Multi-Agent Systems in a Computational Environment of Education: A Chatterbot Case Study

André F. M. Batista, Maria G. B. Marietto, Gislene C. O. Barbosa, Robson S. França and Emerson A. Noronha
Federal University of ABC – Center of Mathematics, Computation and Cognition
Santo André, São Paulo - Brazil

Abstract

This paper proposes a dialogue based interface for virtual environments of learning with a Multi-agent approach. Here, it's shown both aspects involved with the motivation and conception of this interface project, the conceptual and the technological aspect. In the conceptual aspect, the concepts related to distance learning and learning process are presented in the context of interfaces for virtual environment for education. In the technological aspect, it's presented the multi-agent based systems, chatterbots systems and the AIML language applied in the context of the conceptual needs. Finally, based on these concepts and technologies, the paper's contribution is a project for a virtual environment of learning, a middleware architecture based on a multi-agent approach.

1. Introduction

The ability to apply computers in the teaching-learning process by creating educational virtual environments provides interactive situations which the student may arise as an active player in his knowledge construction. Lévy [13] highlights the impact of computers in the individuals' cognitive skills by the interface features, the software itself and its tools. Considering the graphic interface alone, the most part of the interface is based on hypertext navigation systems and/or taking choices by buttons and menus. Both approaches demand an additional cognitive effort by students because the usual form of communication among humans is the natural language. Therefore, the most suitable cognitive interfaces are those which allow the establishment of dialogs in natural language. This makes the interaction student-system more appealing, and with a higher level of empathy. Among others, the software that has this interface feature is a chatterbot system, and its main goal is to get acquainted with people [8].

In this context, this paper goes on investigating the usage of chatterbots as an interface for education in a virtual environment. Specifically, the application of a chatterbot as an interface for clarifying students' doubts about the Java programming language. This interface based on a chatterbot system will be a supporting tool for students in the content of their classes.

The usage of virtual environments in education can contribute in the process of teaching-learning because: (i) it has a more suitable illustrational power on some process and objects, comparing to other types of media; (ii) it allows the student to develop the work at his pace; (iii) it demands more interaction, encouraging the student for a participative role in the learning process. However, without the interpersonal relationship with teachers and classmates in this kind of environment, the student tends to fell discouraged by the artificiality of the human-machine interaction, resulting in loss of interest of the educational process [17].

The application of chatterbots as virtual assistants to simulate a dialog with partners can stimulate students, decreasing their feeling of isolation in these environments. Additionally, these kind of systems provide a feeling of security and satisfaction because they are always accessible, providing to the student an always available virtual tutor to chat and/or to settle his doubts. The following works are examples of chatterbots applied in education: Júnior [13;14], Electra [9], BonoBOT [15].

This paper is organized as follows. Initially, Section 2 presents a general view of multi-agent based systems. Next, in Section 3, the concepts related to chatterbots are presented, along with a description of the AIML language, used in chatterbots (Section 3.1). After that, Section 4 presents the concepts and motivations related to the usage of chatterbots in distance learning and it is followed by the requirements and the proposed software architecture. Finally, Section 4.2 presents the construction of the knowledge base, and this project's mail goal is later illustrated in Section 4.3 in a practical manner.

2. Multi-agent Based Systems

The Distributed Artificial Intelligence (DAI) has emerged from the integration of the Artificial Intelligence (AI) area and Distributed Systems (DS) in the 80s, with theoretical and practical implications from both areas. With cognitive and social theories the DAI area aims to present newer and more comprehensive forms of problem solving, programming languages, planning and knowledge representation. The agents defined in the DAI area allow a rich metaphor of live organisms and its relationship patterns. In that sense, the DAI paradigm allows to solve problems in a cooperative and distributed way, using process (programs) called agents [1; 2].

The DAI approach is justified for several reasons such as, for instance, the existence of domain problems that are distributed in space by nature [3]. The DAI's theories and technologies have been used in a great variety of areas such as industrial and educational systems, entertainment applications, etc. The Multi-agent Based Systems is one research area of DAI and, in the development of agents in MABS, the designer does not take the problem or even the structure in which the agent is inserted into consideration. Instead, the designer makes each one of the agents able to develop actions, in its specialties, according to a given set of knowledge and skills. This way, the agents are independent from the problem and society, which make them able to solve problems that were not considered in its development in the first place [4; 5].

Some of the main concepts in an agent-oriented programming are: (i) the social organization, which is constructed in the base of the communication and collaboration among process involving agent-agent and agent-user; (ii) the emergence, which unanticipated results can be analyzed in a more objective way; (iii) the agent’s autonomy, which the agent can make a decision on its own actions and behavior. Based on these three aspects, it's possible to create a middleware layer to provide support for computational environments. Thus, this paper proposes the development of a multi-agent prototype, considering it as basic building blocks which services can be offered to the educational environment.

