AGENT-BASED ARCHITECTURE FOR WEB DEPLOYMENT OF
MULTIAGENTS AS CONVERSATIONAL INTERFACES

Ranjit Kumar Pothuru, B.Tech., M.S

Problem in Lieu of Thesis Prepared for the Degree of
MASTER OF SCIENCE

UNIVERSITY OF NORTH TEXAS
May 2003

APPROVED:
Paul Tarau, Major Professor
Robert Brazile, Program Coordinator
Krishna Kavi, Chair of the Department of Computer Science
C. Neal Tate, Dean of the Robert B. Toulouse School of Graduate Studies
Pothuru, Ranjit Kumar, *Agent-based architecture for web deployment of multi-agents as conversational interfaces*. Master of Science (Computer Science), May 2003, 33 pp., 6 illustrations, references, 12 titles.

Agent based architecture explains the rationale and basis for developing agents that can interact with users through natural language query/answer patterns developed systematically using AIML (artificial intelligence mark-up language) scripts. This thesis research document also explains the architecture for VISTA (virtual interactive storytelling agents), which is used for interactive querying in educational and recreational purposes. Agents are very effective as conversational interfaces when used along side with graphical user interface (GUI) in applications and Web pages. This architecture platform can support multiple agents with or without sharing of knowledgebase. They are very useful as chat robots for recreational purposes, customer service and educational purposes. This platform is powered by Java servlet implementation of Program D and contained in Apache Tomcat server. The AIML scripting language defined here in is a generic form of XML language and forms the knowledgebase of the bot. Animation is provided with Microsoft® Agent technology and text-to-speech support engine.
ACKNOWLEDGMENTS

At the outset I would like to thank all the people who are part of Alicebot open source community and Dr. Richard Wallace for his vision and effort to make his brain child Alice what it is today. I am very thankful to Dr. Paul Tarau, for his immense guidance and continuous encouragement through out my research work. He has been very patient and great visionary in application of agents for entertainment and educational purposes. My heartfelt thanks to all my friends and colleagues at the department of computer science, for their support and encouragement.
# TABLE OF CONTENTS

ACKNOWLEDGMENTS........................................................................................................... ii

LIST OF ILLUSTRATIONS........................................................................................................ v

INTRODUCTION......................................................................................................................... 1

HISTORY .................................................................................................................................... 3

Turing Test ............................................................................................................................... 4

ALICE - ARTIFICIAL LINGUISTIC INTERNET COMPUTER ENTITY ...................................... 6

Alicebot Technology .................................................................................................................. 7

Program A ................................................................................................................................. 7

Program B ................................................................................................................................. 7

Program C ................................................................................................................................. 8

Program D ................................................................................................................................. 8

The Architecture ....................................................................................................................... 8

The Alicebot Engine .................................................................................................................. 9

The Server ................................................................................................................................. 10

ARTIFICIAL INTELLIGENCE MARKUP LANGUAGE (AIML) .................................................... 13

Atomic ....................................................................................................................................... 14

Default ....................................................................................................................................... 15

Recursion .................................................................................................................................. 15

Targeting ................................................................................................................................. 17

Context ....................................................................................................................................... 17

Normalization ........................................................................................................................... 17

Pattern Matching ..................................................................................................................... 18
The Brain ....................................................................................................................... 19

IMPLEMENTATION ........................................................................................................ 21

The What? ....................................................................................................................... 21

The How? ......................................................................................................................... 22

APPLICATIONS WITH AGENTS .................................................................................. 24

VISTA Project ................................................................................................................ 25

CONCLUSIONS ............................................................................................................... 26

APPENDIX ..................................................................................................................... 28

REFERENCES ............................................................................................................... 33
# LIST OF ILLUSTRATIONS

