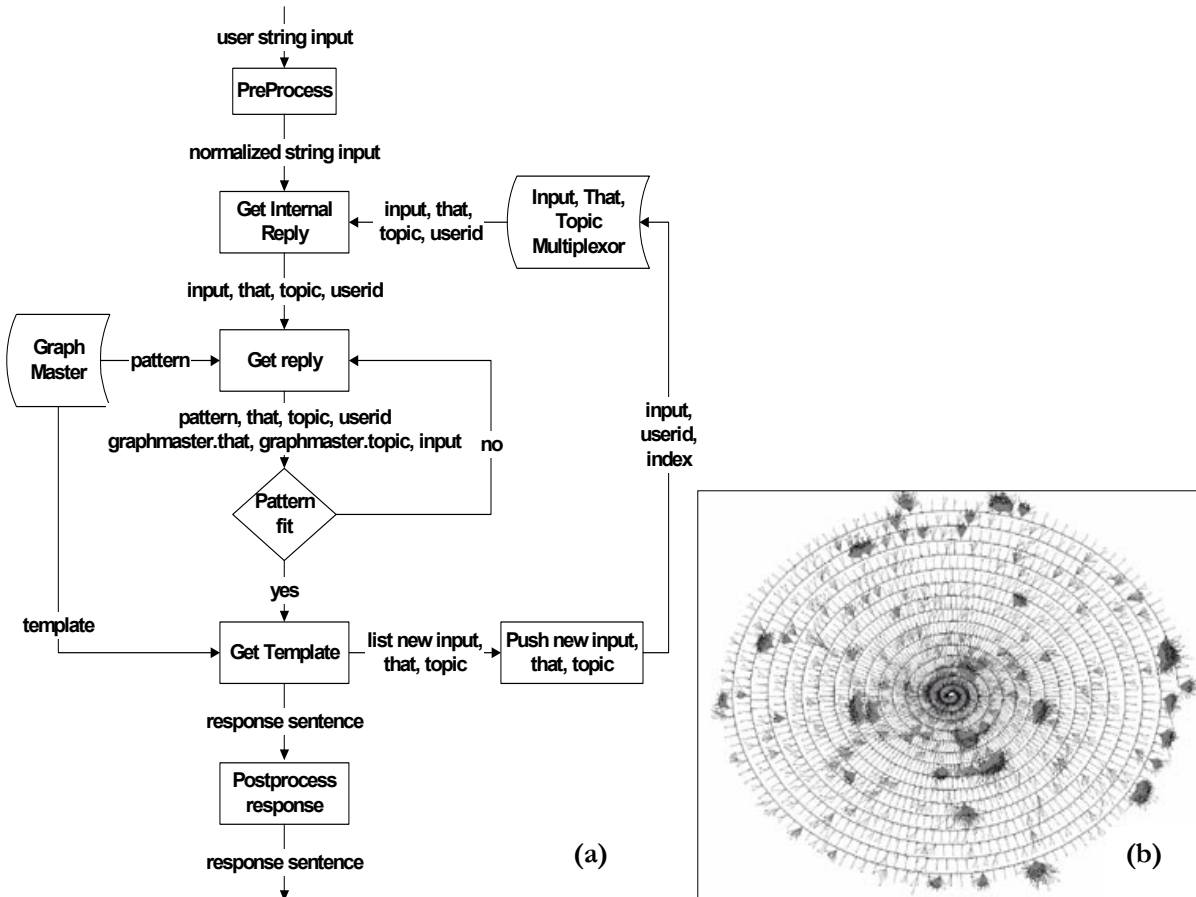


Figure A- 4 (a) Response construction process and (b) Alicebot's Graphmaster

Figure A-4 (b) shows how Alicebot engine constructs response to the user input. The most important processes that happen on the construction are described below.

#### Preprocess user's string input

Alicebot engine has a class called *Substituter* that performs a number of grammatical and syntactical substitutions on strings. One task involves preprocessing sentences to remove ambiguous punctuation to prepare the input for segmentation into individual sentence phrases. Another task expands all contractions and converts all letters to upper case; this process is called "normalization". The Substituter class also performs some spelling correction. One justification for removing all punctuation from inputs is the need to make A.L.I.C.E compatible with speech input systems, which of course do not detect punctuation.



## Get Internal Reply – Using History Preference

AIML has several tags to access conversation history such as:

- **<topic> and <get\_topic/>**, these tags create topical conversation, yet still has the ability to move from one subject to another.
- **<input> and <input index="n"/>**, these tags give the entire user input response. The value "n" is a backward reference to the previous user response.
- **<that/> and <that index="nx,ny"/>**, these tags give access to a previous response that the bot made.
- **<set\_xxx> and <get\_xxx>**. **<set\_xxx> X </set\_xxx>** will create the predicate of "xxx" and set its value to X. The conversation breaks down somewhat at this point, but the robot nonetheless conveys an illusion of understanding the user's response. For example for the fragment below:

```
User: NO I HAVE NO MONEY
A.L.I.C.E: I see. Would you like to have it?
User: YES
A.L.I.C.E: Where do you usually get money?
```

Using **<set\_it>money</set\_it>** with "xxx" = "it", the next reply uses category:

```
<category>
<pattern>YES</pattern><that>WOULD YOU LIKE TO HAVE IT</that>
<template>Where do you usually get <get_it/>?</template>
</category>
```

## Pattern Matching Operation - Using Alicebot's knowledge

There are really only three steps to matching an input to a pattern. If Alicebot engine receives (a) an input starting with word "X", and (b) a Nodemap of the graph, the pattern matching operation steps are [WAL01]:

1. Check if the Nodemap contain the key "\_". If so, search the subgraph rooted at the child node linked by "\_". Try all remaining suffixes of the input following "X" to see if one matches. If no match was found, try:
2. Check if the Nodemap contain the key "X". If so, search the subgraph rooted at the child node linked by "X", using the tail of the input (the suffix of the input with "X" removed). If no match was found, try:
3. Check if the Nodemap contain the key "\*". If so, search the subgraph rooted at the child node linked by "\*". Try all remaining suffixes 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.
4. For completeness there should also be a terminal case: If the input is null (no more words) and the Nodemap contains the <template> key, then a match was found. Halt the search and return the matching node.
5. If the root Nodemap contains a key "\*" and it points to a leaf node, then the algorithm is guaranteed to find a match.

At every node, the "\_" has first priority, an atomic word match second priority, and a "\*" match lowest priority. The patterns need not be ordered alphabetically, only partially ordered so that "\_" comes before any word and "\*" after any word. The matching is word-by-word, not category-by-category. 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. The matching algorithm is a highly restricted version of depth-first search, also known as backtracking. We can simplify the algorithm by removing the "\_" wildcard, and considering just the second two steps and also try understanding the simple case of PATTERNS without <that> and <topic>.



The algorithm searches the best match pattern by ensuring that the most specific pattern matches first. Basically it finds the longest pattern matching an input. The atomic category will always take precedence over any other type of category. If there are two identical patterns of atomic categories but one has a THAT, then the THAT category will take precedence over the other atomic categories, if the THAT matches Alicebot's previous response. If neither of above is true, then a reduction category that matches part of the pattern will give it's response, and finally if none of them matches, then the categories from the pickup file will take over. Any categories that are contained within a TOPIC section will be searched first if the current setting of TOPIC matches a TOPIC section. This result is an extension of the search order to the algorithm in the figure A-5.

```
ATOMIC with a TOPIC and a THAT
ATOMIC with a TOPIC
DEFAULT with a TOPIC and a THAT
DEFAULT with a TOPIC
ATOMIC with a THAT
ATOMIC
DEFAULT with a THAT
DEFAULT
```

Figure A- 5 Searching algorithm of Alicebot's pattern matching operation

This search order permits having identical category pattern within a TOPIC section and in the GENERAL section.

### Push new input, that and topic - Save current conversation

To track conversation, all users' conversations are saved on XML form on the <input> stack, <that> stack and <topic> stack, for example for fragment below:

```
User: MY NAME IS BILL.
A.L.I.C.E: Bill, are you married?
User: YES.
A.L.I.C.E: What kind of a car do you drive?
User: I DRIVE A MAZDA.
```

In this example, at the end of this exchange, here are the values of <input> stack:

```
<input index="1"/> = I DRIVE A MAZDA
<input index="2"/> = YES
<input index="3"/> = MY NAME IS BILL
```

Here are the values of <that> stack:

```
<that index="1,1"/> = WHAT KIND OF A CAR DO YOU DRIVE
<that index="2,1"/> = <set_name> BILL <set_name/> ARE YOU MARRIED?
```

And the value of <topic> stack is:

```
<topic index="1"/> = *
```

The <input index="n"/> tag gives the entire user input response. The <that index="nx,ny"/> tag gives the entire Alicebot response and <set\_name> tag buffers the name of the user.

### Post process the response

Alicebot engine's responses are processed into XML form therefore the client browser can show it to the user right away. The post process of Alicebot engine's response is conducted in order to combine all reply sentences (if the user's string input are broken down) and entail the response with appropriate value such as the bot predicate, host name, and etc.



## Appendix B Program D A.L.I.C.E

After we explain about global idea of A.L.I.C.E and AIML, in this chapter we will explain more specific about A.L.I.C.E; the program D A.L.I.C.E, which is called Alicebot as shorter. This chapter explains about the design of the program in section B.1. In section B.2 we explain how this program is compiled and run. Finally, list of important files and folders is described in section B.3.

### B.1 Design

This section provides a short explanation about the design of program D A.L.I.C.E using UML, since the developers did not provide this document. For detail information and data about this program please refer to [WAL95]. This section is organized as follows. First, the new context diagram of Alicebot is explained in section B.1.1. The next is the use case diagram of the program, which is described in section B.1.2. This diagram explains all new tasks and new actors that are involved in the program. The next is, the relationship between entities in the system is described by object diagram in section B.1.3. Finally, the event and transitions, which happen between the entities in the program, is described by a dynamic model and summarized in section B.1.4.

#### B.1.1 Context Model

There are two entities involved with the system, such as (see figure 6-5 below):

1. The user that sends string inputs and receives reply sentences.
2. AIML categories that contain list of patterns of Alicebot.
3. Multiplexor that deals with storing user-system conversation data during conversation.

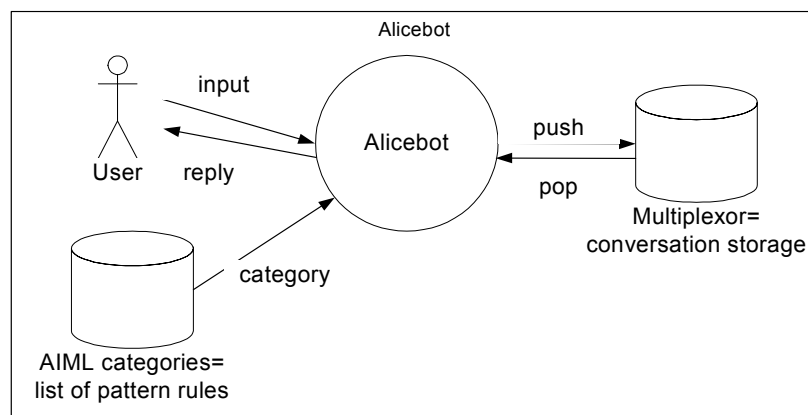


Figure B- 1 Alicebot context diagram

#### B.1.2 Use case

List of use cases are:

1.	Name	:	Sending input
	Description	:	To send input to the system
	Actors	:	The user
	Preconditions	:	-
	Exceptions	:	-
	Results	:	Activate use case normalizing input and syntactic and semantic analysis



2.	Name	:	Normalizing input
	Description	:	Recognizes input and corrects typo
	Actors	:	-
	Preconditions	:	The user has sent the input
	Exceptions	:	-
	Results	:	Input is normalized

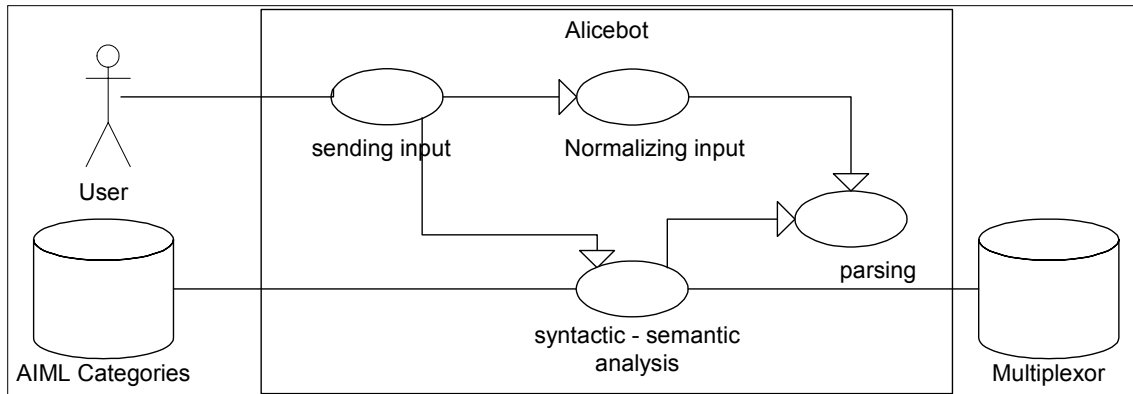


Figure B- 2 Alicebot's use case diagram

3.	Name	:	Syntactic and Semantic analysis
	Description	:	Generating reply sentence
	Actors	:	List of pattern rule and conversation history database
	Preconditions	:	The input has been normalized
	Exceptions	:	-
	Results	:	Reply sentence is sent to the user

4.	Name	:	Parsing
	Description	:	Parse a string word by word
	Actors	:	-
	Preconditions	:	Triggered by almost all task
	Exceptions	:	-
	Results	:	Parsed string

### B.1.3 Object Model

The object diagram in figure B-2 below consists of object classes of Alicebot. Unfortunately, in this figure we do not provide the detail relationship between classes of Alicebot. Alicebot consists of seventeen packages, such as:

1. **org.Alicebot.server.core**, this package contains eleven classes. The important classes are: Graphmaster that contains list of pattern rules, BotProperty that deals with loading and accessing system's preferences data, Global that deals with accessing the server, Multiplexor that deals with storing conversation information during conversation, and AbstractClassifier that accommodates the reply construction.
2. **org.Alicebot.server.core.loader**, this package contains two classes that deal with loading AIML categories to the graphmaster and a watcher that monitor if there is new additional AIML category.
3. **org.Alicebot.server.core.logging**, this package contains four classes that handles logging of Alicebot events to log files.