These services act as a middleware layer, covering all components of the environment and making the cooperation, communication and coordination easier among its parts. The agent platform JADE (Java Agent Development Framework), along with other tools, will be used in the formation of this layer. According to [6], this platform has proven to be robust and presents a reasonable scalability, which allows the development of applications with a great number of agents with support to a fault-tolerance mechanism, which is very important in the development of distributed applications.

3. Chatterbots

The idea of a chatterbot came in the 50s, when Alan Turing brought the following question: can machines think? In his article “Computing Machinery and Intelligence” [18], Turing proposed a game called Imitation Game, now called Turing Test. The purpose of this game was confusing the interrogator which tries to discover if he is talking to a human being or a machine.

Bot is an English simplification for the word robot, which is an agent which cooperates with a user or another program, simulating a human activity. The bots can be classified in many categories, as academics, searches, trades, etc; this paper deals with chatterbots or conversation robots. The goal of this kind of bot is to answer questions, in a way that people could think they are talking to another person, instead of a computer program. The chatterbots have inside its knowledge base a set of simulated dialogs, to communicate with users in a natural language, and can be used as interfaces in a wide range of applications like electronic commerce, distance learning, among others. The chatterbots use Artificial Intelligence to simulate a dialog with a human being based in a “stimulus – response” routine: you make questions and he provides answers based on those questions. After a question is submitted in a natural language, the program queries a knowledge base and sends an answer trying to mimic the human behavior.

The first generation of chatterbots started with ELIZA, developed by the Joseph Weizenbaum Professor in 1966, from the Institute of Massachusetts Technology [19] At the time, small databases were used for this kind of program and there were not specific programming languages and appropriated models for bot programs. In its second generation, the chatterbots started using Artificial Intelligence’s new technologies and the outstanding project of that generation was Julia. Apart from having a knowledge base for a certain subject, Julia had the capability of learning during the dialogs, increasing the felling of intelligence about the software [17]. The third generation began marked by bots that use a specific technology developed for conversation, based on the Extensible Markup Language (XML). The forerunner of this generation is the ALICE (Artificial Linguistic Internet Computer Entity) [1]. This bot was implemented with a supervised learning model which the role of “botmaster” (a person who manages the chatterbot's knowledge base) is fundamental. The botmaster analyzes the dialog, identifies needed improvements and creates new content or “knowledge” in the form of files based on the Artificial Intelligence Markup Language (AIML), so that subsequent responses are more appropriate.

The works [11; 14] mention the virtual learning, the foci of this work, as one area that takes advantages from the usage of chatterbots. First, the bots in this area can be used to replace long FAQs (Frequently Asked Questions) pages, and also they can be used as a helping tool in the students' learning process. In the first case, the students can interact with a bot to get the desired information, instead of a visual search in one long FAQs page. In the second usage, the bot technology can help the students in learning a certain concept inside a topic, using a thematic chatterbot. In both cases, a bot working twenty-four hours a day will be always available to answer and help the students in their most diverse questions. However, if the bot does not have an answer, it can operate in one of these ways, according to the situation: (i) it can request to the student to send a message for a teacher or to a staff member that will send an asynchronous answer; (ii) the bot can directly send to a teacher or staff member the students' question which, in turn, will receive an asynchronous answer as well.

By assisting in the learning process, speaking to a bot works as a stimulus for the student in his learning activity. This happens because this "chat" puts the student into a more active line of work, instead of a simple read of a long text or just a choice-and-click on underlined keywords in a page. The integration with a bot in an interactive virtual environment of learning can be enhanced by chats with other students and learning tools like animations, movies, web sites and sounds.

It's worth emphasizing that the efficiency of the chatterbot application in teaching is directly related to the quality and extent of the knowledge base available to it. That is, the chance of the student's success is related to the quantity of keywords recognized by the bot in the topic and the base's logic consistency. The capacity of the base in proposing friendly answers to the students is also a key factor in the successful virtual teaching activities. In the case of a poorly sized and/or a low logic consistency knowledge base, generating evasive answers in the first case and answers with no logical sense in the second, the student will lose interest, putting himself against the tutor system.

3.1. The AIML Language

Nowadays, Alice is one of the best known open source (in the terms of General Public License) chatterbots. This bot has a knowledge base composed by a vocabulary with about 5,000 words [10]. The Alice's brain is composed by a knowledge base files in AIML, a markup language that has a simple structure by a computational point of view. This language is maintained by the A. L. I. C. E. Artificial Intelligence Foundation that keeps the free distribution of the program.