<table>
<thead>
<tr>
<th>Figure</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1</td>
<td>The Alicebot Engine</td>
<td>9</td>
</tr>
<tr>
<td>Figure 2</td>
<td>The Alicebot Server</td>
<td>12</td>
</tr>
<tr>
<td>Figure 3</td>
<td>Big Spiral</td>
<td>29</td>
</tr>
<tr>
<td>Figure 4</td>
<td>Eye Spiral</td>
<td>30</td>
</tr>
<tr>
<td>Figure 5</td>
<td>Log Spiral</td>
<td>31</td>
</tr>
<tr>
<td>Figure 6</td>
<td>SRAI Spiral</td>
<td>32</td>
</tr>
</tbody>
</table>
INTRODUCTION

A bot (short for robot) that is derived from the Czech word robota meaning work, is a tool for digging through data. On the Web, bots took a different form of life and are mostly used to perform a methodical task. The term bot itself is used interchangeably with agent. But strictly speaking agents are bots that have a mission to accomplish. Some bots operate in place while others are moving on the Internet collecting the information or performing the desired tasks. In the current context we can define agent as a technology that provides a foundation for more natural ways for people to communicate with their computers. It is a set of software services that can be used to incorporate interactive animated characters into their applications and Web pages. Enhancing applications and Web pages with a visible interactive personality will both broaden and humanize the interaction between users and their computers.

Agent enables software developers to incorporate a new form of user interaction, known as conversational interfaces, that leverages natural aspects of human social communication. In addition to input forms like mouse, keyboard it can support speech recognition and perform a certain task or respond back using synthesized speech, recorded audio, or text. The conversational interface approach is not an alternative to conventional approach using graphical user interface, but only enhances it by making it more lively with the added human touch.

Bots have great potential in data mining, the process of finding patterns in enormous amounts of data. Because data mining often requires a series of searches, bots can save labor as they persist in a search, refining it as they go along. Intelligent bots can make decisions based on past experiences, which will become an important
tool for data miners trying to perfect complex searches that delve into billions of data points. This aspect about intelligent bots to make decisions using knowledgebase has been explored and effectively used in conversational interfaces.
HISTORY

Agents (bots) are not invented specifically for Internet. However, from a historical perspective robotic software is generally believed to have been created in the form of Eliza, one of the first public displays of artificial intelligence. Eliza parodies a psychotherapist by answering questions with questions. Eliza's programming is quite simple. It analyzes each question and tries to identify keywords or multiple words that match programmed responses. Artificial intelligence is an advanced field of computer science that aims to develop software capable of processing information on its own, without the need for human direction. With the advancements due to research in the field of natural language processing and artificial intelligence, the application development with agents as conversational interfaces has attracted a lot of attention.

Expert systems are developed by programming computers to make decisions in real-life situations. In the early 1980s, expert systems were believed to represent the future of artificial intelligence and of computers in general. To date, however, they have not lived up to expectations. Many expert systems help human experts in such fields as medicine and engineering, but they are very expensive to produce and are helpful only in special situations.

Natural language processing offers the greatest potential rewards because it would allow people to interact with computers without needing any specialized knowledge. Anybody could simply walk up to a computer and talk to it. Unfortunately, programming computers to understand natural languages has proved to be more difficult than originally thought. Some rudimentary translation systems that translate from one human language to another are in existence, but they are not nearly as good
as human translators. There are also voice recognition systems that can convert spoken sounds into written words, but they do not understand what they are writing; they simply take dictation. Even these systems are quite limited in their capability and would need anybody to speak slowly and distinctly.

Turing Test

Alan C. Turing described a chat robot based game in 1950 referred to as original imitation game (OIG). The OIG has three players: a man (A), a woman (B), and a third person (C) of either sex. The third player (C) is called the interrogator, and its function is to communicate with the other two, through what would nowadays be called a text-only instant messaging chat interface, using two terminals (or today perhaps, two windows) labeled (X) and (Y). The interrogator must decide whether (X) is (A) and (Y) is (B), or (X) is (B) and (Y) is (A), in other words, which is the man and which is the woman. The interrogator’s task is complicated by the man (A), who Turing says should reply to the interrogator with lies and deceptions. For example, if the man is asked, “Are you a man or a woman?” he might reply, “I am a woman.” [1]