4. *org.Alicebot.server.core.node*, this package consists of two classes: Nodemapper and Nodemaster. Both classes map the branches in a Graphmaster structure.
5. *org.Alicebot.server.core.parser*, this package contains twelve classes which handle parsing operation both in AIML and XML form.
6. *org.Alicebot.server.core.processor*, this package contains thirty-six classes which are used to recognize tags in AIML form and used by parser package.

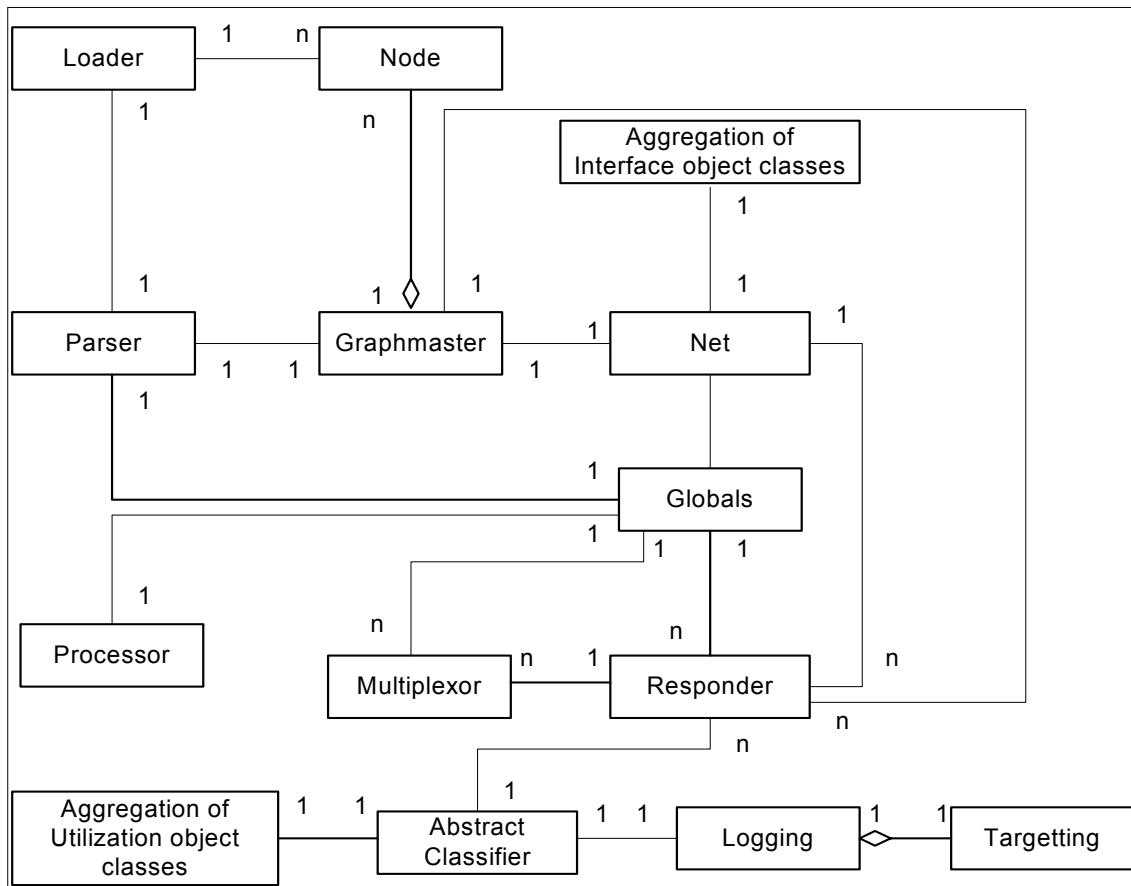


Figure B- 3 Alicebot's class diagram

7. *org.Alicebot.server.core.processor.loadtime*, this package contains eighteen classes. They are used to recognize tags in AIML form from files that must be loaded when the server runs on the first time.
8. *org.Alicebot.server.core.responder*, this package contains ten classes. They are used to process and log input-output via a given channel (TEXT, HTML, FLASH and AIM).
9. *org.Alicebot.server.core.targetting*, this package contains seven classes that used by Parser package to read or write a target file.
10. *org.Alicebot.server.core.targetting.gui*, this package contains two classes for demo GUI purposes.
11. *org.Alicebot.server.core.util*, this package contains thirteen classes. They are used to perform string operation and other utilization that are needed by other packages.
12. *org.Alicebot.server.net*, this package contains three classes that used to configure the server.
13. *org.Alicebot.server.net.listener*, this package contains five classes. They are used to be interface communication of a given channel: AIM, IRC, ICQ, and JAB.
14. *org.Alicebot.server.net.server*, this package only contains one classes: Alice. This class is used to perform Java Servlet.





- The class diagram displayed in figure B-3 does not contain all packages and classes above. We only display relationship between object classes that are involved in input-output process.

Scenario of Alicebot is divided into two parts, such as:

- By the scenario above, we construct the event trace diagram in figure B-4.



To install and run the Alicebot server, we should to download and install a Java 2 version 1.4 compatible JVM. Alicebot program lets us to configure an unlimited number of bots that can run at the same time. Alicebot's botmaster has provided a file to compile the system. We simply use file `./build/build.bat` if we use Microsoft Windows environment and `./build/build.sh` if we use Linux/Unix environment.



Alicebot's botmaster also has provide two executable files to run the server. The `run.bat` file is provided for Windows users so that common problems with the DOS environment can be handled before launching the `server.bat` file (which is started by `run.bat`). The `server.sh` file is provided for Linux users. It is executable by issuing the command: `chmod +x server.sh`. Starting the server script should look like in figure B-5. The bot's name is "Testy" with its id is "TestBot-1". If there are any errors while starting up, an explanation will be printed and the server will shut down. There are several tasks on loading time, they are:

- (a) Initialization of Multiplexor, Graphmaster and loading server configuration.
- (b) Loading list of words/phrases from startup file.
- (c) Loading list of pattern rules into Graphmaster.
- (d) Initiation listening port. In this example Alicebot listening in <http://vasya:2001>.

Once the Alicebot server is started, users can interact with it through a web browser with URL address: `http://localhost:<no-port>`.

```
[noel@vasya ProgramD]$ ./server.sh
Starting Alicebot Program D.
[12:10:21] Starting Alicebot Program D version 4.1.5
[12:10:21] Using Java VM 1.4.0-b92 from Sun Microsystems Inc.
[12:10:22] On Linux version 2.4.2-2 (i386)
[12:10:22] Predicates with no values defined will return: "undefined".
[12:10:22] Initializing Multiplexor.
[12:10:22] Loading Graphmaster.
[12:10:22] Starting up with "../ProgramD/conf/startup.xml".
[12:10:22] Configuring bot "TestBot-1".
[12:10:22] Loaded 287 input substitutions.
[12:10:22] Loaded 19 gender substitutions.
[12:10:23] Loaded 9 person substitutions.
[12:10:23] Loaded 60 person2 substitutions.
[12:10:23] Loaded 4 sentence-splitters.
[12:10:24] 6000 categories loaded so far.
[12:10:25] 12000 categories loaded so far.
[12:10:27] 18000 categories loaded so far.
[12:10:28] 1 bots thinking with 23879 categories.
[12:10:28] Alicebot Program D (c) 1995-2002 A.L.I.C.E. AI Foundation
[12:10:28] All Rights Reserved.
[12:10:28] This program is free software; you can redistribute it and/or
[12:10:28] modify it under the terms of the GNU General Public License
[12:10:28] as published by the Free Software Foundation; either version 2
[12:10:28] of the License, or (at your option) any later version.
[12:10:28] Alicebot Program D version 4.1.5 Build [00]
[12:10:28] 23879 categories loaded in 6.133 seconds.
[12:10:28] The AIML Watcher is not active.
[12:10:28] HTTP server listening at http://vasya:2001
[12:10:29] Interactive shell: type "/exit" to shut down; "/help" for help.
[12:10:29] user> CONNECT : Hello there user and thanks for connecting : * : TestBot-1
[12:10:29] Match: CONNECT : * : * : TestBot-1
[12:10:29] Filename: "../ProgramD/conf/./aiml/standard/std-connect.aiml"
[12:10:29] Response 1 in 117 ms. (Average: 117.0 ms.)
[12:10:29] Testy 1> Hello there user and thanks for connecting!
[12:10:29] [Testy 1] user>
```

**Figure B- 5 Server script when the server runs**

Before the bot server can shut down, it must stop the http server and must save any predicates left in the cache. It also must stop all listeners. There are several different but all of them should cause the Alicebot engine to print information that looks like this:

```
[12:26:14] AliceServer is shutting down.
[12:26:14] Shutting down all BotProcesses.
[12:26:14] Shutting down org.Alicebot.server.net.JettyWrapper@861f24
```



```
[12:26:18] Finished shutting down BotProcesses.  
[12:26:18] Saving all cached predicates (3)  
[12:26:18] Shutdown complete.
```

"Bot Processes" are such things as the http server and listeners.

## B.3 Important Files and Folders

Here are the most important files or folder of A.L.I.C.E [BUS02]:

1. Folder **bots/**. This is where all AIML files are stored. It also contains two other files, such as:
  - a. **startup.xml**. The bots are configured in the **startup.xml** file with the root element tag is called **<programd-startup>**, and that it contains exactly one child element called **<bots>**. Inside **<bots>**, we place one or more **<bot>** elements. These **<bot>** elements are not the same as the AIML tag of the same name. Each **<bot>** element has two important attributes: **id** and **enabled**. The first one assigns an identifier, which should be unique, for the bot. The identifier will be used internally by the engine and will be written to some log resources. The **enabled** attribute should have either of the values **"true"** or **"false"**. If the value is **"true"**, the engine will try to load that bot when the server starts up. Within the **<bot>** element we define:
    - **Bot properties**, bot properties are predicates that cannot be changed during the runtime life of the bot, but which *can* be included in AIML patterns for matching. A common property to define for a bot is **"name"**. Bot properties are defined in individual **<property>** elements inside a bot's **<properties>** element, as in the example: **<property name="name" value="A.L.I.C.E"/>**. This associates the bot property name "name" with the value **"A.L.I.C.E"**. Properties don't mean anything unless your AIML uses them. We can display the value of a bot property inside an AIML template by using the form **<bot name="property-name"/>**.
    - **Substitutions**, substitutions have several different purposes, depending on their type. Input substitutions contribute to the process of [input normalization](#). Each individual substitution specification, regardless of whether it is inside an **<input>**, **<gender>**, **<person>** or **<person2>**, takes the same form as this example from: **<substitute find=" becasue " replace=" because "/>**. This means that, when this substitution is applied, each instance of the separate word "becasue" will be replaced with "because". Note the spaces that pad the values of the **"find"** and **"replace"** attributes. These spaces are, approximately, indications that we want to match only on "word boundaries". Omitting one or both padding spaces in the **"find"** string would mean that its contents should be matched even if they occur as part of a word. This is generally not desirable. **Person** substitutions provide macros for transformations by the **<person>** tag; likewise **person2** and **gender** apply to the **<person2>** and **<gender>** tags, respectively.
    - **Sentence-splitters**, sentence-splitters, as described in the [AIML spec](#), 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. Since sentence-splitters are applied to the input *after* substitution normalizations, they can be more general rules. The entire collection of example sentence-splitters shipped is:

```
<sentence-splitters>  
  <splitter value="."/>  
  <splitter value="!"/>  
  <splitter value="?"/>  
  <splitter value=";"/>  
</sentence-splitters>
```



- b. **startup.aiml**. The bots use this file to load the AIML file. This file consists one category:

```
<aiml>
<category>
  <pattern>LOAD ROBOT</pattern>
  <template>
    <learn filename="/home/Alicebot/brain/<filename-1>"/>
    <learn filename="/home/Alicebot/brain/<filename-2>"/>
    ...
    <learn filename="/home/Alicebot/brain/<filename-n>"/>
  </template>
</category>
</aiml>
```

Using `<learn>` tag, AIML reader will load all categories in the filename into Graphmaster.

