A New Graphical Representation of the Chatterbots' Knowledge Base

Maria das Graças Bruno Marietto, Rafael Varago de Aguiar, Wagner Tanaka Botelho, and Robson dos Santos França

Universidade Federal do ABC, Brazil
{graca.marietto,wagner.tanaka}@ufabc.edu.br,
rafael.varago@aluno.ufabc.edu.br,
robsonsfranca@gmail.com

Abstract. This paper discusses chatterbots that use the Pattern Recognition technique and the Artificial Intelligence Markup Language (AIML) regarding their design and implementation. Usually, chatterbot's Knowledge Base (KB) is written with the AIML language, and it is based on the stimulus-response type block approach. Considering that the chatterbots' KB might be a large collection of question-answer pairs, there is an increasing need of visual design proposals for the chatterbots' dialogues. However, there is a lack of projects related to the chatterbots' KB display. In order to fill this gap, this paper introduces a new graphical model called Dialogue Conceptual Diagram (DCD). It is a unified theoretical framework to describe the chatterbot's knowledge in a graphical, intuitive and compact way. Finally, a subset of the ALICE's KB was modeled using the DCDs, showing that DCD is a suitable framework to graphically represent the human-machine dialogues.

Keywords: Artificial Intelligence, Chatterbot, Pattern Recognition, AIML Language, Knowledge and Data Engineering, Graphical Representation.

1 Introduction

According to [1] the purpose of Ambient Intelligence (AmI) is "*...to broaden the interaction between human beings and digital information technology through the usage of ubiquitous computing devices*". Into the ubiquitous computing area researches have been performed for developing the "natural interfaces". Such interfaces are designed to ease the communication process by applying human-like interaction features such as natural language, gestures and vision. Among these interfaces this paper highlights the chatterbots, computational systems designed to interact with humans through natural language. To be more specific, this paper analyzes the chatterbots that use the Pattern Recognition technique and the Artificial Intelligence Markup Language (AIML) [2]. The AIML allows the modeling of natural language dialogues between chatterbots and humans through a stimulus-response approach. For each possible sentence from the user (stimulus) there are pre-programmed replies available in the chatterbot's Knowledge Base

D. Ślęzak et al. (Eds.): AMT 2014, LNCS 8610, pp. 513–524, 2014.

(KB). The ALICE chatterbot was the first bot to use the AIML language and its interpreter.

The chatterbots' KBs builded over the Pattern Recognition framework and the AIML language are becoming a massive collection of data that requires proper methodologies for their processing, considering three operations: storage, retrieval and viewing. However, despite the relevance of the organization and portrayal of chatterbots' KBs, it is unusual to find very few references in the literature about these topics. This observation becomes even more surprising since one can find a considerable number of of chatterbots available to the research community. Therefore, to help filling this gap, this study presents a new graphical model called Dialogue Conceptual Diagram (DCD). It is a theoretical frame of reference to visually model the sequence of the human/chatterbot dialogues.

The formal structure of the DCD's graphical representation - the one proposed by this paper - is used to build the KBs because: (i) it highlights which dialogues the chatterbot is ready to keep with users (the bot's knowledge); (ii) it allows a shared view of the dialogues' structure; (iii) it works as a guide for a critical analysis of the bot's KB; (iv) it eases the recycling and sharing of KBs; (v) the DCDs can be used by a multidisciplinary team, which some members are responsible to build the KB while others are going to deal with the computational matters; (vi) by working with DCDs, the designer can focus on the chatterbot's KB modeling instead of the implementation details; (vii) the abstraction provided by the DCDs allows that a single DCD can be implemented in different ways, working as a blue print for the chatterbot's KB.

This paper is organized as follows. Section 2 describes the syntactical and semantical structure of the Dialogue Conceptual Diagrams. An application of DCDs for a graphical portrayal of the ALICE chatterbot is provided in the Section 3. Lastly, Section 4 outlines the conclusions of this paper.

2 Dialogue Conceptual Diagram

This section introduces the DCD as a frame of reference to graphical representation of dialogues among humans and chatterbots. It is visually depicted as a horizontal flowchart, which the dialogue sequence must be read from left to right. Also, the DCD's structure is composed of predefined elements that are graphically represented by symbols. The dialogue flow is established by the interaction of three components called actors and defined as follows: (i) User: it is the user's data input; (ii) Inference Machine: it is the computational process performed by the chatterbot; (iii) Chatterbot: it is the chatterbot output that is shown to the user.