Turing’s point is that by considering the OIG as an actual scientific experiment, if we were to actually conduct the OIG with a sufficiently large sample of subjects playing the parts of (A), (B), and (C), then we could measure a specific percentage M of the time that, on average, the interrogator misidentifies the woman, so that (100 - M)% of the time she is identified correctly. Given enough trials of the OIG, at least in a given historical and cultural context, the number M ought to be a fairly repeatable measurement.[1]
The setup for Turing test is similar to the OIG, but now gender plays no role. The player (B) is called “a man” and the player (A) is always a computer. The interrogator must still decide whether (X) is (A) and (Y) is (B), or (X) is (B) and (Y) is (A), in other words, which is the man and which is the machine. The Sterett Turing Test, though flawed as an experiment, has come to be popularized as the modern “Turing Test.” The point is, when Turing predicted that a machine could play his game in “50 years,” he envisioned something more like a general purpose learning machine, which does not yet exist. The concept is simple enough: build a robot to grow like a child, able to be taught language the way we are. In our terms, the role of the botmaster would be fully automated. But even a child does not, or at least should not, go forth into the world, unprotected, to learn language “on the street,” without supervision. This learning process is accomplished by “targeting”.

The Artificial Linguistic Internet Computer Entity (ALICE) is an open source chatterbot developed by a research group called ALICE A.I foundation. ALICE chat robot is based on an experiment specified by Alan M. Turing in 1950. It uses a sophisticated pattern-matching case-statement technology to create a very convincing illusion of natural conversation. We can say, simply, it uses case-based reasoning. In comparison, Eliza's programming is quite simple. It analyzes each question and tries to identify keywords or multiple words that match programmed responses.

ALICE even tells lies and spreads gossip told to it by other users. This is accomplished by its ability to learn and store gossip. The conversation is very lively and impersonates human-like behavior. But it really is a machine. The model of learning in ALICE is called supervised learning because a person, the botmaster, plays a crucial role. The botmaster monitors the robot's conversations and creates new artificial intelligence markup language (AIML) content to make the responses more appropriate, accurate, believable, or “human,” or whatever the botmaster intends. With the process called “targeting,” algorithms for automatic detection of patterns in the dialog data have been developed. [2] This process provides the botmaster with new input patterns that do not already have specific replies, permitting a process of almost continuous supervised refinement of the bot. Alicebot engine uses AIML to formulate responses for the user inputs. The functionality and architecture of the engine will be described in further sections.
Alicebot Technology

Alicebot engine is implemented today in many platform languages like Java™, C, C++, Perl, Lisp and PHP scripts etc are quite popular. But from historical perspective the way Alicebot got started is quite interesting. It was believed there was no theory behind ALICE, no neural networks, no knowledge representation, no deep search, no genetic algorithms and no parsing. But Alicebot is based on a theory in applied artificial intelligence (AI), called case-based reasoning (CBR) that closely resembled the stimulus-response structure of ALICE. The CBR cases correspond to the AIML categories.

Program A

The first edition of ALICE was implemented in 1995 using SETL, a widely unknown language based on set theory and mathematical logic. The original ALICE was available as free software. It later was migrated to the platform-independent Java language in 1998. The first implementation of ALICE and AIML in Java was codenamed "Program A." This version of Alicebot was implemented in pre-Java 2 and became very popular among the research community. The version of AIML supported in this engine is v 0.9. [2]

Program B

Launched in 1999, Program B was a breakthrough in ALICE free software development. More than 300 developers contributed to ALICE Program B. AIML transitioned to a fully XML-compliant grammar, opening up a whole class of editors and

TM Sun Microsystems, Inc. (www.sun.com)
tools for AIML development. ALICE Program B won the Loebner Prize, an annual Turing Test event, in January 2000. Program B was the first widely adopted free AIML software.