2. Folder **build/**. This folder contains executable files to compile Alicebot: build.bat and build.sh.
3. Folder **classes/**. This folder contains classes of Alicebot Java source code as the result compilation process.
4. Folder **conf/**. This folder contains one file: **Jetty.xml**. This file is the web interface of Program D. Program D is designed to be able to work as a "servlet"--that's an application that runs on a server but interacts with users via the web. Jetty is an open source HTTP server and servlet container. Future versions of Program D will hopefully be easier to integrate with other servlet containers. This is the file that is read by the Jetty server when it is invoked by Program D. The parameter that will probably be of most interest here is the port number. This is specified in the place that looks like:

```
<Set name="Port">2001</Set>
```

For testing purposes, the default value of the port is 2001.

5. Folder **database/**. This folder contains executable files to run database: database.bat for Microsoft Windows environment and database.sh for Linux/Unix environment.
6. Folder **ffm/**. This folder contains log files of predicate values (INPUT, THAT and TOPIC) that recorded during conversation per user id.
7. Folder **lib/**. This folder contains all Java libraries that are needed to compile and execute the program.
8. Folder **logs/**. This folder contains chat logs, error logs, database logs and more. This directory doesn't exist until the first run the server.
9. Folder **src/org/alicebot/server/**. This folder contains Alicebot's Java files source codes of its object classes.
10. Folder **template/**. This folder contains two subfolders: **flash/**. and **html/**. The folder **flash/**. contains file **chat.flash** that is used for constructing bot response via FLASH channel. The folder **html/**. contains file **chat.html** that is used for constructing a web page with the bot's response. It is a plain HTML file, with a few important tags:
  - `<reply></reply>`, These tags separate the 'header' of the bot response from the 'main' part and the 'footer'. What comes before is the header, what is inside is the main part



- and what comes after is the footer. The main part is repeated for every sentence in the client input. These must be included, even though the 'main' part may be empty.
- `<userinput/>`, The user input. This may be either the whole input (in the 'header') or one sentence of it (in the 'main' part).
  - `<response/>` The bot output. This may be either whole ('footer') or one sentence ('main').
  - `<bot name="xxx"/>` Includes the value of the bot predicate 'xxx' in the page.

It is possible for us to create different HTML templates for chat with A.L.I.C.E, by following the example provided in `templates/html/chat.html`. To request one of these alternate templates, it is done by adding a `template` parameter to the URL request. The value of this parameter should be the name of the file, minus the extension. For example, if a template file is named `fancy.html` and stored in the `templates/html` directory, then the request is: <http://<host>:<no-port>?botid=<bot-id>&template=fancy>. If we configured more than one bot, we need to request the one we want by appending a `botid` parameter to your request. For example: <http://vasya:2001?botid=TestBot-2>, would request the bot with id `TestBot-2` from the machine `vasya`. Once the user is connected with the bot, they will see on their browser figure B-6.

You said:	CONNECT
Testy 1 said:	Hello there user and thanks for connecting!
<input type="text"/>	
<input type="button" value="Say"/>	
You are speaking with Testy 1 from vasya.	
Testy 1's botmaster is A.L.I.C.E. AI Foundation.	
You can:	
<ul style="list-style-type: none"><li>• <a href="#">log in.</a></li><li>• <a href="#">register a new username and password.</a></li></ul>	

**Figure B- 6 Alicebot's main page**

11. File **server.property.xml**. This file contains the overall configuration of the server including xml startup file `startup.xml`. This formatted file is used to load AIML files into graphmaster. This file contain:
  - Database configuration
  - Shell and console configuration
  - Startup file configuration
  - Program response time out configuration, if the AIML file contains infinite loops
  - User accessing configuration
  - Javascript conviguration
  - Other server property and configuration.



The `server.properties` file is documented inline, therefore we do not have to exhaustively enumerate every property. The example of lines in this files are:

```
# Root directory
programd.home=./

# Bot configuration startup file (relative to programd.home)
programd.startup=bots/standard/startup.xml

# Merge categories with identical pattern:that:topic (true/false)
programd.merge=true

# The registration template.
programd.responder.html.template.register=register.html

# The login template.
programd.responder.html.template.login=login.html

# The change password template.
programd.responder.html.template.change-password=change-password.html

# Standard (text file) logs
programd.logging.access.path=./logs/access.log
programd.logging.chat.path=./logs/chat.log
programd.logging.database.path=./logs/database.log
...
```

12. File **run.bat** and file **server.sh**. These file are used to run the server in Microsoft Windows environment and in Linux/Unix environment.
13. Folder **targets/**. Data for the targeting feature. This directory doesn't exist until the server run.



## Appendix C Affective Knowledge Base

In this chapter all **preference rules of my\_Eliza prototype-1** will be given in tables. Please see section C.1 for preference rules for *concern of the other knowledge base* and section C.2 for preference rules for cognitive reasoning knowledge base. The tables may contain the following columns name:

1. **User-Affect** is a candidate of user's affective state, which contains a condition for rule as the result of *emotive lexicon dictionary parser* against the user's string input. There are seven possible values in this condition, such as happiness, sadness, surprise, fear, disgust, anger, and neutrality.
2. **System-Affect** is a candidate of system's reaction affective state, which contains a condition for rule as the result of *emotive lexicon dictionary parser* against the system's reply sentence. There are also seven possible values in this condition, such as happiness, sadness, surprise, fear, disgust, anger, and neutrality.
3. **Question** is a boolean value whether the user asks question or not. It is also resulted by *emotive lexicon dictionary parser* against the user's string input.
4. **Affect-value** is a value of situation type of the user's string input whether in positive (+), negative (-) or joking (#) situation type. It is resulted by *emotive label memory structure extractor*.
5. **Concern-value** is a value of situation type of the system's reply sentence whether in (+), negative (-) or joking (#) situation type. It is also resulted by *emotive label memory structure extractor*.
6. **Thermometer** is current system's affective state. It is the highest degree of six thermometers in *Intensity Analyzer*. There are seven possible value of this condition, such as happiness, sadness, surprise, fear, disgust, anger, and neutrality.

An empty field in the tables signifies a "don't care"; the corresponding condition may or may not be met. Those tables are divided by *user-affect* value.

### C.1 Preference Rules for Concern of the other Knowledge Base

1. Preference rule of user's affective state = Happiness

Table C- 1 Concern of user happiness preference rules

Rule #	User-affect	Question	Affect-value	Thermometer	First-reaction
1.	Happiness	True	+	Happiness	Joy
2.	Happiness	False	+	Happiness	Happy_for
3.	Happiness	True	-		Uncertainty
4.	Happiness	False	-		Disappointment
5.	Happiness	True	#	Happiness	Joy
6.	Happiness	False	#	Happiness	Happy_for
7.	Happiness		+	Sadness	Relief
8.	Happiness	True	-/#	Not Happiness Not neutrality	Uncertainty
9.	Happiness	False	-/#	Not Happiness Not neutrality	Disappointment
10.	Happiness	True	+	Disgust	Shame
11.	Happiness	False	+	Disgust	Normal
12.	Happiness	True	+	Fear	Shame
13.	Happiness	False	+	Fear	Normal
14.	Happiness	True	+	Surprise	Joy
15.	Happiness	False	+	Surprise	Happy_for
16.	Happiness	True	+	Anger	Shame
17.	Happiness	False	+	Anger	Normal
18.	Happiness	True	+	Neutrality	Joy
19.	Happiness	False	+	Neutrality	Happy_for





## 2. Preference rule of user's affective state = Sadness

Table C- 2 Concern of user sadness preference rules

Rule #	User-affect	Question	Affect-value	Thermometer	First-reaction
1.	Sadness		+/-	Happiness	Fear
2.	Sadness		#	Happiness	Uncertainty
3.	Sadness		+/-	Sadness	sorry_for
4.	Sadness		+/-	Disgust	Shame
5.	Sadness		+/-	Fear	Fear_confirmed
6.	Sadness		+/-	Surprise	Fear_confirmed
7.	Sadness		+/-	Anger	Shame
8.	Sadness		+/-	Neutrality	Fear

## 3. Preference Rule of user's affective state = Fear

Table C- 3 Concern of user fear preference rules

Rule #	User-affect	Question	Affect-value	Thermometer	First-reaction
1.	Fear		+/-	Happiness	Uncertainty
2.	Fear		#	Happiness	Gloating
3.	Fear		+/-	Not happiness Not fear Not surprise Not neutrality	Shame
4.	Fear		#	Not happiness Not fear Not surprise Not neutrality	Reproach
5.	Fear		+/-	Not happiness Not sadness Not disgust Not anger	Fear
6.	Fear		#	Not happiness Not sadness Not disgust Not anger	Uncertainty

## 4. Preference rule of user's affective state = Surprise

Table C- 4 Concern of user surprise preference rules

Rule #	User-affect	Question	Affect-value	Thermometer	First-reaction
1.	Surprise			Happiness	Uncertainty
2.	Surprise		+/-	Not happiness Not surprise Not neutrality	Shame
3.	Surprise		#	Not happiness Not surprise Not neutrality	Reproach
4.	Surprise			Surprise	Fear
5.	Surprise			Neutrality	Fear

## 5. Preference rule of user's affective state = Disgust

Table C- 5 Concern of user disgust preference rules

Rule #	User-affect	Question	Affect-value	Thermometer	First-reaction
1.	Disgust		+/-	Happiness	Uncertainty
2.	Disgust		#	Happiness	Gloating
3.	Disgust		+/-	Sadness	Distress
4.	Disgust		#	Sadness	Reproach
5.	Disgust			Disgust	Joy



Rule #	User-affect	Question	Affect-value	Thermometer	First-reaction
6.	Disgust		+/-	Fear	Distress
7.	Disgust		#	Fear	Reproach
8.	Disgust		+/-	Surprise	sorry for
9.	Disgust		#	Surprise	Gloating
10.	Disgust		+/-	Anger	Anger
11.	Disgust		#	Anger	Reproach
12.	Disgust		+/-	Neutrality	Uncertainty
13.	Disgust		#	Neutrality	Gloating

## 6. Preference rule of user's affective state = Anger

Table C- 6 Concern of user anger preference rules

Rule #	User-affect	Question	Affect-value	Thermometer	First-reaction
1.	Anger		+/#	Happiness	Uncertainty
2.	Anger		-	Happiness	Fear
3.	Anger			Sadness	Distress
4.	Anger		+	Disgust	Uncertainty
5.	Anger		-/#	Disgust	Hate
6.	Anger			Fear	Remorse
7.	Anger			Surprise	Fear-confirmed
8.	Anger			Anger	Anger
9.	Anger			Neutrality	Fear

## 7. Preference rule of user's affective state = Neutrality

Table C- 7 Concern of user neutrality preference rules

Rule #	User-affect	Question	Affect-value	Thermometer	First-reaction
1.	Neutrality		+/#	Happiness	Joy
2.	Neutrality		-	Happiness	Fear
3.	Neutrality		+	Sadness	Hope
4.	Neutrality		-	Sadness	sorry for
5.	Neutrality		#	Sadness	Uncertainty
6.	Neutrality		+	Disgust	Shame
7.	Neutrality		-/#	Disgust	Reproach
8.	Neutrality		+	Fear	Shame
9.	Neutrality		-	Fear	Fear-confirmed
10.	Neutrality		#	Fear	Disappointment
11.	Neutrality		+/#	Surprise	happy for
12.	Neutrality		-	Surprise	Fear-confirmed
13.	Neutrality		+	Anger	Normal
14.	Neutrality		-/#	Anger	Anger
15.	Neutrality		+	neutrality	Normal
16.	Neutrality		-	neutrality	Uncertainty
17.	Neutrality		#	neutrality	Joy

## C.2 Preference Rules of Cognitive Reasoning Knowledge Base

### 1. Preference rule of user's affective state = Happiness

Table C- 8 Cognitive reasoning preference rules for user happiness

Rule #	User-affect	Affect-Value	System-react	Concern-Value	Thermometer	Cognitive-reaction
1.	Happiness	+	Not fear Not surprise Not anger Not disgust	+	Happiness	Happy-for
2.	Happiness	+	Not fear Not surprise Not anger Not disgust	-	Not sadness Not surprise Not anger Not disgust	Resentment



Thesis Report: My\_Eliza, a Multimodal Communication System  
**Appendix C Affective Knowledge Base**