AIML is based on the eXtensible Markup Language (XML), it's easy to learn and use. This language represents a set of tags, or elements, and simple commands that serve to analyze the messages sent by the user and to decide the form that this messages must be answered [5]. The tags are delimited by < e > and its syntax consists of an opening (< ... >) and of an ending tag (</ ...>), as following:

```
<some-tag>
       ... pieces of information ...
</some-tag>
```

When there's no information between the opening and closing tag an abbreviation - consisting of the name of the tag and a slash in the end of it - can be used as follows [5]:

```
<some-tag/>
```

The input patterns from the user are known as categories and are identified by the element <category>. Each category is composed by an input pattern identified by the element <pattern>, associated with one or more models of answer identified by the element <template>. When the user types a question, the bot makes a search to verify if this information is in his knowledge base. When the information is found, the bot sends the response delimited between the tags [5]. Besides of realizing the matching of patterns, the AIML language brings a set of tags that distinguish it from the others because it has features such as memory and the capability to contextualize and revalidate sentences typed by the user. So, the following section details the most common tags.

3.1.1. Tags

Ever since the invention of the printing press, writers have made notes on manuscripts to instruct the printers on matters such as typesetting and other production issues, these notes were called "markup", a collection of such notes that conform to a defined syntax and grammar can certainly be called a "language" [4]. This is the background of the SGML (Standard Generalized Markup Language), the markup language from which XML (Extensible Markup Language) is derived.

AIML is an XML-compliant language and all based markup languages use literal strings of characters, called tags to delimit the major components of the meta data, called elements. And element is everything from the element's start tag to the element's end tag. AIML is composed by defined tags that specifies predicates and, besides the defined tags, this language allows the construction of custom predicates. The most important defined tags in AIML are now presented.

(a)AIML

The tag <aiml> delimits an .aiml file. And inside of that tag must be at least one <category> element. The following box shows a model of its syntax.

```
<aiml version="1.0">
  <category>
    <pattern> WHICH ARE THE PRIMITIVE TYPES
OF CHAVA? </pattern>
    <template> The mains primitive types of
Java are boolean, char, byte, short, int,
long, float, Double. Do you like to know
about a specific one:
    </template>
  </category>
</aiml>
```

(b) CATEGORY

In a knowledge base, the words in a sentence can be visualized as a series of links and nodes. The links are labeled by the words and form a chain starting from the center of the knowledge base and moving outwards, forming a kind of a chain of words that generates the response to the user. So, the specific path through the Knowledge Base along with the Response Template is called a Category.

The basic top tag in this chain is the <category> tag, that can be directly inside of the <aiml> or <topic> tag . Each category tag is composed by an input question, an output question and an optional context. First, the input questions are delimited by the <pattern> tag. Next, the output questions are answers provided by the bot to the user and are contained in the <template> tag. Then, the context is an optional tag indicated by the use of the <topic> and <that> tag [1]. The <category> tag has three specified elements: (i) pattern, (ii) template and (iii) srai, which are detailed as follows:

(i) PATTERN

The <pattern> is a category's tag element and has the possible variations in the input of the users. According to the specification, the <pattern> must be the first element inside a <category> tag and it's allowed only one element of <pattern> per <category>. Besides that, this element must be defined by capital letters.

(ii) TEMPLATE

The <template> is a category's tag element and has the possible variations of the answers of the bot to the users. This element has the most part of the information about the bot, which can be only a phrase or an entire text. This element can be used to transform the answer in a little program, for data saving and activating another programs, or even to give conditional answers and calling answers from others categories.

(iii) SRAI

The <srai> element is used by the programmer to link pattern elements with each other, which is considered a great functionality of the AIML language. In this manner, the interpreter of the language is able to find out an answer in a big set of possibilities in a fast way, according to the users input. This <srai> element can be used to make (a) symbolic reduction, (b) to handling synonyms, (c) to grammatical corrections or (d) to detect key words. The following has a breakdown of each item:

(a) Symbolic reduction: refers to the transformation of complex grammatical forms into more simple forms. In the following illustration code, when the user inputs the question "what is a vector" or "do you know what is a vector?", the interpreter of the AIML language redirects the questions to a <pattern> element, to reduce this question to "vector":

```
<category>
 <pattern> VETOR </pattern>
 <template>
       Vectors are efficient forms to
organize data in rows
 </template>
</category>

<category>
 <pattern> What is a vector? </pattern>
 <template>
        <srai> VETOR </srai>
 </template>
</category>

<category>
 <pattern> Do you know what is a vector?
</pattern>
 <template>
       <srai> VETOR </srai>
 </template>
</category>
```

(b) Handling synonyms: the categories that have synonyms can be redirected to an only answer in a recursive declaration form. The following illustration code shows the synonym "array" of the vector word:

```
<category>
 <pattern> VETOR </pattern>
 <template>
       Vectors are efficient forms to
organize data in rows
 </template>
</category>

<category>
 <pattern> ARRAY </pattern>
 <template>
        <srai> VETOR </srai>
 </template>
</category>

<category>
 <pattern> DATA STRING </pattern>
 <template>
        <srai> VETOR </srai>
 </template>
</category>
```