Program C

Program C was the C/C++ implementation of AIML that was released in 2000. This was followed by a number of development threads in C/C++ that brought the Alicebot engine to CGI-scripts, IRC, Microsoft® Windows®, AOL® Instant Messenger, and COM. This collection of code has come to be known as "Program C," the C/C++ implementations of the Alicebot engine and AIML.

Program D

Program B Java edition was based on pre-Java 2 technology. Although the program ran on many different platforms, it did not take advantage of newer Java features such as Swing and Collections. Jon Baer recoded Program B with Java 2 technology, and added many new features. This giant leap in the interface and the core, and the fact that bot was named "DANY", hence next code letter "D" was assigned to the latest Alicebot Java edition. [2]

This is the version of engine that has been used in implementation of my architecture platform for the virtual interactive storytelling agent project (VISTA) underway and also for developing conversational user interfaces in conjunction with the traditional graphical interfaces (GUI).

The Architecture

The design of Alicebot can be described as made of 2 major parts: the Alicebot engine that drives the bot, and the server wrapper. The two put together enable the bot
to be used in embedding it in any application and provide services via Web. The architecture described here in supports multiple bots simultaneously. These bots can be designed to have unique animation features and can either share AIML scripts with other bots in which case they do not exhibit any special characteristics. Alternately they can have specific AIML script files that are targeted for a particular purpose. This flexibility allows choosing multiple agents and designating each of them for a specific task or having several bots to choose from that can all do the same thing.

*The Alicebot Engine*

- Responder – forms the interface between the user and core routines. It handles the input from the user and necessary formatting is done before it is passed on to the classifier. It also handles the output to be delivered to the user from the classifier.
• Classifier – normalizes and filters the input. It applies substitutions as defined in the knowledgebase appropriately. If the user input is in a complicated larger sentences it will be broken into smaller sentences so they are available to the GraphMaster for readily working on and also handles various AIML instructions. It also delivers the bot response to the responder.

• GraphMaster – forms the core of the Alicebot engine. It organizes storage of brain content. 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. It works just like a dictionary or encyclopedia. In order to look up a word or phrase, one need not start at the beginning or the end and search through every entry until a match is found. Instead, first turn to the section that matches the first letter or word. Then, skip to another section that contains a set beginning with the next letter or word. Continue this process until the word or phrase is found. [5] The root of the GraphMaster is a Nodemapper with about 2000 branches, one for each of the first words of all the patterns. The number of leaf nodes in the graph is equal to the number of categories, and each leaf node contains the <template> tag.

The Server

In its original form Program D has a server implemented in Java with multithreading to handle multiple requests simultaneously, the only limit being the system resources, the server can handle any number of connections theoretically speaking. But in reality keeping in view with the system resources the number of connections can be limited.
Jakarta Tomcat Server

In order to utilize the resources fully and also security point of view the implementation of server has been carried out with the help of servlet containers. Hence the Alicebot engine has been reprogrammed as a servlet and is contained in standard servlet container called Tomcat. Java servlet technology provides Web developers with a simple, consistent mechanism for extending the functionality of a Web server and for accessing existing business systems. Servlets are the Java platform technology of choice for extending and enhancing Web servers. Servlets provide a component-based, platform-independent method for building Web-based applications, without the performance limitations of CGI programs. And unlike proprietary server extension mechanisms (such as the Netscape Server API or Apache modules), Servlets are server and platform independent. Tomcat server is one such open source servlet container and can be coupled with the very popular Apache server. By installing Apache server it makes it feasible to serve the remaining HTML and other application content onside with the Alicebot engine developed as a servlet. This way it provides the flexibility of seamlessly integrating the conversational interfaces into many applications.

The Apache server can handle the Web requests on the ports specified (generally 80) and it is highly scalable and becoming an industrial standard for performance. It uses a software module called Jakarta ant to integrate the Tomcat server with Apache. This way all the http requests are received at one port and the clients are served appropriately from either the Apache server (regular application and html stuff) or Tomcat (which handles the servlet requests. The process of integration is
quite pain taking on Windows® platform and has a lot of steps to be followed. This provides the architectural platform for the server part of the Alicebot.