Figure 1 presents the DCD's basic structure with three actors and their interactions established during the dialogue flow. The flow starts at ① with the user's input as a sentence. After that, such sentence is simplified, normalized and transferred to chatterbot's inference machine at ②. In that stage the Pattern Recognition described in [2] occurs. When the most appropriate answer is found,

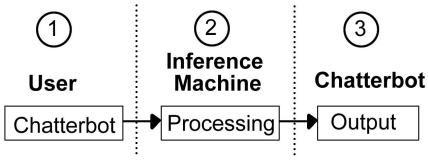


Fig. 1. Dialogue Conceptual Diagram Structure

the inference machine sends the message to be displayed for the user through the output device at ③.

As previously stated, DCDs are built using graphical elements. For each element it is established a syntax and a set of layout rules, as well as semantics related to its meaning. This paper proposes and describes in the next sections ten graphical elements for the DCDs.

2.1 Dialogue Conceptual Diagram's Graphical Elements

This section presents and describes each DCD's graphical element proposed in this work. It is based on its syntactical and semantical structure.

User's Data Input Element. The User's Data Input Element expresses the text informed by the user during the dialogue with the chatterbot. Figure 2(a) shows this element with the text inside as the user's input sentence. This diagrams refers to AIML's <PATTERN> tag.

(a) User's Data Input Element.

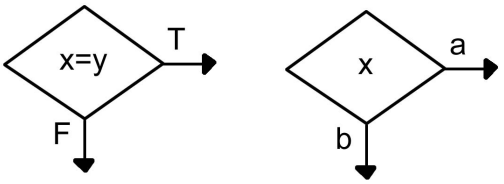
(b) Chatterbot's Data Output Element.

(c) Dialogue Direction Element.

Fig. 2. DCD's Elements: User's Data Input, Chatterbot's Data Output and Dialogue Direction

Chatterbot's Data Output Element. The Chatterbot's Data Output element shown in Figure 2(b) is applied to represent chatterbot's reply to the user. Also, in the AIML language it is implemented using the <TEMPLATE> tag.

Dialogue Direction Element. Figure 2(c) illustrates the Dialogue Direction element, which is used to designate DCD's dialogue flow. It is depicted by a directed line segment with an arrow to indicate the dialogue's flow.



(a) First Scenario. (b) Second Scenario.

Fig. 3. Bivalent Decision Element

Bivalent Decision Element. The Bivalent Decision element is related to the conditional structures, also known as decision or selection structures, which are common in programming languages. This element dictates the running of certain code fragments according to a logical test performed in a control variable.

Into the DCDs' context, the Bivalent Decision element allows the distinction of two scenarios. The first happens when the control variable is compared with a certain value, and such comparison is either true or false. Figure 3(a) shows the scenario with the X control variable being compared to the Y value. If it is true (symbol T), the dialogue is directed to a certain path. Otherwise, the symbol F leads the dialogue to another path.

The second scenario defined in Figure 3(b) happens when the control variable is able to have two values. In the figure, the X control variable can take A or B as a value. There are two ways to program the Bivalent Decision element in AIML: by the <CONDITION> and <LI> tags, or by the <IF> and <ELSE> tags.

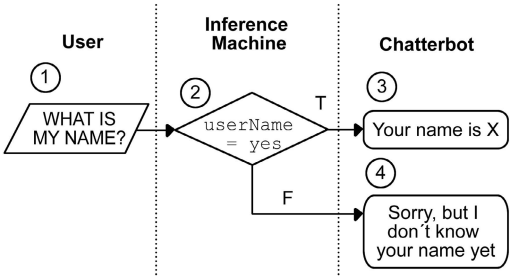


Fig. 4. Dialogue Conceptual Diagram with the Bivalent Decision Element

In order to provide an example of Bivalent Decision element in a dialogue context, Figure 4 shows a DCD that, at first in ① the user enters with the sentence "WHAT IS MY NAME?". After that, the inference machine, using the Bivalent Decision element in ② evaluates the conditional test upon the USERNAME control variable. This variable states whether the chatterbot has the user name in its memory or not. If this variable contains the YES value - implying that the user has already informed his/her name - the outcome from the comparison is

the true value. Thus, the reply from the chatterbot is "Your name is X" in ③, as X being the user name stored in the chatterbot's memory. However, if the comparison ends up in a false value, the reply becomes "Sorry, but I don't know your name yet" in ④.