(c) Grammatical corrections: the programmer must prevent grammatical errors from the user and treat these errors, replacing it with the correct spelling, redirecting the text for the correct <pattern> element. The following code illustrates some possible writing mistakes for the word "vector", linking these writing mistakes with the correct word "vector" specified by a <pattern> element:

```
 <category>
 <pattern> VETOR </pattern>
 <template>
       Vectors   are   efficient   forms   to
organize data in rows
 </template>
</category>

<category>
 <pattern> VECTOR </pattern>
 <template>
        <srai> VETOR </srai>
 </template>
</category>

<category>
 <pattern> VETO </pattern>
 <template>
       <srai> VETOR </srai>
 </template>
</category>
```

(d) Detecting key words: in AIML it's common to use answers which are activated with the some key word inside an expression. To link a pattern with a set of key words the AIML has the wildcards "*" or "_". In the following code, it does not matter the word "vector" comes alone, after a sentence, with prefix or suffix, the answer will be the same:

```
<category>
 <pattern> VETOR </pattern>
 <template> Vectors are efficient forms to
organize data in rows </template>
</category>

<category>
 <pattern> VETOR </pattern>
 <template>
 <srai> VETOR </srai>
 </template>
</category>

<category>
 <pattern> *VETOR </pattern>
 <template>
 <srai> VETOR </srai>
 </template>
</category>

<category>
 <pattern> _VETOR </pattern>
 <template>
 <srai> VETOR </srai>
 </template>
</category>

<category>
 <pattern> VETOR_ </pattern>
 <template>
 <srai> VETOR </srai>
 </template>
</category>
```

(f) Other tags

This section presents other important tags specified by the AIML Language, according to the specification presented in [1].

Tag	**Description**
	Specification / Illustration Code
<set>	Allows storing variables in the chatterbot memory. The set element instructs the AIML interpreter to set the value of a predicate to the result of processing the contents of the set element. The set element has a required attribute name, which must be a valid AIML predicate name. If the predicate has not yet been defined, the AIML interpreter should define it in memory. In the following column there is the <gossip> specification.

	`<!-- Category: aiml-template-elements -->`
	`<aiml:set name = ` *`aiml-predicate-name`* `>`
	
	`<!--Contents: ` *`aiml-template-elements`* ` -->`
	
	`</aiml:set>`

<condition>	The tag allows the definition of roles from the stored variables. In the AIML description, the interpreter has stored the "Java" (line 3) for the variable "language" (line 1). So, in line 4, the return value for this dialog is "object oriented paradigm".

	`<category>`
	`  <pattern>I like </pattern>`
	`  <template> You should study`
	`   <condition>`
	`     <li name="language"`
	`         value="java">`
	`       objected oriented paradigm.`
	`      </li>`
	`      <li name="language"`
	`         value="C">`
	`         imperative paradigm.`
	`      </li>`
	`    </condition>`
	`  </template>`
	`</category>`
	`---`
	
	EXAMPLE:
	1. USER: I'm studying programming languages.
	2. BOT: Computer languages are interesting.
	3. USER:I like Java.
	4. BOT: You should study object oriented paradigm.

<system>	The language AIML also supports interface with other languages. This tag runs and put in the response an accessible program. For example, a command from the operating system.	`<category>`
		`   <pattern>`
		`      What is the JVM version?`
		`   </pattern>`
		`   <template>`
		`      The JVM version is`
		`      <system>`
		`         java -version`
		`      </system>`
		`   </template>`
		`</category>`

<gossip>	Gossip is a content-capturing element and must have been inside the <template> tag. The gossip element instructs the AIML interpreter to capture the result of processing the contents of the gossip elements and to store these contents in a manner left up to the implementation. In the following column there is the <gossip> specification.	`<!-- Category: aiml-template-elements →`
		
		`  <aiml:gossip>`
		
		`     <!--Contents: ` ***`aiml-template elements`*** ` -->`
		
		`  </aiml:gossip>`

<javascript>	Allows the execution of scripts inside the responses [Fre07]. The javascript element instructs the AIML interpreter to pass its content (with any appropriate preprocessing, as noted above) to a server-side JavaScript interpreter on the local machine on which the AIML interpreter is running. The javascript element does not have any attributes.	`<!-- Category: aiml-template-elements -->`
		
		` <aiml:javascript>`
		
		`    <!-- Contents: ` ***`character data, aiml-template-elements`*** ` -->`
		` </aiml:javascript>`

<formal>	Change, for a capital letter, the first letter of each word in a sentence. If the character is not in the Unicode pattern, the original string is returned.	`<!-- Category: aiml-template-elements →`
		
		`  <aiml:formal>`
		
		`      <!-- Content: ` ***`aiml-template-elements`*** ` -->`
		
		`  </aiml:formal>`

<uppercase>/<lowercase>	Change all the letters to capital or lowercase. If the character is not in the Unicode pattern, the original string is returned.	`<!-- Category: aiml-template-elements →`
		