![Diagram of Alicebot Server](image)

**Figure 2** The Alicebot Server

---

* Microsoft Corporation, [www.microsoft.com](http://www.microsoft.com)
Artificial intelligence mark-up language (AIML) enables people to input knowledge into chat-bots based on the ALICE software technology described earlier. AIML is an XML-compliant language. The Alicebot free software community developed it during 1995-2000. AIML describes a class of data objects called AIML objects and partially describes the behavior of computer programs that process them. AIML objects are made up of units called topics and categories, which contain either parsed or unparsed data.Parsed data is made up of characters, some of which form character data, and some of which form AIML elements. AIML elements encapsulate the stimulus-response knowledge contained in the document. Character data within these elements is sometimes parsed by an AIML interpreter, and sometimes left unparsed for later processing by a responder.

The most important units of AIML are:

- `<Aiml>`: the tag that begins and ends an AIML document
- `<Category>`: The basic unit of knowledge in AIML is called a category.
  Each category consists of an input question, an output answer, and an optional context.
- `<Pattern>`: The question, or stimulus, is called the pattern. The AIML pattern language is simple, consisting only of words, spaces, and the wildcard symbols `*` and `*`. The words may consist of letters and numerals, but no other characters. The pattern language is case invariant. Words are separated by a single space, and the wildcard characters function like words. [1]
The answer, or response to the user's question is called the template. In its simplest form, the template consists of only plain, unmarked text. More generally, AIML tags transform the reply into a mini computer program which can save data, activate other programs, give conditional responses, and recursively call the pattern matcher to insert the responses from other categories.

If we just assume we have only pattern and template tags there are three general types of categories. [1]

- atomic
- default
- recursive

Strictly speaking, the three types overlap, because "atomic" and "default" refer to the <pattern> and "recursive" refers to a property of the <template>.

Atomic

These are the categories with atomic patterns, i.e. the patterns do not contain wild card "*" or "_" symbol. Atomic categories are very simple and basic categories and hence the name. It is very easy to add them to AIML. [1] For example, consider

```xml
<category>
<pattern>WHAT IS A COMPUTER</pattern>
<template> A Computer is a programmable electronic device that can store, retrieve, and process data. </template>
</category>
```

The above category does the following:
- Matches the client input of "What is a computer"
- Sends the client the response: "A computer is a programmable electronic device that can store, retrieve, and process data."

Default

The name "default category" derives from the fact that its pattern has a wildcard "**" or "_. The ultimate default category is the one with `<pattern>*</pattern>`, which matches any input. These default responses are often called "pickup lines" because they generally consist of leading questions designed to focus the client on known topics. This helps the bot to handle, when a question with not in the knowledgebase is asked, so that it will redirect the user by smartly changing the topic of conversation and keep it going.

The more common default categories have patterns combining a few words and a wildcard. For example, consider:

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

This category responds to a variety of inputs all that start with “I need help.”

Putting aside the philosophical question of whether the robot really "understands" these inputs, this category elucidates a coherent response and gives the client an impression of the robot understanding the client’s intention.

Recursion

AIML implements recursion with the `<srai>` operator. In the acronym, "AI" stands for artificial intelligence, but "SR" may mean "stimulus-response," "syntactic
rewrite," "symbolic reduction," "simple recursion," or "synonym resolution." The disagreement over the acronym reflects the variety of applications for $<\text{srai}>$ in AIML.

1. Symbolic reduction – reduce complex grammatical forms to simpler ones. Usually the atomic patterns in categories storing robot knowledge are stated in the simplest possible terms. Many of the more complex forms reduce to simpler forms using AIML categories designed for symbolic reduction.

2. Divide and conquer – many individual sentences may be reduced to two or more sub sentences, and the reply formed by combining the replies to each.

3. Synonyms – the AIML 1.01 standard does not permit more than one pattern per category. Synonyms are perhaps the most common application of $<\text{srai}>$. With this many ways to say the same thing reduce to one category, which contains the reply. [1]

4. Spelling or grammar corrections based on context.

5. Detecting keywords anywhere in the input. By doing this it is possible to activate an AIML template when ever certain keywords are found anywhere in the input sentence.