Listing 1.1 presents the AIML code that matches with the DCD shown in Figure 4.

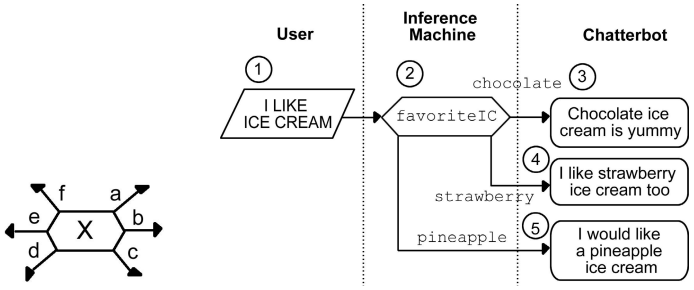
Listing 1.1. AIML Code for the DCD in Figure 4

```
<?xml version="1.0" encoding="UTF-8"?>
<aiml version="1.0">
<category>
   <pattern>WHAT IS MY NAME?</pattern>
   <template>
      <condition name = "userName">
         <li value = "yes"> Your name is <get name = "
             userName"> </li>
         <li> Sorry, but I don't know your name yet </li>
      </condition>
   </template>
</category>
</aiml>
```

Multivalent Decision Element. Just like the Bivalent Decision element, the Multivalent Decision element is a conditional decision structure. However, the control variable can take from 3 to n distinct values, where $n \in N | n \geq 3$. Therefore, these values define the direction of the commands running flow.

Figure 5(a) shows the element that represents the multivalent decision commands. For each X control available values, depicted in the element's edges, there is a specific dialogue sequence. This element can be implemented using the AIML in two ways: using the <CONDITION> and <LI> tags or the <IF> and <ELSE> tags.

Figure 5(b) provides a DCD as an application of the Multivalent Decision element. The user in ① starts with the sentence "I LIKE ICE CREAM". This message is forward to the inference machine by ②, and later the decision command is applied. The inference machine uses the information found in the FAVORITEIC control variable to direct the dialogue flow.This variable stores the user's choice for favorite ice cream, and it can take the values: CHOCOLATE, STRAWBERRY or PINEAPPLE. With these values, the chatterbot replies as follows: (i) If FAVORITEIC is CHOCOLATE the chatterbot replies in ③ with "Chocolate ice cream is yummy"; (ii) If FAVORITEIC is STRAWBERRY the chatterbot replies in ④ with "I like strawberry ice cream too"; (iii) If FAVORITEIC is PINEAPPLE the chatterbot replies in ⑤ with "I would like a pineapple ice cream too".



(a) Multivalent Decision Element.

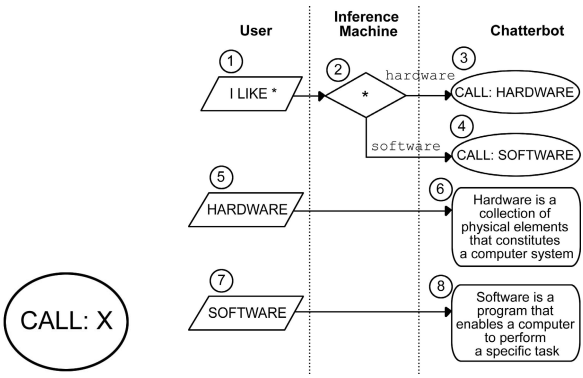
(b) Dialogue Conceptual Diagram with the Multivalent Decision Element.

Fig. 5. Multivalent Decision Element

By lack of space the AIML code related to this element and the followings will not be showed.

Category Call Element. The Category Call element directs the flow of distinct user inputs to a single chatterbot's reply. Thus, the bot identifies whether two or more user sentences are similar (with the same meaning), even when they are typed in different ways. It is also shown in Figure 6(a), where X is the pattern that defines the category to be called. When the category X is called into the context of another category Y (for instance), the category X's instructions are performed. When that processing of the category X is over, the chatterbot's running flow returns to the category Y, and the dialogue flow continues.