Rule #	User-affect	Affect-Value	System-react	Concern-Value	Thermometer	Cognitive-reaction
					Not fear	
3.	Happiness	+		#		Gratitude
4.	Happiness	-	Not fear Not surprise Not anger Not disgust Not neutrality	+	Not sadness Not surprise Not anger Not disgust Not fear	Gloating
5.	Happiness	-	Not fear Not surprise Not anger Not disgust Not neutrality	-	Not sadness Not surprise Not anger Not disgust Not fear	Remorse
6.	Happiness	-		#		Pride
7.	Happiness	#	Not fear Not surprise Not anger Not disgust Not neutrality	+	Not sadness Not surprise Not anger Not disgust Not fear	Joy
8.	Happiness	#	Not fear Not surprise Not anger Not disgust Not neutrality	-	Not sadness Not surprise Not anger Not disgust Not fear	Reproach
9.	Happiness	#		#		Gratification
10.	Happiness	+/-/#	Not happiness Not sadness Not anger Not disgust Not neutrality	+		Hope
11.	Happiness	+/-/#	Not happiness Not sadness Not anger Not disgust Not neutrality	-		Fear
12.	Happiness	+/-/#	Disgust	+		Disliking
13.	Happiness	+/-/#	Disgust	-		Hate
14.	Happiness	+/-/#	Anger	+		Uncertain
15.	Happiness	+/-/#	Anger	-		Anger
16.	Happiness	-	Neutrality	+	Happiness	Hope
17.	Happiness	-	Neutrality	-	Happiness	Resentment
18.	Happiness	#	Neutrality	+	Happiness	Joy
19.	Happiness	#	Neutrality	-	Happiness	Reproach
20.	Happiness	+/-/#	Happiness	+	Sadness	Hope
21.	Happiness	+/-/#	Happiness	-	Sadness	Fear
22.	Happiness	+/-/#	Not happiness Not surprise Not anger Not disgust Not fear	+	Sadness	Sorry-for
23.	Happiness	+/-/#	Not happiness Not surprise Not anger Not disgust Not fear	-	Sadness	Distress
24.	Happiness	+	Not sadness Not surprise Not anger Not disgust Not fear	+	Surprise	Satisfaction
25.	Happiness	+/-/#	Not surprise Not anger Not disgust Not fear	-	Surprise	Fear-confirmed
26.	Happiness	-	Happiness	+	Surprise	Sorry-for
27.	Happiness	+/-/#	Sadness	+	Surprise	Sorry-for
28.	Happiness	+/-/#	Not surprise Not anger Not disgust Not fear	+	Fear	Relief



Rule #	User-affect	Affect-Value	System-react	Concern-Value	Thermometer	Cognitive-reaction
29.	Happiness	+/-/#	Not surprise Not anger Not disgust Not fear	-	Fear	Fear-confirmed
30.	Happiness	+/-/#	Sadness	+	Fear	Sorry-for
31.	Happiness	+/-/#	Not surprise Not anger Not disgust Not fear	+	Not surprise Not sadness Not disgust Not fear Not happiness	Normal
32.	Happiness	+/-/#	Not surprise Not anger Not disgust Not fear	-	Disgust	Disliking
33.	Happiness	+/-/#	Not surprise Not anger Not disgust Not fear	-	Anger	Anger
34.	Happiness	+	Happiness	+	Neutrality	Joy
35.	Happiness	+	Sadness	+	Neutrality	Uncertain
36.	Happiness	+/#	Neutrality	+	Neutrality	Joy
37.	Happiness	+/#	Neutrality	-	Neutrality	Uncertain
38.	Happiness	-	Neutrality	+	Neutrality	Hope
39.	Happiness	-	Neutrality	-	Neutrality	Fear

## 2. Preference rule of user's affective state = Sadness

Table C- 9 Cognitive reasoning preference rules for user sadness

Rule #	User-affect	Affect-value	System-react	Concern-value	Thermometer	Cognitive-reaction
1.	Sadness	+/#	Not surprise Not fear Not disgust Not anger	+	Not surprise Not fear Not disgust	Normal
2.	Sadness	+	Not surprise Not fear Not disgust Not anger	-	Not surprise Not fear Not disgust Not anger Not sadness	Fear
3.	Sadness	+		#		Pride
4.	Sadness	-	Not surprise Not fear Not disgust Not anger	+	Not neutrality Not disgust Not anger	Sorry-for
5.	Sadness	-	Not surprise Not fear Not disgust Not anger	-	Not surprise Not fear Not disgust Not anger	Reproach
6.	Sadness	-		#		Gloating
7.	Sadness	#	Not surprise Not fear Not disgust Not anger	-	Not surprise Not fear Not disgust Not anger	Remorse
8.	Sadness	#		#		Gratification
9.	Sadness	+/-/#	Not happiness Not sadness Not disgust Not anger Not neutrality	+	Happiness	Hope
10.	Sadness	+/-/#	Not happiness Not sadness Not disgust Not anger Not neutrality	-	Happiness	Fear
11.	Sadness	+/-/#	Disgust	+/-	Not disgust Not anger	Disliking
12.	Sadness	+/-/#	Anger	+/-	Not disgust Not anger	Uncertain



**Thesis Report: My\_Eliza, a Multimodal Communication System**  
**Appendix C Affective Knowledge Base**

Rule #	User-affect	Affect-value	System-react	Concern-value	Thermometer	Cognitive-reaction
13.	Sadness	+	Happiness	+	Sadness	Hope
14.	Sadness	+	Not surprise Not fear Not disgust Not anger Not neutrality	-	Sadness	Distress
15.	Sadness	-	Happiness	+	Sadness	Sorry-for
16.	Sadness	+/-/#	Anger	+/-	Not surprise Not happiness Not disgust Not anger Not neutrality	Distress
17.	Sadness	+	Neutrality	-	Sadness	Distress
18.	Sadness	+/#	Not surprise Not fear Not disgust Not anger Not neutrality	+	Not happiness Not sadness Not disgust Not anger Not neutrality	Relief
19.	Sadness	+/#	Happiness	-	Not happiness Not sadness Not disgust Not anger Not neutrality	Disappointment
20.	Sadness	+/-/#	Not surprise Not fear Not disgust Not anger Not neutrality	-	Not happiness Not sadness Not disgust Not anger Not neutrality	Fear-confirmed
21.	Sadness	+/#	Neutrality	+	Not happiness Not sadness Not anger Not neutrality	Normal
22.	Sadness	+	Neutrality	-	Surprise	Disappointment
23.	Sadness	-/#	Neutrality	-	Not happiness Not sadness Not disgust Not anger Not neutrality	Distress
24.	Sadness	+	Neutrality	-	Fear	Fear-confirmed
25.	Sadness	+/#/-		-	Not happiness Not sadness Not surprise Not fear Not neutrality	Hate
26.	Sadness	-	Happiness	+	Disgust	Shame
27.	Sadness	-/#	Disgust	+	Disgust	Disliking
28.	Sadness	+/-/#	Anger	+	Disgust	Uncertain
29.	Sadness	+/-/#		-	Anger	Anger
30.	Sadness	-	Not surprise Not fear Not disgust	+	Anger	Uncertain

**3. Preference Rule of user's affective state = Fear**

**Table C- 10 Cognitive reasoning preference rules for user fear**

Rule #	User-affect	Affect-value	System-react	Concern-value	Thermometer	Cognitive-reaction
1.	Fear	+/#	Not fear Not surprise	+	Not fear Not surprise	Normal
2.	Fear	+		#		Gratitude
3.	Fear	-	Not fear Not surprise Not disgust Not anger	+	Not fear Not surprise Not disgust Not anger	Sorry-for
4.	Fear	+/-/#	Not fear Not surprise	-	Not fear Not disgust Not anger	Fear
5.	Fear	-		#		Gloating



Thesis Report: My\_Eliza, a Multimodal Communication System  
**Appendix C Affective Knowledge Base**

Rule #	User-affect	Affect-value	System-react	Concern-value	Thermometer	Cognitive-reaction
6.	Fear	#		#		Gratification
7.	Fear	+/-/#	Surprise	+		Hope
8.	Fear	+/-/#	Surprise	-		Fear
9.	Fear	-	Not fear Not surprise Not happiness Not sadness Not neutrality	+	Not fear Not surprise Not disgust Not anger	Normal
10.	Fear	+/-/#		+	Not happiness Not sadness Not disgust Not anger Not neutrality	Hope
11.	Fear	+/-/#		-	Not happiness Not sadness Not disgust Not anger Not neutrality	Fear
12.	Fear	+/#	Not fear Not surprise Not disgust Not sadness Not anger	-	Not fear Not surprise Not happiness Not sadness Not neutrality	Shame
13.	Fear	-	Not fear Not surprise	+	Not fear Not surprise Not happiness Not sadness Not neutrality	Uncertain
14.	Fear	-	Not fear Not surprise Not disgust Not sadness Not anger	-	Not fear Not surprise Not happiness Not sadness Not neutrality	Disliking
15.	Fear	+/#	Not fear Not surprise Not happiness Not neutrality Not anger	-	Not fear Not surprise Not happiness Not sadness Not neutrality	Distress
16.	Fear	-	Not fear Not surprise Not happiness Not neutrality Not anger	-	Not fear Not surprise Not happiness Not sadness Not neutrality	Hate
17.	Fear	+	Anger	-	Not fear Not surprise Not happiness Not sadness Not neutrality	Hate
18.	Fear	-	Anger	-	Not fear Not surprise Not happiness Not sadness Not neutrality	Anger
19.	Fear	#	Anger	-	Not fear Not surprise Not happiness Not sadness Not neutrality	Remorse

4. Preference rule of user's affective state = Surprise

Table C- 11 Cognitive reasoning preference rules for user surprise

Rule #	User-affect	Affect-value	System-react	Concern-value	Thermometer	Cognitive-reaction
1.	Surprise	+/#	Not fear Not surprise	+	Not fear Not surprise	Normal
2.	Surprise	+	Not fear Not surprise	-	Not fear Not surprise	Disliking



**Thesis Report: My\_Eliza, a Multimodal Communication System**  
**Appendix C Affective Knowledge Base**

Rule #	User-affect	Affect-value	System-react	Concern-value	Thermometer	Cognitive-reaction
			Not sadness		Not disgust Not anger Not sadness	
3.	Surprise	+		#		Pride
4.	Surprise	-	Not fear Not surprise	+	Not fear Not surprise Not anger	Normal
5.	Surprise	-/#	Happiness	-	Happiness	Reproach
6.	Surprise	-		#		Gratitude
7.	Surprise	#		#		Gratification
8.	Surprise	+/-	Sadness	-	Happiness	Distress
9.	Surprise	+/-/#	Surprise	-	Happiness	Hope
10.	Surprise	+/-/#	Surprise	-	Happiness	Fear
11.	Surprise	-/#	Not fear Not surprise Not neutrality Not happiness Not sadness	-	Happiness	Disliking
12.	Surprise	+/-/#	Not fear Not surprise	-	Not fear Not surprise Not happiness Not anger	Distress
13.	Surprise	+/-/#	Not fear Not surprise	-	Not happiness Not neutrality Not disgust Not anger Not sadness	Fear
14.	Surprise	+/-/#	Not fear Not surprise	+	Not happiness Not neutrality Not disgust Not anger Not sadness	Hope
15.	Surprise	+/-/#	Not fear Not surprise	-	Anger	Anger

**5. Preference rule of user's affective state = Disgust**

**Table C- 12 Cognitive reasoning preference rules for user disgust**

Rule #	User-affect	Affect-value	System-react	Concern-value	Thermometer	Cognitive-reaction
1.	Disgust	+/#	Not fear Not surprise	+		Normal
2.	Disgust	+	Not fear Not surprise	-	Not fear	Disappointment
3.	Disgust	+		#		Pride
4.	Disgust	-	Not fear Not surprise	+	Not fear Not surprise	Uncertain
5.	Disgust	-	Not fear Not surprise Not disgust Not anger	-	Not sadness Not fear Not surprise Not anger Not disgust	Disliking
6.	Disgust	-		#		Gloating
7.	Disgust	#	Not fear Not surprise	-	Not disgust Not anger	Resentment
8.	Disgust	#		#		Gratitude
9.	Disgust	+/-/#	Not happiness Not sadness Not disgust Not anger Not neutrality	+		Hope
10.	Disgust	+/-/#	Not happiness Not sadness Not disgust Not anger Not neutrality	-		Fear





Rule #	User-affect	Affect-value	System-react	Concern-value	Thermometer	Cognitive-reaction
11.	Disgust	-	Not happiness Not sadness Not surprise Not fear Not neutrality	-	Not disgust Not sadness Not surprise Not fear Not anger	Hate
12.	Disgust	+/#	Not happiness Not sadness Not surprise Not fear Not neutrality	+	Not sadness Not disgust Not fear Not neutrality	Disliking
13.	Disgust	-	Not fear Not sadness Not surprise Not anger Not neutrality	+	Not happiness Not disgust Not anger Not fear Not neutrality	Normal
14.	Disgust	-	Happiness	-	Sadness	Disliking
15.	Disgust	-	Sadness	-	Sadness	Distress
16.	Disgust	-	Not surprise Not fear	-	Surprise	Fears-confirmed
17.	Disgust	-	Happiness	+	Fear	Hope
18.	Disgust	+/-	Not surprise Not fear	-	Fear	Fears-confirmed
19.	Disgust	-/#	Not surprise Not fear Not happiness Not sadness	+	Fear	Normal
20.	Disgust	-	Not surprise Not fear Not anger Not disgust Not neutrality	+	Not happiness Not sadness Not surprise Not fear Not neutrality	Normal
21.	Disgust	-	Not surprise Not fear	-	Not happiness Not sadness Not surprise Not fear Not neutrality	Hate
22.	Disgust	-	Not surprise Not fear Not happiness Not sadness	+	Not happiness Not sadness Not surprise Not fear Not neutrality	Disliking