6. Conditionals – certain forms of branching may be implemented with $<\text{srai}>$.

7. Any combination of 1-6.
Targeting

Bot personalities are created and shaped through a cyclical process of supervised learning called targeting. Targeting is a cycle incorporating client, bot, and Botmaster, wherein client inputs that find no complete match among the categories are logged by the bot and delivered as targets the botmaster, who then creates suitable responses, starting with the most common queries. The targeting cycle produces a progressively more refined bot personality.

The art of AIML writing is most apparent in creating default categories, which provide noncommittal replies to a wide range of inputs.

Context

The keyword that in AIML refers to the robot’s previous utterance. Specifically, if the robot responds with a multiple sentence paragraph, the value of that is set to the last sentence in the sequence. The choice of the keyword that is motivated by its use in ordinary language. Sometimes human beings in their conversation may respond with a simple “yes” or “no” in which case it is necessary to keep track of the conversation and the relevance of the response. This way it would be possible to figure out as to what they are saying “yes” to and follow from there. [5]

Normalization

In the Alicebot, Program D has a class called Substituter that performs a number of grammatical and syntactical substitutions on strings. The process called normalization involves pre-processing sentences to remove ambiguous punctuation to prepare the input for segmentation into individual sentence phrases and expand all
contractions and converts all letters to upper case. By doing this patterns are ready to be used in case based matching.

Pattern Matching

The pattern-matching algorithm is a modified version of the DFS (depth first search). Unlike the normal DFS, the order in which the relevant subnodes traveled is important, because (when using wildcards) the number of paths that match any given input is quite large (somewhere between $3^n$ and $4^n$ for an input of $n$ words), and we need only one (preferably the best). This pattern matching is carried out in the GraphMaster. The algorithm followed in accomplishing this can be explained in 3 steps.

Given (a) an input starting with word "X", and (b) a Nodemapper of the graph:

1. Does the Nodemapper 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. Does the Nodemapper 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. Does the Nodemapper 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.

The terminal condition is satisfied when the input string is null (no more words) and the Nodemapper contains the <template> key then a map is found.
If the root Nodemapper contains a key "**" and it points to a leaf node, then the algorithm is guaranteed to find a match.

Some of the things of importance in pattern matching described above are:

1. At every node, the "_" has first priority, an atomic word match second priority, and a "***" match lowest priority.

2. The patterns need not be ordered alphabetically, only partially ordered so that "_" comes before any word and "***" after any word.

3. The matching is word-by-word, not category-by-category.

4. 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.

5. The matching algorithm is a highly restricted version of depth-first search, also known as backtracking.

6. The algorithm can be simplified by removing the "_" wildcard, and considering just the second two steps.

The Brain

As explained earlier the knowledgebase of the bot is the vital part in its impersonating of human-like behavior. The GraphMaster loads all the categories in the files and the brain of the bot is thus constructed. To better understand the structure of the AIML patterns and their distribution graphs of Alice have been created. More than just an elegant graph of the ALICE brain, these spiral images outline a territory of language that has been effectively "conquered" by ALICE and AIML. A complex theory
of learning, neural nets or cognitive models are not needed to chat within the limits of Alice’s 50,000+ categories. The stimulus response model followed here in is quite simple and is as good as any theory for these cases. Some of the pictures of Alice Brain are enclosed in the Appendix (A1- A4)
IMPLEMENTATION

The What?

The Alicebot engine has been implemented in Java originally. The engine has been reimplemented as a servlet by using the core classes of GraphMaster and jetty class. The program has been implemented with multithreaded architecture and is highly scalable. It can be customized and the number of connections can be set to a predetermined number. Also it is possible to specify the port on which the bot can be accessed. The bot at present can be run stand alone as an interactive chatterbot (short for chat robot) and can be connected to many of the popular chat programs. It exhibits quite a human like behavior and uses an extensive knowledgebase of more than 50,000 different artificial intelligence mark-up language (AIML) categories. All these categories reside in different text files that are digested by the bot at the time of starting the bot. The engine constructs a graph as defined earlier in GraphMaster module. It uses a static approach to category matching. This means that the whole path (with all its components) gets created at load time.