The DCD shown in Figure 6(b) in ① depicts the situation when the user types "I LIKE *", which the * wildcard marks any character sequence that the user typed. This DCD is ready to identify the following user inputs with the * wildcard: "I LIKE HARDWARE" and "I LIKE SOFTWARE". Therefore, if the user types any of these sentences, the inference machine directs the dialogue flow by the Bivalent Decision element defined in ②. It compares the value stored in * with HARDWARE and SOFTWARE values. Based on the outcome of such comparison, two dialogue flow directions are available: (i) If * is HARDWARE, the Category Call element calls the category with the "HARDWARE" pattern, as shown in ③. Thus, the chatterbot's reply related to ⑤ and ⑥ will be "Hardware is a collection of physical elements that constitutes a computer system"; (ii) If * is SOFTWARE, the Category Call element calls the category with the "SOFTWARE" pattern, as shown in ④. Based on such choice, the chatterbot reply is defined by ⑦ and ⑧ items as "Software is a program that enables a computer to perform a specific task".

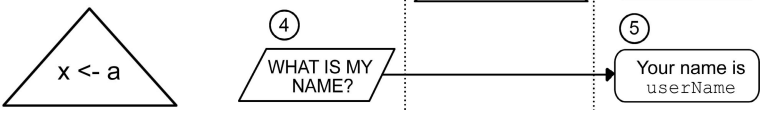


(a) Category Call Element.

(b) Dialogue Conceptual Diagram with the Category Call Element.

Fig. 6. Category Call Element

Variable and Temporary Files Storage Element. The Variable and Temporary Files Storage element depicts the files storage and/or variables that will be solely available to the chatterbot when the program is running. Then, if the program ends the files or variables are erased from the memory. This element is shown in Figure 7(a), where the x variable or temporary file has the "a" value. AIML uses the <SET> tag to perform the temporary variable storage.



(a) Variable and Temporary Files Storage Element.

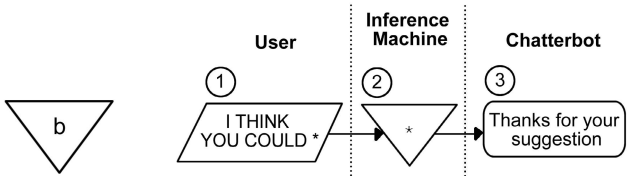
(b) Dialogue Conceptual Diagram with the Variable and Temporary Files Storage Element.

Fig. 7. Variable and Temporary Files Storage Element

Figure 7(b) shows an application of DCD's Variable and Temporary Files Storage Element. In ①, the user enters with the sentence "MY NAME IS *", where the * wildcard represents any character sequence typed by the user. In this particular instance, the goal is to grab the user's name with *. After that,

this message goes to the inference machine, as seen in ②, where the value found in * is stored in the USERNAME variable. In the next step ③, the chatterbot presents to the user the message "Nice to meet you". In the dialogue ④, the user asks "WHAT IS MY NAME?". To answer this question, the chatterbot uses the previously stored information found in the USERNAME variable, and replies to the user "Your name is USERNAME", as shown in ⑤.

Permanent File Storage Element. The Permanent File Storage Element allows the chatterbot's data persistence. Figure 8(a) shows this element details, and the "b" value is the information that the chatterbots needs to store.



(a) Permanent File Storage Element.

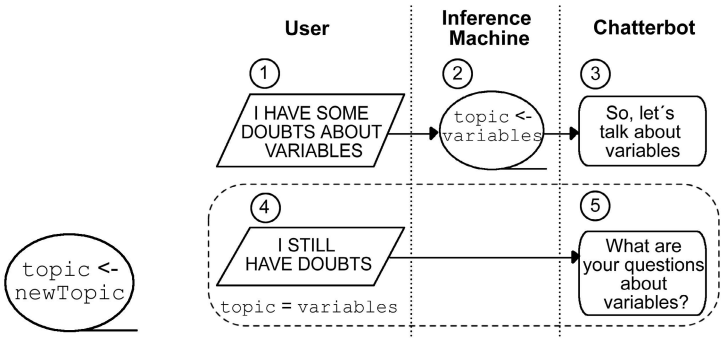
(b) Dialogue Conceptual Diagram with the Permanent File Storage Element.