## 6. Preference rule of user's affective state = Anger

Table C- 13 Cognitive reasoning preference rules for user anger

Rule #	User-affect	Affect-value	System-react	Concern-value	Thermometer	Cognitive-reaction
1.	Anger	+/#	Not fear Not surprise	+/-	Not fear Not surprise Not disgust Not anger	Uncertain
2.	Anger	+/-		#		Pride
3.	Anger	-	Not fear Not surprise	-/+	Not fear Not sadness Not disgust Not anger Not surprise	Fear
4.	Anger	#		#		Gratification
5.	Anger	-	Not fear Not surprise	+/-	Sadness	Remorse
6.	Anger	+/-/#	Not happiness Not sadness Not disgust Not anger Not neutrality	+/-		Fear
7.	Anger	+/-/#	Not fear Not surprise	+	Not happiness Not sadness Not surprise	Uncertain



Rule #	User-affect	Affect-value	System-react	Concern-value	Thermometer	Cognitive-reaction
					Not fear Not neutrality	
8.	Anger	+/-/#	Not fear Not surprise	-	Disgust	Hate
9.	Anger	+/-/#	Not fear Not surprise	-	Anger	Anger

## 7. Preference rule of user's affective state = Neutrality

Table C- 14 Cognitive reasoning preference rules for user neutrality

Rule #	User-affect	Affect-value	System-react	Concern-value	Thermometer	Cognitive-reaction
1.	Neutrality	+	Not fear Not surprise	+	Not fear Not sadness Not disgust Not anger Not surprise	Joy
2.	Neutrality	+/-/#	Not fear Not surprise	-	Not fear Not sadness Not disgust Not anger Not surprise	Uncertain
3.	Neutrality	+/#		#		Gratification
4.	Neutrality	-	Not fear Not surprise	+	Not fear Not sadness Not disgust Not anger Not surprise	Normal
5.	Neutrality	-		#		Gloating
6.	Neutrality	#	Not fear Not surprise	+	Not fear Not sadness Not disgust Not anger Not surprise	Love
7.	Neutrality	+/#	Not fear Not surprise	+	Sadness	Hope
8.	Neutrality	+/-/#	Not fear Not surprise	-	Sadness	Distress
9.	Neutrality	-	Not fear Not surprise	+	Sadness	Sorry-for
10.	Neutrality	+/-/#	Not happiness Not sadness Not disgust Not anger Not neutrality	+		Hope
11.	Neutrality	+/-/#	Not happiness Not sadness Not disgust Not anger Not neutrality	-		Fear
12.	Neutrality	+/-/#	Not fear Not surprise	+	Not happiness Not sadness Not disgust Not anger Not neutrality	Hope
13.	Neutrality	+/-/#	Not fear Not surprise	-	Not happiness Not sadness Not disgust Not anger Not neutrality	Fear
14.	Neutrality	+/-/#	Not fear Not surprise	+	Not happiness Not sadness Not fear Not surprise Not neutrality	Normal
15.	Neutrality	+/#	Not fear Not surprise	-	Disgust	Disliking
16.	Neutrality	-	Not fear	-	Disgust	Hate



Thesis Report: My\_Eliza, a Multimodal Communication System  
**Appendix C Affective Knowledge Base**

Rule #	User-affect	Affect-value	System-react	Concern-value	Thermometer	Cognitive-reaction
			Not surprise			
17.	Neutrality	+/-/#	Not fear Not surprise	-	Anger	Anger



## Appendix D Symbols and Notations

This appendix provides the reader with a short explanation about symbols and notations that is used in design process of My\_Eliza system. We use flowchart to design process flow in my\_Eliza system and UML to design the whole system. This chapter is organized as follows. First, Short description about flowcharting is presented in section D.1. Finally the UML diagram is explained in section D.2.

### D.1 Flowcharting

Flowcharting is a graphical representation of the sequence of all operations, movements, inspections (a.k.a. approvals), delays, decisions and storage activities of a process [HAN02]. Flowcharting uses symbols that have been in use for a number of years to represent the type of operations and/or processes being performed. The standardized format provides a common method for people to visualize problems together in the same manner. The use of standardized symbols makes the flow charts easier to interpret, however, standardizing symbols is not as important as the sequence of activities that make up the process. We can use flowcharts for:

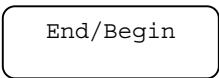
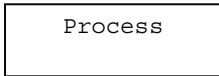
- Documents process and interrelationship of process steps.
- Identifies actual and ideal paths that any product or process flows.
- Identifies problems and potential improvements.
- Can be completed on entire processes assemblies with all components, one person or component through a process, combinations of people and machines, transactions following forms or other documents.

There are three basic types of flowcharts, they are:

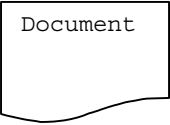


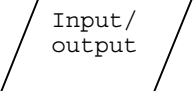

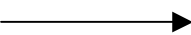
1. *Functional Chart* - A chart that is used to describe how activities interact with one another within an organization as well as with other organization and/or systems.
2. *Process Flow Chart* - A chart that is used to describe the sequence and relationship of the tasks that makes up an activity.
3. *Process Flow Description Chart* - A detailed description of the tasks outlined in a Process Flow Chart. Typically used to show the kinds of tasks performed within a process; the number of operations, review, and transfers; and the amount of storage and time required to complete an activity.

In my\_Eliza design process, we use *Process Flow Chart* to describe the sequence and relationship of the processes in my\_Eliza system. Therefore we only describe the symbols that belong to this type in table D-1 below. For purposes of this table, only the most useful basic symbols used for industrial engineering and process writing are covered in this area. Flowcharts use special shapes to represent different types of actions or steps in a process. Lines and arrows show the sequence of the steps, and the relationships among them.

**Table D- 1 Process flowchart basic symbols**

<i>Symbol</i>	<i>Description</i>
	The terminator symbol marks the starting or ending point of the system. It usually contains the word "Start" or "End."
	A box can represent a single step ("add two cups of flour"), or and entire sub-process ("make bread") within a larger process.



<i>Symbol</i>	<i>Description</i>
 Document	A printed document or report.
 Stored data	A stored data.
 Decision	A decision or branching point. Lines representing different decisions emerge from different points of the diamond.
 Input/ output	Represents material or information entering or leaving the system, such as customer order (input) or a product (output).
 Conne ctor	Indicates that the flow continues on another page, where a matching symbol (containing the same letter) has been placed.
 Flow	Lines indicate the sequence of steps and the direction of flow.

A flowchart illustrates the steps in a process. By visualizing the process, a flowchart can quickly help identify bottlenecks or inefficiencies where the process can be streamlined or improved.

## **D.2 UML Diagram in My\_Eliza Design**

UML (Unified Modeling Language) is a modeling language that is developed by Grady Booch, Jim Rumbaugh, and Ivar Jacobson [ERI98]. UML describes a system in several views, such as:

1. *Use case view* – A view that describes the functionality of the system that is expected by the actor(s).
2. *Logical view* – A view that describes how functionality is designed in the system, in a static form of the system and in dynamic behavior of the system.
3. *Component view* – A view that describes organization of coding component.
4. *Concurrency view* – A view that describes a concurrency in the system, including a case of communication and synchronization in the concurrence system.
5. *Deployment view* – A view that describe the application of the system inside physical architecture of computer and its peripheral, which is called *nodes*.

Every view above is described in a number of diagrams that contains information stressed in certain aspects of the system. The explanation of diagrams in UML that is used in my\_Eliza design is presented below.

### **D.2.1 Use Case Diagram**

Use case diagram shows use case elements. Use case elements represent functional aspects of the system and the actor(s) (other systems or human) as external elements that connects to the

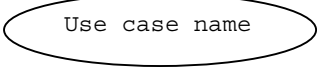

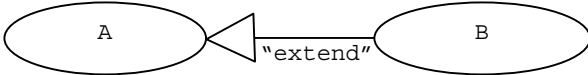
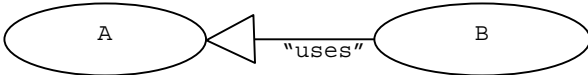


system. Every use case can be described in detailed by textual description, condition diagram or activity diagram. *Relation* represents relationship between use cases with the actor(s) or between use cases themselves. There are three standard of relation, such as:

1. *Communicates* – This relation is describes participation an actor in a use case. This is the only relation between an actor and a use case.
2. *Extends* – An extends relation of a use case A to a use case B indicates that use case B can includes some behavior in A but not all behavior of A can be used by B.
3. *Uses* – A uses relation of a use case A to a use case B indicates that B can use all behavior A.

Use case diagram consists of collection of actor graphs, use cases union inside the system boundary, communication between actors and use cases, and generalization between use cases. The summary of use cases notations can be seen table D-1 below.

**Table D- 2 Use case diagram notations**

<i>Symbol</i>	<i>Description</i>
	A use case
	An actor
	An extend relation
	An uses relation

## D.2.2 Class Diagram

A class diagram is described a static aspect of the system by showing classes and relationship between classes. Every class is divided into several *compartments*. UML does not define the syntax o each compartment therefore a designer may use his/her own syntax. UML has defined three compartments for each class: class name, attributes and methods. The designer may add other parts fit to the needs. Attributes describe the characteristic of the user. An attribute has a type that may a simple type such boolean, real and enumeration or may also class type. Methods in a class describe what the class can do. Methods can be used to manipulate attributes or do other actions. Attributes and methods have three types of visibility, such as:

1. *Public*, means visible to every class and they can access it.
2. *Private*, means not visible by other classes.
3. *Protected*, means only visible to its subclasses.

A class diagram consists of class and relationship between classes. There are four types of relationships:

1. *Association* – Relationship between class and object from this class. It represents that this object class is associated to each other.
2. *Aggregation* – It is a special case of association. It shows the relationship between classes is *whole-part*. An example of aggregation is a car has four wheel, an engine, and so on.



3. *Aggregation Composition* – This aggregation has a tight owner relationship. The *whole* part holds the owner of *part* part.
4. *Generalization* – A relationship between general classes and specific classes. A specific class (namely subclass) inherits all behavior of general class (namely superclass). Attributes, methods, other parts and all associations are inherited by subclass.

A class diagram consists of all static elements above such as classes and their relationships that connected as graph. Table D-3 shows the summarization of diagram class notations.

Table D- 3 Class diagram notations

Symbol	Description
	A class
<ul style="list-style-type: none"> <li>- Can use arrow after or before an association name to show association flow</li> <li>- One of the ends of an association can be added arrow to show the navigation flow.</li> </ul>	An association
<p>A is a whole class and B is part class</p>	An aggregation
<p>A is a whole class and B is part class</p>	An aggregation Composition
<p>B inherit A</p>	A generalization

### D.3 Event Transition Diagram

An event transition diagram shows interaction between objects in a certain time chronologies. This diagram shows an object involved in an interaction with its lifetime through lifetime, activation, and message of the object that are exchanged by the object. A lifetime of an object shows an object type that plays a certain role in the interaction. An activation shows time and duration activity of an object. A message shows the message that brings information and operation to execute an action. There are several types of messages: procedure call, return (after procedure call), branches and iterates.

An event transition diagram shows procedural message explicitly. An event transition diagram is the best tool for real time specifications and complex scenarios. This diagram has two dimensions, such as: a vertical dimension shows time and a horizontal dimension shows an object. Communication between objects is represented by horizontal line. Usually, this diagram is read from top to bottom to see the messages exchange of objects. Table D-4 shows the notations of event transition diagram.





**Table D- 4 Event transition diagram notations**

<i>Symbol</i>	<i>Description</i>
	An object life time
	A activation
	A message <ul style="list-style-type: none"> <li>- Branches</li> <li>- Procedure call</li> <li>- Return</li> <li>- Iteration</li> </ul>


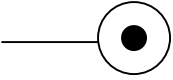
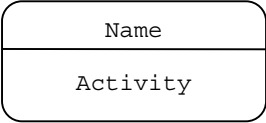
## D.4 State Transition Diagram

A condition or state transition diagram shows an object, system, and subsystem life cycle. This diagram shows the status of an object or a system and how an event (message, error or true/false condition) influences the status. Table D-5 shows the notations of this diagram with their description.

**Table D- 5 State transition diagram notations**

<i>Symbol</i>	<i>Description</i>
	A transition, a relationship between two states shows that an object is in the first status and will move to the second status. This changing state is influenced by filling in certain event(s) and transition condition(s).



<i>Symbol</i>	<i>Description</i>
	A beginning state
	An ending state
 <p>Every state has one or more compartments. Every part is optional.</p>	<p>A state, a condition in object's life that fits to certain condition. An object does an action and waits an event in a certain state.</p> <p>UML defines two compartments:</p> <ul style="list-style-type: none"> <li>- Name – contains a string to shows the state name</li> <li>- Activities – consist of internal actions as a causal of an event. This action does not change a state of the object.</li> </ul>



## Appendix E Scientific Paper

### My\_Eliza,

## A Multimodal Communication System

Siska Fitrianie and Dr. Drs. L.J.M. Rothkrantz

Knowledge Based Systems, Department of Mediamatics

Faculty of Information Technology and Systems

Delft University of Technology

August 2, 2002

email: [s.fitrianie@kbs.twi.tudelft.nl](mailto:s.fitrianie@kbs.twi.tudelft.nl)

### Abstract

My\_Eliza is a computer model for a multimodal communication system, a combination of natural language processing and nonverbal communication. The development of this system is based on a famous question-answering system - QA system, Weizenbaum's Eliza [WEI66]. A human user can communicate with the developed system using typed natural language. The system will reply with text-prompts and appropriate facial-expressions. In order to communicate using a nonverbal facial display, the system should be able to process natural language and emotional reasoning. A first prototype as a proof of concept has been developed that consists of a dialog box, an emotional recognizer based on stimulus response, and a facial display generator.