		` <aiml:uppercase>`
		
		`    <!-- Contents: ` ***`aiml-template-elements`*** ` -->`
		
		` </aiml:uppercase>`

<sentence>	Change the first letter of a sentence to capital. If the character is not in the Unicode pattern, the original string is returned.	`<!-- Category: aiml-template-elements -->`
		
		`  <aiml:sentence>`
		
		`    <!-- Content: ` ***`aiml-template-elements`*** ` -->`
		
		`  </aiml:sentence>`

<random>	Allows the bot to answer the same question with a set of responses. Each response is declared in the <li> element.	`<category>`
		`  <pattern>My name is _`

	`</pattern> <template> <random> <li> Hi </li> <li> How are you </li> <li> Hello </li> </random> <set name="name"> <star /> </set> </template> </category> ----` **EXAMPLE:** 1. USER: My name is Gislene 2. BOT: Hi Gislene.
<get>	This tag must be inside the <template> tag, and returns a value stored by the <set> tag. The get element tells the AIML interpreter that it should substitute the contents of a predicate, if that predicate has a value defined. If the predicate has no value defined, the AIML interpreter should substitute the empty string "".
	`SPECIFICATION: <!-- Category: aiml-template-elements --> <aiml:get name =` ***`aiml-predicate-name`*** `/>` `CODE: <category> <pattern>Good Night!</pattern> <template> Good Night <get name="name" /> </template> </category> ----` **CONTINUING THE ABOVE EXAMPLE:** 3. USER: Good Night! 4. BOT: Good Night Gislene
<that>	The pattern-side that element is a special type of pattern element used for context matching. The pattern-side that is optional in a category, but if it occurs it must occur no more than once, and must directly follow the pattern and directly precede the template. A pattern-side that element contains a simple pattern expression. The pattern-side that tells the AIML interpreter that its category is only a valid match result if the matches a previous bot output.
	`<!-- Category: aiml-category-elements --> <aiml:that index = (`*`single-integer-index`*` | `*`comma-separated-integer-pair`*`)>` `<!-- Content:` ***`aiml-pattern-expression`*** `--> </aiml:that>`

4. Chatterbots and Distance Learning

Distance learning is the teaching-learning processes mediated by technologies, where the professors and students are separated spatially and/or temporally[7]. According to [2], for distance learning a media technology can be used in two ways, or to simulate the classroom education or to explore new possibilities of learning through the exploration of intrinsic characteristics of the used technology. The purpose of this proposed project is to explore the new possibilities, with a learning tool that helps students in their individual studies. The main characteristic of the distance learning is the establishment of a dual canal of communication, and this project understand that a dialog based interface is an important technology to obtain this canal of communication with a software that presents static content.

The advent of information technology and communication (ITC) revived the practices of distance learning due to the flexibility of time, breaking spatial barriers, and receive instant issuance of materials. This allows exploring both the mechanist forms of knowledge transmission by hypermedia and the interactivity potential of ITCs technologies, developing activities in distance, in the basis of the interaction, for knowledge production [2]. It is in this contemporary setting of distance learning that this paper proposes a tutor software interface, based on dialog interaction, for the teaching of programming languages.

The possibility of providing a high availability service for students of computing, without space/time barriers, is one of the main factors of motivation of this paper. It's notable the importance of this kind of learning tool when we consider the physical distances, mainly in a big country like Brazil. The State has the obligation to offers education to their citizens, which is a hard thing to a country with continental dimension, to offer properly

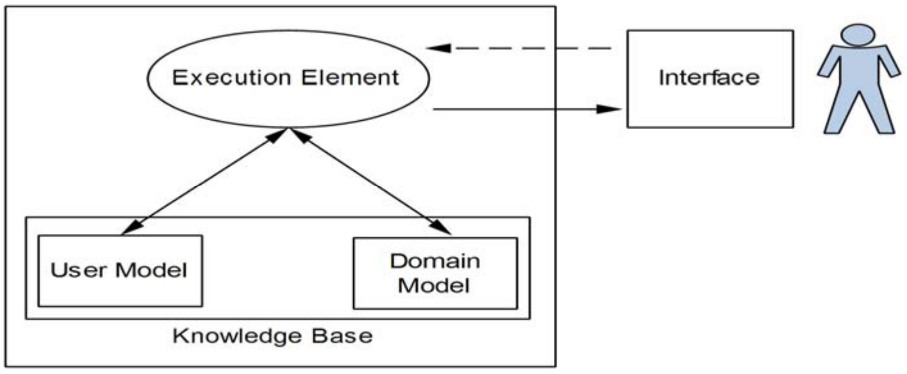

Figure 1. Generic Architecture of a Chatterbot

education to all cities and communities. Virtual environment learning tools makes knowledge available to anyone with a point access to the web. Therefore, researches to offer good mechanisms for this kind of learning environment have a great importance.