Fig. 8. Permanent File Storage Element

The Permanent File Storage element is implemented using the AIML's <GOSSIP> tag. For example, Figure 8(b) shows a DCD that uses this element. The user starts to type in ① "I THINK YOU COULD*". The proposal for this message is to grab a user's suggestion to the chatterbot with the * wildcard. After that, the inference machine, through the Permanent File Storage element, saves the information found in *, as shown in ②. Finally, the chatterbot replies with the text "Thanks for your suggestion" (DCD Item ③).

Topic Change Element. A topic is a theme, a subject that the chatterbot is able to talk. For instance, a bot can be able to talk about topics/subjects like gastronomy, ioga, philosophy, and so on. A topic groups the categories that deal with the same subject, which allows the chatterbot to simulate an important feature of human dialogues: to direct the dialogue to a specific subject, to talk about this subject and later identify when there is a subject change during the dialogue (adding the ability for the bot to shift topics during chat).

In Figure 9(a), the Topic Change element is used when the subject is changed. It occurs when the TOPIC variable is set with the new subject, indicated by the NEWTOPIC value. Finally, it is describe in AIML with the <TOPIC> tag.



(a) Topic Change Element.

(b) Dialogue Conceptual Diagram with the Topic Change Element.

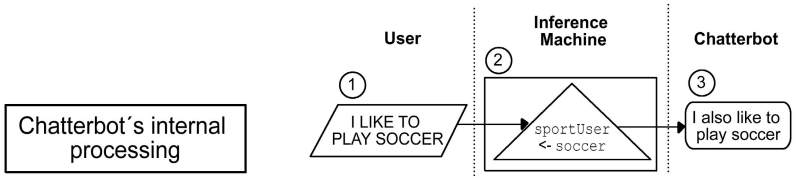
Fig. 9. Topic Change Element

Figure 9(b) shows an example of the VARIABLES topic definition that is used to group the categories related to the "variable" concept in the programming logic. In this figure, the topic and its categories are wrapped by dashed lines. In ① the user says "I HAVE SOME DOUBTS ABOUT VARIABLES". After that, in ② the inference machine sets the TOPIC variable with the VARIABLES value, pointing out the dialogue's theme change. From now on, the next user inputs detected by the chatterbot are sought initially in the VARIABLES topic category. Next, the chatterbot shows in ③ the sentence "So, let's talk about variables".

In the dialogue's following moment at ④, the user types the sentence "I STILL HAVE DOUBTS". Since the topic variable was set as TOPIC = VARIABLES, the chatterbot performs the pattern matching first in the VARIABLES' topic categories. Thus, the inference machine acknowledges that the user's intention is to talk about programming languages, and the chatterbot shows in ⑤ the reply with the question: "What are your questions about variables?".

Internal Processing Element. The Internal Processing element, depicted in Figure 10(a), represents the processing activities performed by the chatterbot that should not be seen by the users. In the AIML language there is a relationship between the Internal Processing element and the <THINK> tag.

Figure 10(b) shows an example of the Internal Processing element application. In this case, the chatterbot stores a certain user preference in a variable, but it does not display the data related to that action. At first, the user types the sentence "I LIKE TO PLAY SOCCER" in ①. After the recognition of that input pattern, the inference machine puts the SOCCER value into the SPORTUSER variable. Since this action is not supposed to be displayed to the user, the elements Variable and Temporary Files Storage is placed in the Internal Processing scope in ②. In ③, the chatterbot shows the sentence "I also like to play soccer".



(a) Internal Processing Element.

(b) Dialogue Conceptual Diagram with Internal Processing Element.

Fig. 10. Internal Processing Element

3 Dialogue Conceptual Diagram Graphically Displaying ALICE Chatterbot's Knowledge Base: A Case Study

In order to verify if DCDs are proper frames of reference for the graphical representation of chatterbots' dialogues, this section presents a DCDs usage's case study in the graphical representation of an ALICE [2] chatterbot KB's snippet. The ALICE chatterbot was chosen due to its acceptance as a model for chatterbots' developers. Also, it is an open source system with very specific implementation instructions and source codes. The code reuse allows the design of systems with no need to build them from scratch.