To implement the dialog box, the work of Wallace, A.L.I.C.E [WAL95], has been used as a starting point. My\_Eliza system has a rule engine that determines current system's affective state as reaction to the user's string input and conversation content.

### 1. Introduction

The way QA systems represent and retrieve information is transparent by their **memory structure**. The memory structure functions as the systems' "brain" and is the foundation of the ability level of the system to "speak" in human natural language. The QA system retrieves the information from its memory and uses syntactic and semantic analysis to output a string as an answer to the user's string input.

Eliza worked by simple pattern-matching operation and substitution of keywords. First, the system identifies the "most important" keyword occurring in the user's input string. Next, it chooses an appropriate transformation rule and its mechanism. There are two transformation rules that are associated with certain keywords.

*Decomposition rule* serves to decompose a data string according to certain criteria (pattern). *Reassemble rule* serves to reassemble a decomposed string according to certain assembly specifications (reply sentence). If Eliza finds a keyword, she will pattern-match the string input against each decomposition rule for that keyword. If it matches, she randomly selects one of the reassemble rules (for that decomposition rule). Finally, Eliza uses a selected reassemble rule to construct the reply. The keyword lists, and the list of decomposition rules and reassembly rules are constructed in a script, which controls all the behavior of Eliza. Figure 1 displays an example of one unit Eliza's memory structure (asterisk sign shows that it can contain any words or phrases).

```
keyword: your
decomposition rule: * your *
  reassemble rule: Why are you concerned about
                    my (2) ?
  reassemble rule: What about your own (2) ?
  reassemble rule: Really, my (2) ?
decomposition rule: ...
  reassemble rule: ...
. . .
Example fragment:
User : What is your name?
Eliza: What about your own name?
User : Only your name, please!
Eliza: Really, my name, please?
User : Just tell me your name!
Eliza: Why are you concerned about my name?
```

**Figure 1 Example of one unit Eliza's memory structure**

The pattern matching operation of the original Eliza still has three major problems [SIM70]: (1) lack of anaphoric analysis, it cannot use previous question-answers to keep the continuity of the conversation content and to store information about the user, (2) lack of ability to restrict the conversation on its topic and (3) lack of ability to get the meaning beyond the sentence.

Another limitation of Eliza system is that users can only communicate with Eliza by exchanging text prompts.



However beyond speech, human people can express their feelings or thoughts through the use of their body, facial expressions, and tone of voice. As indicated by Mehrebian [KIN97], it is proved that about 55 percent of the emotional meaning of a message is communicated through the nonverbal channel, which includes gestures, postures, and facial signals. Nonverbal communication is behavior other than spoken or written communication that creates or represents meaning. Human face-to-face conversation has provided an ideal model for designing a multimodal human-computer interaction (HCI) [TAK93], [SCH00]. Characteristics of face-to-face conversation are the multiplicity and multi modality of the communication channel. Multimodal user interfaces are interfaces with multiple channels that act on multiple modalities. Conversation is supported by multiple coordinated activities of various cognitive levels. As a result communication becomes highly flexible and robust, so that failure of one channel is recovered by another channel and a message in one channel can be explained by another channel. This is the basic idea how a multimodal HCI should be developed to facilitate realistic human-machine interaction.

Nowadays, as computer acts as electronic secretaries or communication mediators, they become common entities in human society [ELL94], [NAK99]. The capability of communicating with humans using both verbal and nonverbal communication channels would be essential. This will surely make interaction between computers and humans more intimate and human-like [LEE99], [PRE01], [CAS94]. Face to face communication is inherently natural and social for human-human interaction and substantial evidence suggest this may also be true for human-computer interaction. Using human-like faces as means to communicate have been found to provide natural and compelling computer interfaces.

Eliza has shocked AI community because it gave the impression of deep semantic linguistic processing but it was in fact based on shallow language processing. Many people become emotionally involved with the QA system. Automating the recognition of users' emotion would therefore be highly beneficial in order to give a proper user reply, both in the verbal channel and in the nonverbal channel. In recent advances of QA systems, facial expression recognition and adapting life-like agents open up the possibility of automatic emotion recognition from user interaction in conversation between human and computer. Emotions are an essential part of human lives; they influence how human think and behave and how human communicate with others, and facial displays are human primary means of communicating emotion [VEL97], [SCH00]. However, there are only a few researches involving research on human emotion recognition, because it is difficult to collect a large amount of utterances that

contain emotion [NAK99]. Only a few of them work in recognizing emotion from text and none of them work in facilitating emotion recognition in a QA system. Moreover, the interpretation of emotion eliciting factors is strongly situation and culture dependent [WIE99].

As a first step in achieving automatic analysis of human behavior and face-to-face communication, automated emotion recognition in human conversation between the users and a QA system has been investigated. This paper discusses the results of the research, which ensued in the development of the **my\_Eliza** – an advance version of the original Eliza. My\_Eliza was aimed at the design and establishment of a QA system of a *semi* automated emotion recognition from human user written conversation. A user or client can communicate with the system using **typed natural language**. The system will reply by text-prompts and appropriate facial-expressions.

The problem of automating emotion recognition and generating appropriate nonverbal facial displays on a QA system as defined in this research comprises into three sub-discussions: (1) automatic generation of system's reply text prompts with ability of anaphoric analysis and ability to respond the conversation based on its topic (2) semi automatic emotion recognition of user's affective state and its intensity, and (3) automatic facial display selection from a facial expressions database based on emotion analysis.

In conversation, my\_Eliza displays two kinds of emotional expressions: first, related to stimulus response when she hears the utterance and second, related to cognitive processing when she realizes the situation and the conversation content to convey her reply sentences.

## 2. Natural Language Processing

Nowadays, a QA system is also called *chatbot* - a short for "chatter" and "bot" [LAV96], spreading in Internet. Bot is short for "robot". A.L.I.C.E is an example of this class of programs. Tackling the three limitations of Eliza above, Wallace proposed to expand memory structure using an extended-XML (Extensible Markup Language) script specification for programming the memory structure for a QA system, called AIML (Artificial Intelligence Markup Language) [WAL95]. The most important AIML units are [BUS01]:

- **<aiml>**, the tag that begins and ends an AIML document.
- **<category>**, the tag that marks a "unit of knowledge" in the system's memory structure.
- **<pattern>**, the tag that contains a simple input pattern rule that matches what a user may type.
- **<topic>**, the tag that contains current conversation topic pattern rule.



- **<that>**, the tag that refers to system's previous reply as a history pattern rule.
- **<template>**, the tag that contains the response to a user input.

In Eliza, **<pattern>** tag part is namely the *decomposition rule* and **<template>** tag is the *reassemble rule*. Figure 2 displays an example of A.L.I.C.E.'s memory units in a topic about name. **<set>** and **<get>** tags are used to store information during conversation. See also [BUS01], and [WAL95]. A.L.I.C.E. has much more possibilities of reply sentences based on their topic and history. Using AIML gives the possibility to create new content by a dialog analysis.

```
<category> <that>*</that>
<pattern>WHAT IS YOUR NAME</pattern>
<template>My <set_topic>name</set> is
    <bot name="name">.</template>
</category>

<topic name="NAME">
<category><that>MY NAME IS *</that>
<pattern>CAN I CALL YOU *</pattern>
<template><random>
    <li> <star/>? Huh! Like I've told you my name
        is <bot name="name">.</li>
    <li> You can call me whatever you like</li>
</random> </template>
</category>

<category><that>MY NAME IS *</that>
<pattern>I HATE *</pattern>
<template><random>
    <li> I don't care, you can only call me,
        <bot name="name">.</li>
    <li> Why? A sad memory perhaps?</li>
</random> </template>
</category>
</topic>
...
Example fragment:
User : What is your name?
Alice: My name is Alice
User : Can I call you Madonna?
Alice: Madonna? Huh! Like I've told you my name
        is Alice.
User : I hate that name.
Alice: Why? A sad memory perhaps?
```

Figure 2 An example of A.L.I.C.E.'s memory units

The matching operation is word-by-word, not category-by-category. The algorithm searches the best match pattern by ensuring that the most specific pattern matches first basically it finds the longest pattern matching an input. If there are two identical patterns but the later contains the same **<that>** tag, then it will take precedence over the other categories, if inside **<that>** tag matches the previous response. Any categories that are contained within a **<topic>** tag will be searched first if the current topic matches it. If neither of above is true, there is a default category with **<pattern>\*</pattern>**. We used this AIML schema to build my\_Eliza's memory structure.

### 3. Nonverbal Communication

This section deals with emotion reasoning and facial display generator. The main goal here is to explore the issues of design and implementation of a nonverbal QA

system that could recognize the user's emotion and show a proper facial display accordingly. In general, three steps can be distinguished in tackling this issue: (1) define which and how many emotions can be recognized by the system, (2) define mechanisms for extracting emotion-eliciting factors in the observed text prompt, which devise the categorization mechanism and the emotion interpretation mechanism, and (3) define some set of categories of emotions that we want to use for facial displays classification and facial displays generation mechanism.

Currently, the interpretation of the emotion-eliciting factor is still semi automatic since we assume to use the memory structure approach of Weizenbaum's or Wallace's pattern matching operation. The memory structure of this approach does not store the semantic meaning of the text. It needs human intervention to interpret the affective semantic meaning.

### 4. Emotion Classification

How many and what kind of emotional expressions are to be treated in a QA system are interesting but difficult issues. In this research we investigate three classification methods:

1. Reddy's [RED01] basic emotions: every emotion is either pleasant or unpleasant and every emotion has a varying intensity regarded as either shaping one's goals or reflecting one's goals.
2. Ekman and Friesen's [EKM75] seven universal emotions: neutrality, happiness, sadness, anger, fear, disgust, and surprise, in terms of facial expressions and mainly concentrated on primary or archetypal emotions, which are universally associated to distinct expressions.
3. Ortony, Clore and Collins theory's twenty-four emotions [BAZ01], [ELL93] (OCC's theory 1988, see table 1). It is based on grouping human emotions by their eliciting conditions events, their consequences of their action, and their selections of computational implementation. They are resulted in three branches: (1) *Attraction* relates to emotions that are arising from aspects of the object, (2) *Consequences of event* relates to reaction of others' fortunes and (3) *Attribution* relates to approval of self or other. In addition, there is a *compound* class that involves the emotions of gratification, remorse, gratitude and anger.

Table 1 Twenty-four emotion types according to OCC's theory

Name and Emotion Type
<b>Joy</b> : pleased about an event
<b>Distress</b> : displeased about an event
<b>Happy-for</b> : pleased about an event desirable for another
<b>Gloating</b> : pleased about an event undesirable for another
<b>Resentment</b> : displeased about an event desirable for another
<b>Sorry-for</b> : displeased about an event undesirable for another
<b>Hope</b> : pleased about a prospective desirable event



Name and Emotion Type
<b>Fear:</b> displeased about a prospective undesirable event
<b>Satisfaction:</b> pleased about a confirmed desirable event
<b>Relief:</b> pleased about a disconfirmed undesirable event
<b>Fears-confirmed;</b> displeased about a confirmed undesirable event
<b>Disappointment:</b> displeased about a disconfirmed desirable event
<b>Pride:</b> approving of one's own act
<b>Admiration:</b> approving of another's act
<b>Shame:</b> disapproving of one's own act
<b>Reproach:</b> disapproving of another's act
<b>Liking:</b> finding an object appealing
<b>Disliking:</b> finding an object unappealing
<b>Gratitude:</b> admiration + joy
<b>Anger:</b> reproach + distress
<b>Gratification:</b> pride + joy
<b>Remorse:</b> shame + distress
<b>Love:</b> admiration + liking
<b>Hate:</b> reproach + disliking

Since classifications of some emotion eliciting factors are in a gray area, in this research, we add one emotion type: uncertainty.

#### 4.1. Emotion Eliciting Factor Extraction

Most of developed systems that are able to devise emotion-eliciting factor information still need manual human intervention. Following three experiments dealing with representing and extracting emotions' information on the system's memory structure and how we map them in my\_Eliza:

##### 1. Emotive lexicon dictionary look-up parser.

This approach uses a list of lexicons associated to different type of emotions. Those lexicons, which are composed by words or phrases, are selected from the way human people expresses their feelings with its intensity. The system uses a shallow word matching parser to extract affective state from the context. Elliott [ELL92], [ELL93] used this approach for his model of a multi-agent world where each agent is able to reason about emotion episodes that take place in one another's lives. He used an extended base lexicon of spoken phrases that includes 198 emotion words associated with twenty-four OCC's theory emotion types. Those words describe relationship, mood and emotional intensity. Each emotion type has a set of eliciting conditions. When the eliciting conditions are met, and various thresholds have been crossed, corresponding emotions result. The system applies minimal the detection of user's emotional inflection. Using this approach, it allows the user to teach the computer keywords in a new vocabulary relatively quickly and the system remains understandable no matter in which context the user is.

