DiaSys – A dialogue system based on Conditional Knowledge in client-server technology

Cristina Zamfir (Tudorache)

***Abstract*—**A conditional schema is a graph-based structure which is able to represent conditional knowledge. In [1] the structure of a conditional schema was introduced and in [2] the inference mechanism corresponding to the conditional schema representations was developed. The Knowledge Representation and Reasoning (KR&R) is based on the idea that propositional content can be rigorously represented in formal languages, such that the resulted representations can be productively reasoned over by humans and machines [16]. Conditional Knowledge Representation and Reasoning represents a new brand of KR&R, for which several formalisms have been developed. The DiaSys application based on conditional knowledge was presented in [4]. In this paper we describe the evolution of that dialogue system into a client-server application. The conditional schema will be defined and exemplified. The server and the client will be presented step by step in two separate sections. The two sections are oriented specifically towards the task that the described part of the application performs.

***Keywords*—**Client-server, Dialog System, Knowledge base, Natural language, XML.

I. INTRODUCTION

In prior articles there was presented a practical implementations of a system that interrogates a conditional knowledge base. In [3] there was presented a Question Answering System that can be use in automatic training and system malfunctions diagnostics.

In [4] there was presented a Dialog System application called DiaSys. In this paper we will present the DiaSys application that evolved into a client server application from a programmatic perspective.

A dialogue system is a computer system intended to converse with a human, with a coherent structure. Dialogue systems have employed text, speech, graphics, gestures and other means of communication on both the input and output channel.

This article was produced under the project "Supporting young Ph.D students with frequency by providing doctoral fellowships", co-financed from the EUROPEAN SOCIAL FUND through the Sectoral Operational Program Development of Human Resources

In general by a *dialogue system* we understand a software system designed to provide answers to questions that are formulated by a user, in natural language. Usually a dialog system maintains the conversation by asking the user different questions in order to produce a more accurate answer. This will be further discussed in the sections below.

To find the answer to a question, a system may use a database, a collection of natural language documents or a knowledge base. Various such systems were developed.

The term client/server describes the relationship between two computer programs in which one program, the client, makes a service request to another program, the server, which fulfills the request. Although the client/server idea can be used by programs within a single computer, it is a more important idea in a network. In a network, the client/server model provides a convenient way to interconnect programs that are distributed efficiently across different locations. Computer transactions using the client-server model are very common.

The client-server model has become one of the central ideas of network computing. Most business applications being written today use the client/server model because the client-server model architecture provides a convenient way to interconnect programs that are distributed efficiently across different locations. Computer transactions using the client/server model are very common. There are two main entities for every client/server application: one entity is the client and another is the server. Usually the server and the client are found on separate computers.

The client/server programming is based on the idea that the server is a provider of services. Client programs request service from a server by sending it a message. Server programs process client requests by performing the tasks requested by clients. Servers are generally active as they wait for a client request. During these waiting periods servers can perform other tasks. Unlike the client, the server must continually run because clients can request service at any time. Clients on the other hand only need to run when they require service. Many server applications allow for multiple clients to request service.

In this paper we use the client-server technology based on Java mechanisms using sockets and threads. Both the server side and the client side of the application are presented.

The paper is organized as follows: section II gives the definition of the Conditional Schema as introduced in [2], the conditional graph and an example of a conditional graph given

by an example. Section III describes the DiaSys application functions and clarifies how a normal user and an administrator can work with the system. Section IV gives some general information about the client-server technology and the subsections A and B explain step by step the actions performed by the server and by the client. Section V contains the conclusions and some possible extent of this paper.

II. Conditional knowledge

The concept of conditional knowledge was introduced in [1]. The inference mechanism of this structure is treated in [2].