Another fact is, in an empirical manner, it's possible to affirm that the students that begin their studies in programming languages usually have difficulties in the learning of the content related with this disciplines. It is expected that the students related with the computer area have a quickly and positive adaptation with an environment of virtual tutor. This proposed virtual learning environment aims to serve as a complement, an additional tool, for supporting the learning process of the students. Through this virtual environment the students can count on a tool that complements their individual studies, a tool that uses the potential of the cognitive process related to the natural language in the learning process. It's exactly by the cognitive aspect that an environment based on a tutor software can facilitate the process of learning.

A virtual environment teaching based on dialogs by natural language puts the student in an active position, working and activating the cognitive process related with both, the visual perception and with the speech. In contrast with a conventional graph interface, that holds the student in a static position, a mere asynchronous receiver of information. Thereby, is this cognitive advantage that puts the proposed dialog based interface as a good tool when it's compared with the traditional softwares interfaces. Thereby, in the context of Distance Learning, is this cognitive advantage that puts the proposed interface, based in dialog, as a better complement for softwares with only traditional interfaces. Another aspect that a dialog based interface has advantage is the fact that it take off the students' feeling of isolation when they are using a virtual learning environment.

4.1. Requirements and the Architecture of the Implemented System

The Ask4Java is capable to promote a dialog in the context of the Java language. With the intent to facilitate the access of the students, the browser was chose as interface of this project. The AIML interpreter used was the Chatterbean, a free software which gives support to the use of the knowledge base in AIML. According to [7], its architecture is formed in the following manner:

(a) User Model: has information about the users of the system with the purpose to provide a more natural dialog with the machine;
(b) Domain Model: contains the information of the subject dominated by the program;
(c) Interface: the way by which happens the interaction among the user and the bot. In this interface, the user makes the request and the bot sends an response;
(d) Element of Execution: component that process the user request and searches for the more assertive option for an answer.

The Figure 1 shows the architecture above used. Based on this architecture, this work proposes the construction of a multi-agent chatterbot, where the JADE multi-agent platform acts as a middleware among the user and the bots. A prototype based on this architecture has been developed to experience the structure of an chatterbot agent. This prototype helps the user with questions about the Java language concepts.

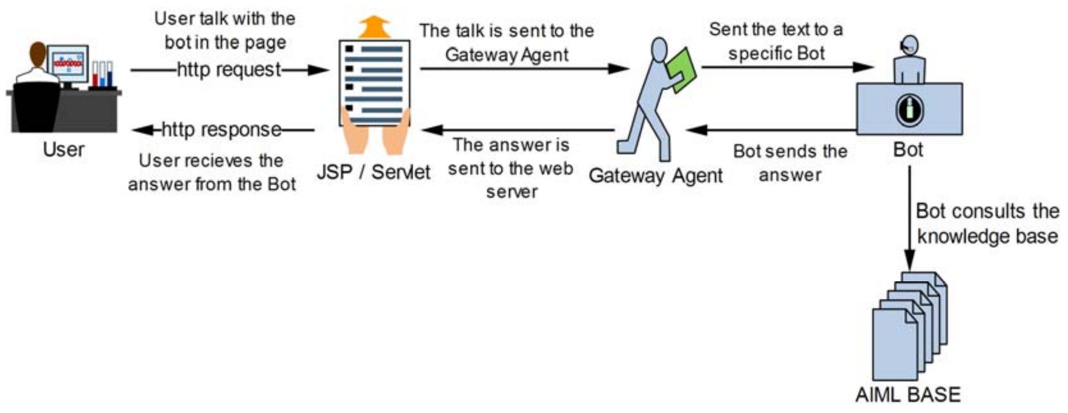
User talk with the bot in the page
http request
User
http response
User recieves the answer from the Bot
JSP / Servlet
The talk is sent to the Gateway Agent
The answer is sent to the web server
Gateway Agent
Sent the text to a specific Bot
Bot sends the answer
Bot
Bot consults the knowledge base
AIML BASE

Figure 2. The Scheme of the proposed Chatterbot

This proposed arquitecture will provide an agent (GatewatAgent) that will communicate both with the servlet and with the others agents in the platform (bots agents). The gateway agent activation and finalization is made automatically by the platform, in the way that the developer do not need to concern with it. The servlet technology make more easy to add functionalities in the web server to attend and process http requests.

The Figure 2 shows the process of a “talk” among the a user and the chatterbot. When the user sends his question by the browser, this arrives at the servlet in a webserver. The webserver interprets its contents and then sends it to the gateway agent, which is responsible for forward that to the chatterbot in JADE platform. Thereby, with this architecture, will be possible to create specialized gateways agents that, in turn, will be able to handle the request according to specific subjects.