My\_Eliza uses this approach to extract emotion-eliciting factor information in the text prompt in the conversation both of the user's string input and the

system's reply sentence. Since the first prototype is dedicated as a "proof of concept", only six universal emotion types (Ekman's) will be used in emotive lexicons classification instead of twenty-four OCC's theory emotion types. We define six dictionaries containing lexicons in the following form: [*<lexicon>*: *<intensity value>*] with *<intensity value>* is an integer value [1..3]. We also define six affective counters *C* for each emotion type. The parser parses the sentence word-per-word against the dictionary. If it finds the same emotive lexicon in the dictionary, it will calculate the counters using following equations:

$$\forall \text{Lexicon } l_i \in d_i \mid C_{i(t)} = C_{i(t-1)} + I_i \cdot s ;$$

*i* = active emotion type  
*I* = intensity level; *s* = summation factor

$$\forall j \neq i \mid C_{j(t)} = C_{j(t-1)} - \text{distance}[j, i]$$

*j* = {happiness, sadness, anger, fear, disgust, surprise}

For the first prototype we use Hendrix and Ruttkay's distance values between expression emotions ([HEN98], table 2) for distance[*j*, *i*]. The result of this calculation is the candidate of affective state both for the user and the system, which is taken from the emotion type with the highest level of all counters.

Table 2 Distance value between emotions [HEN98]

	Happiness	Surprise	Anger	Disgust	Sadness
Happiness	0	3.195	2.637	1.926	2.554
Surprise		0	3.436	2.298	2.084
Anger			0	1.506	1.645
Disgust				0	1.040
Sadness					0

##### 2. Emotive labeled memory structure extraction.

This approach labels each unit of memory structure with one or more of the emotions types. Most of the examples for systems using this approach are automatic story telling systems and automatic digitizer for cartoon movies. Each dialog sentence of each actor is labeled with an emotion type and decomposed in its phonological representation. Therefore, the system can show appropriate intonation and nonverbal display when it reads the dialog. Pelachaud et.al. [PEL94] used it in their research by assuming the input as a file containing an utterance already decomposed and written in its phonological representation with its prosody in its bracketed elements. At each input, it specifies the desired affectual parameters and their intensity. Using this approach means to sidestep the issue of emotion recognition. The modeling of affect is not meaning based and it needs human manual work to label each memory unit.

We label my\_Eliza's memory structure by adding two additional tags in AIML schema: *<affect>* tag that labels the user's affective situation and *<concern>* tag that labels the system's reaction situation. Inside those two new tags, based on Reddy's basic emotion we define four-possibility emotive situation type: (1) positive "+", (2) negative "-", (3) joking "#", and (4)





normal/any “\*”. Figure 3 (below) displays an example of my\_Eliza’s memory structure units.

```
<category> <affect name="*">
<pattern>WHAT IS YOUR NAME</pattern>
<that>*</that>
<template><think><setconcern>+</setconcern>
<setaffect>+</setaffect></think>My
<set_topic>name</set> is <bot name="name">.
</template>
</affect>
</category>

<topic name="NAME">
<category> <affect name="*">
<that>MY NAME IS *</that>
<pattern>YOUR *</pattern>
<template><random>
<li><think><setconcern>#</setconcern>
</think>Your <star/> also!</li>
<li><think><setconcern>+</setconcern>
</think>Mind that, what is your name?</li>
</random> </template>
</affect></category>

<category> <affect name="*">
<that>MY NAME IS *</that>
<pattern>I HATE YOUR *</pattern>
<template><random>
<li><think><setconcern>-</setconcern>
</think>Why? A bad memory perhaps?</li>
<li><think><setconcern>-</setconcern>
</think>Why?</li>
</random>
<think><setaffect>-</setaffect></think>
</template>
</affect></category>

<category> <affect name="-">
<that>MY NAME IS *</that>
<pattern>YOUR *</pattern>
<template>
<li><think><setconcern>+</setconcern>
</think>Mind that, what is yours?</li>
<li><think><setconcern>-</setconcern>
</think>Don't be a witty!</li>
</random> </template>
</affect></category>
</topic>
```

Figure 3 Example of my\_Eliza’s memory structure units

Using the memory structure units in figure 3, we expect my\_Eliza is able to converse with human user in positive situation (table 3) and in negative situation (table 4).

Table 3 Example of fragment in positive situation











User:	What is your name?	 My_Eliza
 My_Eliza:	Your name is lovely.	
User:	My name is Eliza.	 My_Eliza
 My_Eliza:	Your name is lovely also!	

Table 4 Example of fragment in negative situation:

User:	What is your name?	 My_Eliza
 My_Eliza:	My name is Eliza.	
User:	I hate your name.	 My_Eliza
 My_Eliza:	Why? Bad memory perhaps?	
User:	Your name reminds me to my enemy.	 My_Eliza
 My_Eliza:	Mind that, what is yours?	

Using extra two tags, my\_Eliza has the possibility to have a bigger memory structure than A.L.I.C.E. In matching-operation algorithm, the memory structure with the same situation type will be checked first. If none of them is matches, the system picks a category with a normal situation type (\*).

### 3. Goal based emotion reasoning.

This approach sets some goals, principles, preferences and moods in the system. Extraction emotion-eliciting factors leading to emotions falls into four major categories: those rooted in the effect of an event on the goal of the system, those rooted in the standard and principle invoked by an act of the user, those rooted in tastes and preferences with respect to an object (including the user treated as an object), and lastly a selected combination of the first three categories. Another way to view these categories is that they are rooted in the system’s assessment of the *desirability* or *undesirability* of some event, the *praiseworthiness* or *blameworthiness* of some act, the *attractiveness* or *unattractiveness* of some object, or selected combinations of these assessment. Elliot [ELL93a] also used this approach for his model to simulate social interactions between agents in incorporated models of individual affect and personality. Each agent interprets situations that are characterized in terms of the way they may or may not meet the eliciting conditions of emotions. Agents use a case based heuristic classification system to reason about the emotions of other agents’ personalities that will help them to predict and explain future emotion episodes involving observed agents. Embodied in the simulation system, Elliott used a set of rules for the mapping from four categories emotion-eliciting factors above into twenty-four OCCs theory emotion types.





Mapping to my\_Eliza, we define the system's goals, affective status and preferences (GSP) while she converses with the user. Several goals of my\_Eliza are:

- *Answering questions* – if the user asks something, my\_Eliza's goal is to answer it.
- *Persuasive agreement* – if the user persuades to do something or invites my\_Eliza to do something, my\_Eliza's goal is to show whether she agrees or not.
- *Topical focus* – to keep on conversing on the same topic and beware if it is changing.
- *Explanation statements* – to reply the user's statements that require specification and explanation.
- *Reflecting feeling* – to keep consistent with the user's current affective state.
- *Alignment* – to keep consistent with the system's current reaction affective state (system's status) and system's preferences.

The system recognizes the dialog using a dialog scheme adopted from [CAR95]. By distinguishing the by distinguishing a dialog state as a certain dialog act like a question, statement, acknowledgement, or pause, the system has to know which goal to pursue. Whether a certain goal is appealing or not appealing may influence the system's affective state. We also define my\_Eliza's preference as the personal data about the system and can be used during conversation, for example: her name, birthday, the things she likes or hates, and so on. To be fair, using <set> tag the system also stores the user's personal data during conversation, for example the user's name, birthday, favorite stuff, personal data about family and so on. These data about system's GSP and user's personal data can be used for pragmatic analysis when the system constructs the reply sentence and defines its current affective status.

## 4.2. Emotion Recognition

For the activation of an emotion, [ELL93], [VEL98], [PRE01], and [BAZ01] proposed the use of threshold values by counting all associated elicitation factors, the excitatory (positive) and inhibitory (negative), from other emotions. They used an activation level range [0, max] where *max* is an integer value determined empirically. All emotions are always active, but their intensity must exceed a threshold level before they are expressed externally. The activation process is controlled by a knowledge-based system that synthesizes and generates cognitive-related emotions in the system.

We define six affective thermometers classified by six Ekman's universal emotion types. These thermometers observe the affective state of the system as reaction to the user's string input and the dialog content – the system's reaction affective state. If an emotion is active, the system calculates all of thermometers  $T_i$ , with the following equations:

$$T_i(t) = T_i(t-1) + I_i \cdot s$$

$i$  = an active emotion type

$s$  = summation factor;  $I$  = intensity

$$\forall j \neq i \mid T_j(t) = T_j(t-1) - \text{distance}[j, i]$$

where  $j = \{\text{happiness, sadness, anger, fear, disgust, surprise}\}$

We also use Hendrix and Ruttkay's distance values to calculate those equations. The system takes the highest degree of all thermometers as the most dominant emotion. If all thermometers are equal then the dominant emotion is neutrality.

To determine the system's affective state we formulate two knowledge based systems: (1) determines the system's reaction affective state as stimulus response to the user's input string and (2) determines the system's reaction affective state as the result of cognitive process of the conversation content to convey its reply sentence. We have defined a set of rules that specify the emotion recognition process of the system. We call these rule-sets **preference rules**, since they indicate preferences to exhibit system's "preference" reaction affective state rather than performing explicit actions, such as facial displays. Every rule in the set defines conditions of emotion eliciting factors and the affective thermometers to activate the rule and a preference that is expressed upon activation. The result from each knowledge-based system is one of twenty-four OCC's theory emotion types with addition of two emotion types: normal and uncertainty, for example:

### 1. Preference rule for stimulus response:

This rule will fire the preference first reaction **joy** if the following conditions are met:

- user is happy,
- user asks question,
- situation type of user is not negative,
- current maximum system's affective thermo is happy.

In this case my\_Eliza will answer any questions from the user joyfully, because she enjoys the situation and she met the goal: making the user feel happy.

### 2. Preference rule for cognitive process:

This rule will fire the cognitive processed preference **resentment** if the following conditions are met:

- user is sad,
- system's reply is sad,
- situation type of user is joking,
- situation type of the system is negative,
- current maximum system's affective thermo is sad.

Here my\_Eliza does not like the user makes a joke while she feels sad.

## 4.3. Facial Display Generator

In most of the works in facial display generation are used one to one corresponding facial display and emotions, distinguished by intensity [ELL93], [PRE01]. The other works used the correspondence between communication categorization and Ekman & Friesen's Facial Action Coding System (FACS) [TAK93] and between emotions with FACS [PEL94]. FACS is a notation to describe visible facial expression based on



anatomical studies; how a feature is affected by specifying its new location and the intensity of changes. For the first prototype we use one to one corresponding facial display and emotion. We use twenty-two smiley nonverbal facial display classified by eight emotion types (neutrality, happiness, sadness, anger, surprise, fear, disgust and uncertainty) and three level of intensity (LOW, MEDIUM, HIGH) except for neutrality. Since the system's reaction affective state may be one of twenty-four OCC's theory emotion types we cluster every those emotion types into six Ekman's universal emotion types. We cluster normal into neutrality.

## 5. Design

The architecture of my\_Eliza is illustrated in figure 4, which takes the idea of message passing on a blackboard system. The message flow and message process are always on the blackboard. If a new message comes, it will be analyzed, synthesized, and the result will always be put back on the blackboard. In my\_Eliza, the message is the user's **string input** and the results are the **reply sentences and facial displays**. My\_Eliza works by the following steps:

### 1. Generating a stimulus-response nonverbal signal

- User types a *string input* and puts it on the blackboard system.
- The *Parser* parses the input into words and puts it on the list on the blackboard system.
- The *Lexical Analysis layer* normalizes the string input by eliminating incorrect or incomplete words or phrases and checking relations between words or phrases. This layer puts the result on the blackboard system.

- The *Affective State Analysis layer* activates its two sub layers: *Emotive Lexicon Dictionary Parser* and *Emotive Labeled Memory Structure Extraction*. The *Emotive Lexicon Dictionary Parser layer* identifies the emotive lexicons from the user's input and the reply sentence (after the system has constructed the reply sentence). The *Emotive Labeled Memory Structure Extraction layer* extracts the label from the system's memory unit. Those results are put on the blackboard system.
- The *Syntactic-Semantic Analysis layer* performs a pattern matching operation based on the user's string input pattern to add the user's affective information on the blackboard system. In this step, the system starts to work in parallel with the process to construct system's reply sentence. As a result, *Syntactic-Semantic Analysis layer* performs pattern-matching operation to generate system's reply sentence and puts its candidate on the blackboard system.
- The *Affective Attributing Analysis layer* activates its sub layer, the *Concern of the other Analysis layer*, to perform emotion-based reasoning to deliver current system's reaction affective state and put it on the blackboard system. This analysis based on stimulus response and directly related to the user's input string.
- The *Intensity Analysis layer* processes the message and calculates the current system's affective 'thermometers' and puts the calculation result on the blackboard system.
- The *Facial Selection* selects my\_Eliza's facial display and puts the selection on the blackboard.
- The *Wrap Process* delivers and displays my\_Eliza's **stimulus-response facial display** or the first facial display.