A conditional schema is a tuple $S = (Ob, C_s, E_r, A, V, B_{cr}, h, f)$ such that:

- Ob is a set of the object names. This set is divided into two subsets Ob_{ind} and Ob_{abstr} such that $Ob = Ob_{ind} \cup Ob_{abstr}$ and $Ob_{ind} \cap Ob_{abstr} = \emptyset$;
- C_s is a finite set of symbols named conditional symbols;
- E_r is a finite set of symbols used to designate conditional binary relations over Ob;
- A is a set of attribute names for the elements of Ob;
- V is a set of values for the elements of A;
- $B_{cr} \subseteq 2^{((Ob\times I)\times(Ob\times I))\times(Cs\cup\{T\})}$ is the set of the conditional binary relations;
- $h : E_r \rightarrow B_{cr}$ is a mapping that assigns a conditional binary relation for every symbol of E_r;
- $f : Ob_{ind} \rightarrow 2^{A\times V}$ is a mapping that assigns initial knowledge to the individual objects of Ob_{ind}.

The conditional graph ([3]) generated by S is the system $G_S=(X \cup Z, \Gamma_X \cup \Gamma_Z)$ where:

- $X \subseteq Ob \times I$ is the set of nodes such that $x \in X$ if and only if $\exists r \in E_r$, $y \in Ob \times I$ such that $(x,y) \in_c h(r)$ or $(y,x) \in_c h(r)$;
- $\Gamma_X \subseteq X \times E_r \times X$ and $((n,w_1),r,(m,w_2)) \in \Gamma_X$ if and only if $((n,w_1),(m,w_2)) \in_c h(r)$; the elements of Γ_X are named *arcs of first category*;
- $Z = \{f(x) | x \in Ob_{ind}\}$ and $\Gamma_Z = \{(f(x),x) | x \in Ob_{ind}\}$; the elements of Γ_Z are named *arcs of the second category*.

Let us consider the following knowledge piece:

„*Jane is 37 years old. She teaches math. Jane is the mother of Dana. Dana has big grades in math if her mother is a math teacher. Jane likes to read SF novels. Jane has a brother named Tom that has a son named Mathew. Tom is 35 years old and he is a school director. If Mathew's dad is a school director then Mathew has to study hard to get big grades.*"

The graphical representation of this knowledge piece is shown in Fig. 1. The individual nodes are marked by introducing the letter "i" next to the object name. The abstract objects are marked with the letter "a". The arks are named by the relations they signify.

III. The DiaSys Application

DiaSys is a natural language application that is constructed in Java and interacts with a user trough a graphical interface. The application can be used by two types of users: an administrator that configures the system and an end user that consults the application in order to access the desired information.

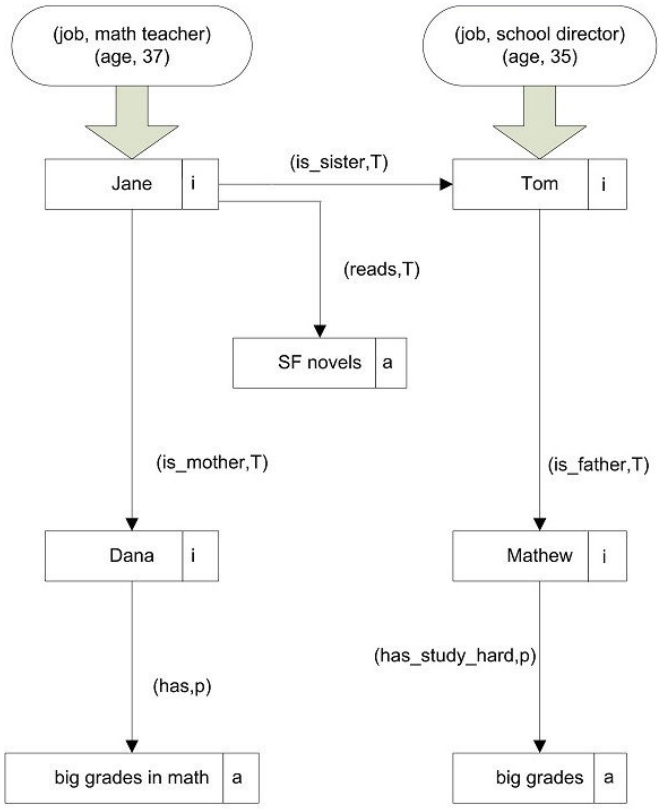

Fig. 1 - The graphical representation of the knowledge piece

The architecture of this application has already been presented in [4].

DiaSys is a system that is used to perform a conversation between a user and the system. The conversation uses natural language.