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 ALICE brain). The number of leaf nodes in the graph is equal to the number of categories, and each leaf node contains the <template> tag. The GraphMaster takes each string from input file, separates it into 'words' and stores each of these as nodes in a tree. A simple analogy could be of a directory structure, where the individual words make up the directory names - hence the name 'path' of the category. [2]
Apache server has been installed along with Tomcat server as a stand-alone Web service. They are both integrated by Jakarta Ant software module and JK connector. This JK Connectors are used to resolve the data requests received at the port 80 and are suitably serviced by either Apache server or Tomcat as needed. The process of integrating Tomcat and Apache server is quite seamless in Linux®† environment and is slightly more complicated in case of Windows®‡. Since the implementation of Alicebot itself carried out using Java Servlets, they provide a component-based, platform-independent method for building Web-based applications, without the performance limitations of CGI programs. The interface also incorporates the animation using Microsoft Agent® technology. It employs a proprietary text-to-speech engine at this point of time. This text-to-speech engine is free software and can be downloaded run time at the client side. At this point of time it has scripts written for the VISTA project that have a couple of stories that can be told to kids by using animation and speech and falls back on to Alicebot for all its capabilities to handle any user queries. If the query is not part of the predefine story line, it banks on to the default categories of the extensive knowledgebase in AIML.

The How?

Initially when the bot is started it reads all the preferences and settings defined in a file and initializes them. Then the loader reads categories one by one as they appear in the file, parses and stores the contents of the pattern, that and topic elements and when it reaches the end of the category, calls a method to store them in the tree

† Registered trademark name belonging to Linus Torvalds.
‡ Microsoft, Inc., www.microsoft.com
together with the template. At this stage it does not do any processing of the contents of
the templates but just stores them. It is also possible to use a standard database like
SQL or JDBC and store these categories and retrieve them sequentially. This is an
alternate way of implementing.

Now when the bot is up and running it waits for user input, preprocesses it before
it can be matched to a category. In preprocessing first substitutions are applied in a
rather straightforward manner with a main purpose to protect the sentences and
eliminate spelling mistakes which might be an obstacle for case matching. At this stage
synonyms can also be defined so that consistent response is possible for all similar
questions. After this all the redundant white space is removed and the input is split into
sentences if needed. The key here is though it analyzes the sentences it does not
necessarily respond to every single sentence. The final response is formed after the
stage of pattern matching is carried out. Once we have matched the input and have a
response template, we need to process it with the help of a parser. Then the multiple
sentences are combined together and returned as a final response to the user input.
APPLICATIONS WITH AGENTS

Agent is a technology that provides a foundation for more natural ways for people to communicate with their computers. As described earlier it is actually a set of software services that can be used to incorporate animated characters into software application and Web pages. These characters can speak or accept voice commands. In order to accomplish this they use text-to-speech (TTS) and speech-to-text Engines available on the Web. Microsoft Corporation’s TTS engine is very popular and is supported in most browsers. These agents are easy to embed into applications and Web pages by using the OLE concept. They can be included with Web pages or application using VB, ASP and Java server pages. The Microsoft Agent Virtual Basic script appears as embedded HTML in the client reply. The clients need not have the software plug-ins required for the animation to work. When a Web page or application with the agents embedded in it is accessed they are downloaded on the fly from Microsoft server when the script starts. Sometimes it is quite time taking and depends on the bandwidth availability. Some of the latest operating systems like Windows XP and software packages like Microsoft Office XP or later come ready with TTS and animation capability and does not need any downloads for that to work.

Where applicable

They have a limitless number of roles and functions.

- As a welcome host
  - Greet new users
  - Provide a guided tour.