4.2. Building the Knowledge Base

Before the construction of a knowledge base itself, it's necessary to make a survey in the subject of the knowledge, the Java language in this case. For this area, a departure of the concepts makes easier the construction of the base. James Gosling organizes the language concepts in a crescent manner, from fundamental concepts to the more complex one. This division is as follow:

Categories of Concepts	Dependent Concepts
Language	Key words, Comments, Javadoc, Main, Garbage, Collection, Variables and Initialization, Modifiers and Jar.
Wrapper Classes	Boolean, Int, Short, Char, Float, Double, Long, String, Date and Calendar.
Operators and Assignments	Assignments, Unary Operators, Arithmetic Operators, Shift Operators, Comparison Operators
Flow Controls and Exceptions	Loop Constructs, Selection Statements, Exceptions
Objects and Classes	Objects, Class Definition, Constructors, Methods and Operations, Overloading and Overriding
Collections	Array, Iterator, LinkedList, HashMap, TreeMap, Sort, Stack, Deque, Vector
Threads	Thread Fundamentals, Controlling Threads, Monitors

Each of these concepts has a great amount of other dependent concepts and, in a set, they form the knowledge base. To illustrate this association of concepts in the process construction of the knowledge base, it's used the survey about the vector concept. First of all, a conceptual map of the vector concept has been made (See Figure 3).

The conceptual map in the Figure 1 shows all the dependent concepts of the vector concept. That is, all the necessary concepts to a student fully understand the vector concept in Java. In the context of this map, the student must comprehend the following concepts that are related with vector: declaration, initialization, primitive types, flow control, key words and construction. Only after a fully understand all of these concepts the student will be able to study the vector concept, applying these dependent concepts on that one.

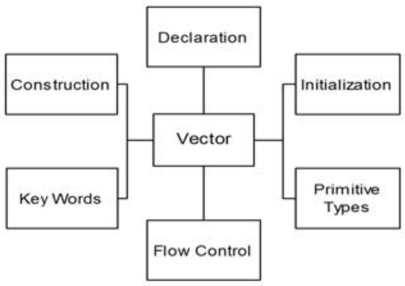

Figure 3. The Conceptual Map of the Vector

The direct information that vectors are efficient forms to organize the data in strings is not sufficient to remove the students' doubts. The source code may provoke only the passive reaction of code repetition by the student that will reproduce a code without a conceptual understanding of what he is doing. This fully comprehension is important to the development of both the student and the science computer itself.
In this manner, in the basis of the students' questions the tutor program can evaluate which concepts the student needs to know to get a real comprehension about the vector concept. Thereby, when the tutor program is asked about the vector concept, it returns the specific answer using the necessary dependent concepts. If the student doesn't know about the concepts, he will return more questions and the program will discover, in this process, the dependent concepts that the student needs to understand. Inserting the student in an interactive and incremental process of conceptual discoveries.

4.3 A Scenario Dialog to Illustrate.

This Section presents a dialog among a student and the chatterbot about the concept of Array in the Java Language. Here, the objective is illustrating the final intent of this virtual learning environment. Follows the dialog:

STUDENT: Hello.
BOT: Good Night! What is your name?
STUDENT: Robert
BOT: Hello Robert.
STUDENT: I have some doubts about Java
BOT: Java is what I Know, and what you going to learn very well!
STUDENT: I'm not sure about the vector concept.
BOT: A Java Array is an ordered collection of primitives, object references, or other arrays. Java Arrays are homogeneous, all elements in a array must be of the same type.
STUDENT:Wow!
BOT: Learn Java is a great thing, I enjoy very much to talk about it! Doesn't you Robert?
STUDENT: YES. I know about primitives and its relation with an array.
BOT: Oh Yeah!
STUDENT: what about this relation of an array and object references?
BOT: In Java an array is capable of keep references to objects.
STUDENT: Why an array keep references to objects?
BOT: The programmer can organize a collection of objects in an array, such as it can do with primitive types.
STUDENT: But is an vector not capable to store the objects itself?
BOT: All the objects in Java are stored in the JVM's Heap memory, the variables keep a reference to the objects.
STUDENT: Does the variables not stores the objects itself?
BOT: Instance variables has references to objects that are in the Heap memory. And variables of primitive data types stores the data itself.
STUDENT: The variables does not store the objects itself?
BOT: The variables keep references, memory address, for objects that are in the JVM's Heap memory. Make this exercise: print in the screen an Instance Variable.
STUDENT: OK, I'll do it.
BOT: Great! Keep walking!

5. Final Considerations

This work presented an overview of the construction of chatterbots, addressing elements that constitute its architecture and the AIML Language.

Additionally, we proposed an architecture of a multi-agent system that uses the JADE agent platform as an intermediate layer, allowing applications to be included in computing environments. More specifically, the prototype was directed to the construction of a chatterbot aimed to be an interface for a virtual learning environment about the Java Language.