Table 1. Number of Occurrences for AIML Tags in the ALICE Chatterbot's KB

Tag	Occurrences	Tag	Occurrences
CATEGORY	93,607	THAT	1,542
PATTERN	93,607	CONDITION	49
TEMPLATE	93,607	TOPIC	10
SRAI	25,894	GET	8,265
SET	8,265	GOSSIP	0
THINK	3,071	IF	0
STAR	2,325	ELSE	0

The ALICE current version has 45.000 categories arranged in sixty-four (64) AIML files. Among these files it is possible to find the following: AI.AIML, ALICE.AIML, ASTROLOGY.AIML, GEOGRAPHY.AIML, POLITICS.AIML, RELIGION.AIML and WALLACE.AIML. ALICE KB's graphical representation using the DCDs is defined in two steps. First, a survey of how many times each AIML tag was used in the ALICE's KB was performed. Table 1 displays the outcome of this survey.

Table 1 gives an overview of which DCDs are the most common in the ALICE's KB. For instance, the Bivalent Decision and Multivalent Decision elements can be implemented with the CONDITION tag. Considering that this tag appears 49 times in the ALICE's KB, then this bot has dialogues that can be graphically represented by these elements. In a second step, all the 64 ALICE KB's related

Listing 1.2. AIML Code of the ALICE's KB Related to the Category Call Element.

```
<category>
<pattern>WHAT IS A CHAT ROBOT</pattern>
   <template>
      A chat robot is a program that attempts to simulate
          the conversation or "chat" of a human being. The
          Chat robot "Eliza" was a well-known early attempt
          at creating programs that could at least
          temporarily fool a real human being into thinking
          they were talking to another person.
   </template>
</category>
<category><
   pattern>WHAT IS A CHATTERBOT</pattern>
   <template>
      <srai>WHAT IS A CHAT ROBOT</srai>
   </template>
</category>
```

files were analyzed, while dialogues were chosen according to the tags shown in Table 1 and modeled with the DCD elements.

Hereafter the application of the Category Call element for the graphical representation of ALICE's KB is presented. Such element can be found in the file AI.AIML of the ALICE's KB, as shown in Listing 1.2. In this example, the user wants to ask what is the meaning of chatterbots. Figure 11 shows the DCD related to the code in the Listing 1.2.

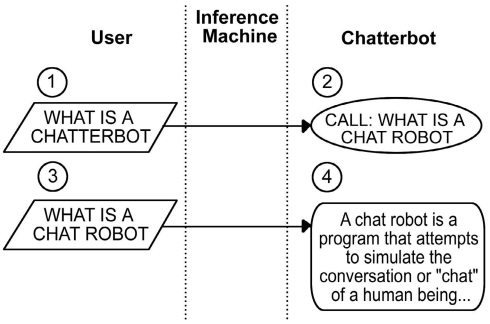


Fig. 11. Dialogue Conceptual Diagram Related to Listing 1.2

In ① the user enters the text "WHAT IS A CHATTERBOT". After that, the dialogue flow changes with the Category Call element for the category that corresponds to the pattern "WHAT IS A CHAT ROBOT" in ②. The category ③ has the chatterbot reply in ④.

4 Conclusions

The DCDs proposed by this work provide graphical common languages that can be used by domain specialists and computing researchers/developers with no extra knowledge. Domain experts' main concern is modeling the chatterbot's KB, while the programmers and computing experts handle the implementation details. Thanks to the DCDs, both parties can see the big picture and have a better control of the entire process using such graphical portrayal as a guideline.

In order to present the DCD as a tool and how it works, this paper started showing two related works and the AIML language with its main tags. Also, the correlation between dialogue's visual elements and AIML tags was considered. Another challenge this work surpassed was the presentation of a practical use case to see the usage of DCDs in an ordinary, real-life situation. A subset of the ALICE's KB was modeled using DCDs in order to portray how robust and viable they are as devices to graphically display bots' KBs.

The studies shown in this paper is not the end of the DCD, it is hunt the beginning. The purpose of this paper is to invite researchers, domains specialist from varied fields and artificial intelligence designers to try out a new, graphical and complete way of describing the knowledge for natural language processing. Thus, the next DCD's challenge will be the test of time and real-life applications.

References

1. Alcaniz, M., Rey, B.: New Technologies for Ambient Intelligence. In: Riva, G., Vatalaro, F., Davide, G., Alcaniz, M. (eds.) Ambient Intelligence: The Evolution of Technology, Communication and Cognition Towards the Future of Human-Computer Interaction, pp. 3–15 (2005)
2. Wallace, R.: The Elements of AIML Style. ALICE A.I. Foundation (2003)