- A friendly tutor
  - Could lead someone through a task
  - Walk a decision tree with instructions step-by-step
As a virtual interactive story-telling agent (VISTA), can interact with user on their questions

- A messenger
- An assistant
  - could perform tasks for you, like looking up information on the Internet and then reading it out loud.
- Customer service and support

**VISTA Project**

The Vista (virtual interactive story-telling agent) Project is part of University of North Texas digital storytelling project developed to support the online teaching of storytelling. Storytelling is a performance based learning experience and the use of digital technologies is of critical importance. Professional storytellers are recorded in the studio and streamed on demand to the user. To make the interaction between the user and the storyteller the agents have been deployed on this site. These agents have questions and categories related specifically to each of the story loaded into their knowledgebase. These AIML scripts are generated from transcripts of the live performances of artists and professionals. The agents interact with the user through natural language query/answer patterns to establish dialogue about the narrative content. This type of interactions allows the users to learn about the content of a story by asking questions and some clarifications. The AIML scripts are designed to answer questions about themes, characters, events and motifs. The answers support individual’s learning style and needs.
CONCLUSIONS

This paper has described the agent-based architecture for VISTA project for virtual interactive story telling and the technical design of the agent used to build interactive agents. This project uses Alicebot engine implemented using Java Servlets as knowledge processor, Web interface through Apache Tomcat server and artificial intelligence mark-up language (AIML) scripts that incorporate the knowledge base for the stories as well as more than 40,000 general categories. Multiple bots can be supported on this platform all sharing the same knowledgebase or each bot can be designated with a specific purpose and carries unique knowledge. The agents interact with the user through natural language query/answer patterns to establish dialogue about the narrative content. This architecture in general establishes a platform that can be used for building and using agents as conversational interfaces in applications and Web pages. These conversational interfaces are not an alternative for graphical user interfaces (GUIs) but enhance it by humanizing the interaction between the computer and user. By humanizing the interaction it provides both entertainment pleasure and serves cognitive purpose by enhancing the narration and providing proper guidance in navigation etc. The agent technology has applications in online teaching, accepting voice commands and accomplishing specific tasks, walking through a decision process, story narration and querying and building virtual environments. They are successful in engaging human beings in a meaningful conversation. Finally with enhanced capabilities of text to-speech and speech to-text, these agents will be of very good use in customer service answering the routine questions when a knowledgebase is
developed for this purpose. They do not complain of long hours and never get emotional at the user irrespective of the repetition in conversation and are highly consistent.
APPENDIX

ILLUSTRATIONS OF BRAIN OF AIML BOT
This high-resolution 1024x768 plot shows the ALICE Brain with 24,637 categories loaded. The spine is actually a log spiral, but with an exponent close to unity, so very nearly the same as a linear spiral. Gray lines indicate nodes with exactly one branch. Black lines are nodes with two or more branches. The leaf nodes have two branches because they store both <template> and <filename>.
Figure 4  Eye Spiral (used with permission: Dr. Richard Wallace, Alicebot.org)

The eye-shaped log spiral plots all 24,000 categories in the ALICE Brain. The spiral itself represents the root. The trees emerging from the root are the patterns recognized by ALICE. The branching factor for the root is about 2000, but the average branching factor of the second pattern word is only about two.
This is a logarithmic plot of spiral distribution of categories. The cortical architecture that enables real-time, attention-based visual processing, can in fact be applied to linguistic processing as well.
This plot shows 1/4 of the patterns in Srai.aiml, or about 2000 patterns total. The four big trees in the lower right are all the patterns rooted at the words WHAT, WHEN, WHERE, and WHO respectively (from right to left). These areas show the compression power of the Graphmaster graph at its best.
REFERENCES


List of bots information. Available at http://www.botspot.com

Sandy Fritz, Scientific American, Rodney Brooks. Understanding Artificial Intelligence (Science Made Accessible)


Microsoft Agent technology: Microsoft Agent User Interface. Available at http://www.microsoft.com/msagent/default.asp


33