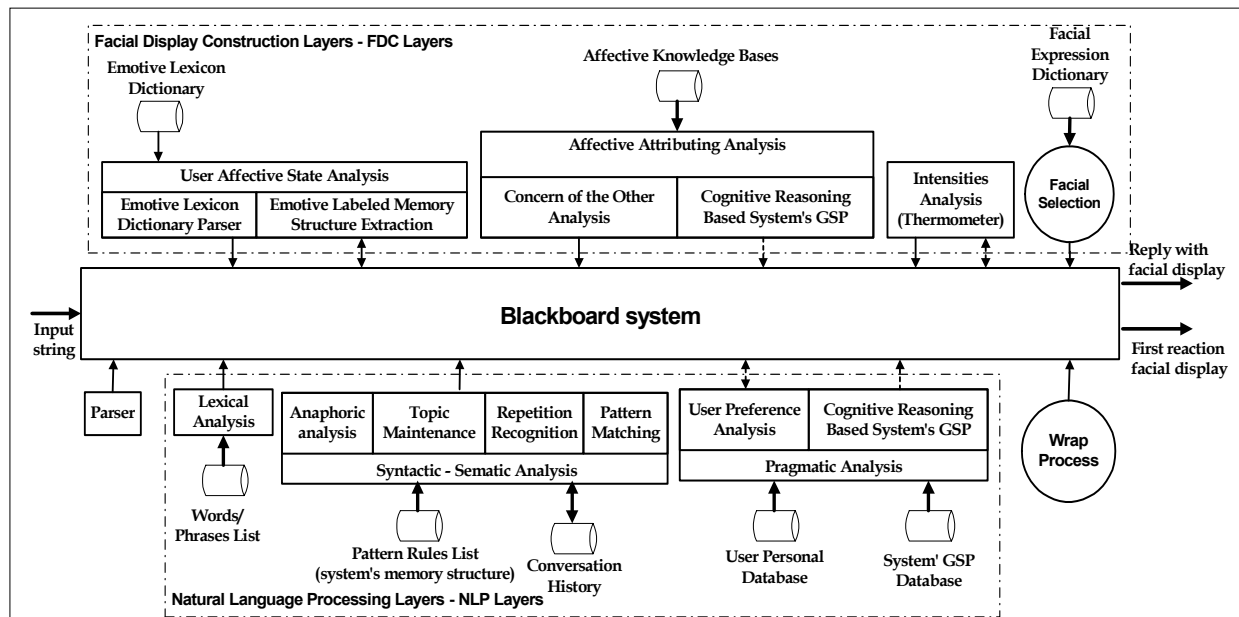


Figure 4 My\_Eliza's blackboard system



## 2. Constructing the candidate of reply sentence and generating a cognitive-processed facial display:

- The *Pragmatic Analysis layer* processes the candidate reply sentence and puts the result on the blackboard system. This layer reviews the candidate of reply sentence whether it violates user's preference and/or system's goals, status and preferences – system's GSP. If it does, the result will be sent back to the *Syntactic-Semantic Analysis Layer* to get new a reply sentence.
- The *Affective Attributing Analysis layer* activates its sub layer, the *Cognitive Reasoning layer*, to perform emotion-based reasoning to deliver current system's reaction affective state and put it on the blackboard system. This analysis is based on cognitive processing of system's GSP, system's reply sentence and user's affective state.
- Again, the *Intensity Analysis layer* processes the message and calculates the current system's affective 'thermometer' level. This layer also puts the result on the blackboard system.
- The *Facial Selection* selects my\_Eliza's facial display and puts the selection on the blackboard.
- The *Wrap Process* delivers and displays my\_Eliza's reply sentence and cognitive-processing result facial display.

## 6. Implementation

There are three incremental implementation layers: (1) create a dialog box that can engage in human conversation based on typed natural language and recognize the user's affective state and the system's reaction affective state, (2) build a stimulus-response of facial displays based on spontaneously spinal brain reasoning on user's string input, (3) build a cognitive processor of facial displays based on anaphora analysis, pragmatic analysis, dialog content and system's goals, status, and preferences.

Currently, we are in the second implementation layer and the result is called my\_Eliza prototype-1. We use Program D A.L.I.C.E [WAL95] as a starting point to build my\_Eliza's dialog box. Program D A.L.I.C.E is written on Java Development Kit version 1.3 and XML, therefore we use a compiler and classes contained in the same languages for my\_Eliza prototype-1. My\_Eliza's dialog box contains many packages most of them derive from Program D A.L.I.C.E. This program has provided a robust client server and multi user communication. Therefore, My\_Eliza is also controlled by a collection of autonomous client-server communication via TCP/IP. The user can communicate with my\_Eliza's server through the HTTP server, that provides the blackboard system. My\_Eliza uses the **Java expert system shell** (Jess) for *affective attributing* knowledge based system shell [FRI00]. Jess is a rule engine and

scripting environment written entirely in Java. The script of the knowledge base is written in .clp.

Currently, my\_Eliza prototype-1's emotive lexicon dictionary contains: 48 lexicons for happiness, 170 lexicons for sadness, 34 lexicons for surprise, 33 lexicons for fear, 93 lexicons for disgust, and 69 lexicons for anger. This prototype also has 1953 categories in its list of pattern rules. Its affective knowledge bases contain 77 preference rules of stimulus response knowledge base and 151 preference rules of stimulus response knowledge base. We can add these databases and knowledge bases easily even while the server is still running. With this benefit we can build the system's memory structure and knowledge base incrementally by trial error. Figure 5 displays the main page of my\_Eliza prototype-1 when she felt sorry-for the user's (namely Siska) misfortune.

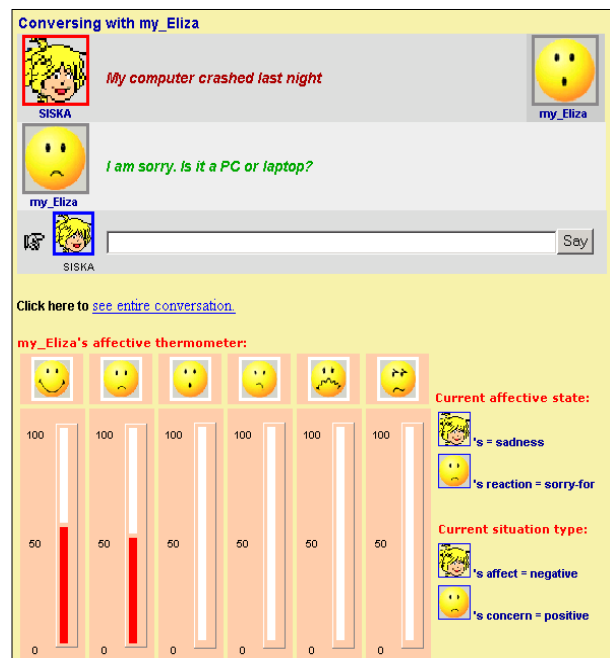


Figure 5 My\_Eliza prototype-1 main page

## 7. Conclusion

Surveying the literature about the emotion recognition process leads to the conclusion that none of the research is particularly in the work of recognizing emotion in a QA system. However, from the work of researchers in the fields of multi agents system, emotion recognition from human speech intonation, automated character animation system and communicative facial display system, could give inspiration which allows us to fit in our approach with the fields. However from most of the work still need human manual intervention. A pragmatic advantage of using AIML form and preference rules to exhibit system's behavior is that the systems' memory structure and its behavior can be extended easily. One of future works in this research is to extend the rule-sets in system's affective knowledge



bases. Current rule-sets that we have implemented in the system do not know the correlation emotion of current dialog and the emotion of entire conversation content. The rules-sets could also be made temporal. We can make specific rules for the opening of the conversation, during discussion or the end of the discussion. By adding more emotion eliciting-factors in the rule-sets, extra rules are needed for new eliciting factors. The system also should have the ability to learn from conversation history. These additions to the system will be valuable assets to add new memory structure units and to add rule-sets of the system to generate its affective knowledge bases autonomous.

Additionally the server interface should have extended functionality to show the thinking process of the system. Realistic virtual environments not only include believable appearance and simulation of the virtual world but also imply the natural representation of participants. That can be fulfilled by visualization of human character embodiment with animation. Moreover, using more possible facial displays, the system is able to convey many different kinds of emotion as different social situation arise. It needs to explore several ways so that real-time, animated, and virtual human characters can be given more intelligence and communication skills, therefore they can act, react, make decisions, and take initiatives. In a similar fashion, the system should be able to communicate with a broad range of conversation topics and it should be able to visually support these conversations with an equally broad range of emotion and expressions behaviors.

## References

- [BUS01] Bush, Noel, *AIML Specification ver. 1.0.1*, <http://Alicebot.org/TR/2001/WD-aiml/>, 2000.
- [BAZ01] Bazzan & Bordini, A Framework for the Simulation of Agents with Emotions, Report on Experiments with the Iterated Prisoner's Dilemma, AGENT'01, Communication of ACM, p292-p299, Canada, 2001.
- [CAR95] Carletta, J., et. al., *The Coding of Dialogue Structure in a Corpus*, in *Corpus-based Approaches to Dialogue Modeling*, 9<sup>th</sup> Twente Workshop on Language Technology, p25-p34, University of Twente, 1995.
- [CAS94] Cassell, J. et.al. *Animated Conversation: Rule Based Generation of Facial Expression, Gesture and Spoken Intonation for Multiple Conversation Agents*, in SIGGRAPH'94, 1994.
- [EKM75] Ekman & Friesen, *Unmasking the Face*, Prentice Hall, New Jersey, USA, 1975.
- [ELL92] Elliott, Clark, *Thesis Report, The Affective Reasoner: A Process Model of Emotions in a Multi-Agent System*, Ph.D. Desertation, Northwestern University, The Institute for the Learning Sciences, Tehnical Report No.32, 1992.
- [ELL93] Elliott & Siegle, *Variables Influencing the Intensity of Simulated Affective States*, In AAAI Technical Report for Spring Symposium on Reasoning about Mental States: Formal Theories and Applications, 58-67, American Association for Artificial Intelligence, 1993.
- [ELL93a] Elliott, *Using the Affective Reasoner to Support Social Simulations*. In Proceeding of the 13<sup>th</sup> International Joint Conference on Artificial Intelligence, 194-200, Chambery, 1993.
- [FRI00] Friedman-Hill, Ernest, *Jess: Rule Engine of the Java Platform*, Sandia National Laboratory, <http://herzberg.ca.sandia.gov/jess/>, USA, 2000.
- [HEN98] Hendix & Ruttkay, *Exploring the Space of Emotional Faces of Subjects without Acting Experience*, ACM Computing Classification System: H.5.2, I.5.3, J.4, <ftp://ftp.cwi.nl/ub/CWIreports/INS/INS-R0013.ps.Z>, 1998.
- [KIN97] King, Donnell, A., *Nonverbal Communication*, <http://www2.pstcc.cc.tn.us/~dking/>, 1997.
- [LAV96] Laven, Simon, *The Simon Laven Page, Chatterbot Central*, [www.simonlaven.com](http://www.simonlaven.com), 1996.
- [LEE99] Lee, Kwanyong, *Integration of Various Emotion Eliciting Factors for Life-Like Agents*, ACM Multimedia'99, Part 2, p155-p158, 1999.
- [NAK99] Nakatsu, Ryohei, et al. *Emotion Recognition and Its Application to Computer Agent with Spontaneous Interactive Capabilities*, Creativity and Cognition 99, Communication of ACM, p135-p143, UK, 1999.
- [PEL94] Pelachaud et.al., *Generating Facial Expression for Speech*, Dept. of Computer and Information Science, University of Pennsylvania, 1994.
- [PRE01] Prendinger & Ishizuka, *Social Role Awareness in Animated Agents*, AGENTS'01 ACM, p270-p276, 2001.
- [RED01] Reddy, W.M., *The Navigation of Feeling, A Framework for the History of Emotion*, Cambridge University Press, Edinburgh, 2001.
- [SCH00] Schiano et.al., *Face to Interface: Facial Affect in (Hu)Man and Machine*, CHI Letters, Communications of the ACM, Vol. II No. 1, p193-p200, 2000.
- [SIM70] Simmons, R.F., *Computational Linguistic, Natural Language Question Answering System*, 1969, Communications of the ACM, Vol. XIII No. 1, p15-p29, January 1970.
- [TAK93] Takeuchi & Nagao, *Speech Dialogue with Facial Displays: Multimodal Human-Computer Conversation*, Agent Interfaces, Section VII, p572-p579, 1993.
- [VEL98] Velasquez, J.D., *Modelling Emotions and Other Motivations in Synthetic Agents*, American Association for Artificial Intelligence (AAAI), [www.aaai.org](http://www.aaai.org), 1997.
- [WAL95] Wallace, Richard, *Alicebot*, <http://www.Alicebot.org>, 1995.
- [WEI66] Weizenbaum, J., *ELIZA – A Computer Program for the Study of Natural Language Communication between Man and Machine*, Communication of the ACM 9(1): p36-p45, 1966.
- [WIE99] Wierzbicka, Anna, *Emotional Universal*, Language Design 2, p23-p99, Australia, 1999.