However, the modeling of the chatterbot is generic, allowing the robot to deal with different areas, assisting in various applications such as distance education. For example, chatterbots who talk with students about what occurred in the environment since their last visit, on the course schedule, etc.

This proposal was based on the fact that the dialogue with a chatterbot tends to become more attractive to students than simply reading a text, this calls for active participation in the communication process. It`s possible to build new chatterbots, so that knowledge becomes interdisciplinary and questions that had not been shaped by developers, can be answered by bots in the future.

The JADE platform, along with all the tools used in the construction of the prototype implemented, provided a middleware layer that need not necessarily be used to chatterbots. With this layer we can take advantage of the dynamic and multi-agent distribution inherent in the Internet to manage networked computing environments, among other applications. However, the Portuguese language is very complex, and a multi-agent middleware can be capable to make evaluations about the bots according to a reputation scheme. Choosing, then, the more capable bot according to type of question.

6. References

[1] A.L.I.C.E. AI Foundation Working Draft - (rev 005), available at http://www.alicebot.org/TR/2001/WD-aiml-1.0.1-20011018-005.html#section-pattern-side-that, accessed in 02/26/2010.

[2] Almeida, M.E.B de., Educação a distância na internet: abordagens e contribuições dos ambientes digitais de aprendizagem, Educação e Pesquisa, São Paulo-Brasil, vol.29, n.2, pp. 327-340.

[3] Banach, Z. Construimos nuestro propio bot. Software 2.0 Extra!, n. 1, 2004. Avaliable at http://www.software20.org/es/attachments/SI aiml ES-.pdf, accessed in 12/20/2005.

[4] Birbeck, M. et al - Professional XML. Wrox Press, 2nd edition, Jan., 2000.

[5] Dutra, R.L. de S., Tarouco, L.M.R., Leonhardt, M.D. and Castro, D.D.de C., Elektra: Um chatterbot para uso em ambiente educacional, available at http://www.cinted.ufrgs.br/renote/set2003/artigos/, accessed in 02/20/2010.

[6] Freese, E., Enhancing aiml bots using semantic web technologies - Extreme Markup Languages, available at http://www.dm.ufscar.br/~waldeck/curso/java/part22.html, accessed in 02/27/2010.

[7] Laureano, E. A. G. C. ConsultBot - Um Chatterbot Consultor para Ambientes Virtuais de Estudo na Internet. Trabalho de Graduação. Universidade Federal de Pernambuco, 2002.

[8] Leonhardt, M. D. "ELEKTRA: Um Chatterbot para Uso em Ambiente Educacional". Novas Tecnologias na Educação, Vol.1, No2, 1-11. 2003.

[9] Levy, P.. "As Tecnologias da Inteligência: o Futuro do Pensamento na Era da Informática". Rio de janeiro: Ed. 34, 1993.

[10] Barretos, L, Amaral, V: Chatterbots e suas aplicações na educação, Universidade Federal da Bahia (UFBA), Bahia, Brasil, 2006.

[11] Lima, A.B. "E.life-Robôs que falam". Avaliable at http://informatica.terra.com.br, accessed in 04/25/2008.

[12] Moram, J.M. O que é educação a distância, Ministério da Educação: Secretaria de Educação a Distância, Brasília-Brasil,http://www.eca.usp.br/prof/moran/dist.htm, accessed in 02/26/2010.

[13] Primo, A. F. T., et al. "O Uso de Chatterbots na Educação a Distância". Porto Alegre: UFRGS, 2000. http://www.nied.unicamp.br/oea/mat/chatterbots_lec.pdf. Accessed in 03/03/2004.

[14] Primo, A. F. T., et al. "Júnior, um Chatterbot para Educação a Distância". 2000b. Avaliable at http://www.c5.cl/ieinvestiga/actas/ribie2000/papers/255/, accessed in 06/19/2004.

[15] Sganderla, R., Ferrari, D. N., Geyer, C. F. R. "BonoBOT: um Chatterbot para Interação com o Usuários em um Sistema Tutor Inteligente". In: SBIE´03. Anais. Rio de Janeiro: UFRJ 24, 2003. p.463-473.

[16] Silva, A. B.; Um Chatterbot em AIML Plus que conversa sobre horóscopo. Recife, PE, Brasil, 2002.

[17] Teixeira, S., Menezes, C. S. "Facilitando o uso de Ambientes Virtuais através de Agentes de conversação". In: SBIE´03. Anais... Rio de Janeiro: UFRJ 24, 2003.

[18] Turing, A.M. (1950). Computing machinery and intelligence. Mind, 59, 433-460. http://www.loebner.net/Prizef/TuringArticle.html, accessed in 02/28/2010.

[19] Weizenbaum, J. ELIZA - A Computer Program For the Study of Natural Language Communication Between Man and Machine. Communications of the ACM Volume 9, Number 1 (January 1966): 36-45. Disponível em: Acesso em: 24.fev. 2003.