A user must connect to the system through the graphical end user interface that is presented in Fig. 2. The user can address the system with a direct question or it can start a so called person-to-person dialogue.

For example, the phrase "What car should I buy?" is considered to be a direct question. In this case the system needs to ask some additional questions in order to select the appropriate knowledge base. This will start a person-to-person dialogue.

An example of the possible dialogue is presented below:

```
User: What car should I buy?
System: Hello. I am DiaSys. In order to
answer your question I will need some
information about you. How old are you?
```

```
User: 25.
System: Are you a male or a female?
User: Male.
System: Do you own a driving license?
User: Yes.
System: What should be the price range for the car?
[... Some other questions that will help the system narrow down the list of cars will be asked. The system will send the response by mail to the user...]
```

All the questions that the system has asked the user are meant to narrow the list of possibilities that the system has. Furthermore, by answering the questions the user will obtain a personalized answer that will satisfy the needs of the user. The questions are asked based on the attributes described in the knowledge base.

An example of a person-to-person dialogue is shown below.

```
User: Hello
System: Hello. I am DiaSys. Can you please tell me your name?
User: I am Peter.
System: How can I help you, Peter?
User: I would like to receive a copy of my phone invoice.
System: I need to verify some information first.
[... The system will ask the user for personal information ...]
System: A copy of the invoice has been sent to the e-mail that you provided. Is there some other information that I can help you with?
User: Yes. I would like to know the address of the University of Pitesti.
System: The address is Str. Targu din Vale nr. 1, Pitesti, Romania. Is there some other information that I can help you with?
User: No, thank you.
System: Thank you for using DiaSys. Have a nice day.
```

Based on the information gathered from the user, the dialogue system is capable of selecting the appropriate knowledge base. If the information that the user offers does not suffice additional questions will be asked by de system.

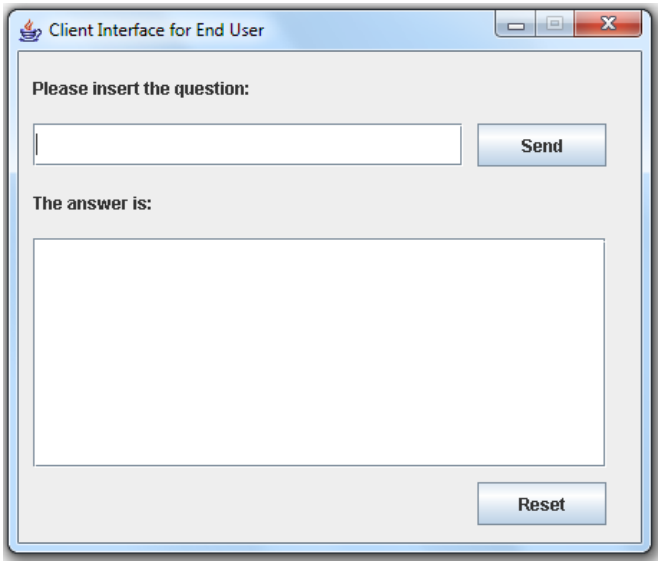

Fig. 2 – The end user graphical interface

The user has to enter a question in the text field and press the "Send" button. The question will be transferred to the Analyze Module that will verify the text and will extract the semantics. If the text is not grammatically correct then the message received from the user will be considered invalid. This information will be passed to the Answer Generation Module that will communicate an appropriate message to the user. If the text received from the user is valid, the Analyze Module will transfer it to the Inference Engine that will perform the inference on the objects. After the inference is performed, the result found by the Inference Module will be transferred to the Answer Generation Module that will transform the answer into a natural language text. The text will be transferred to the graphical interface and the user will be able to see the answer in the text area.

The administrator (also called knowledge engineer) can configure the application through a graphical interface that is exemplified in Fig. 3 or directly by editing the xml files.

Through the administration graphical interface the knowledge engineer can create a conditional schema, add new components to the conditional schema, modify the existing components, delete the components, visualize the schema, delete the conditional schema, and define, modify and visualize the mapping functions.

All the above can also be done by directly writing the xml files that the application uses to retain all the conditional schema information. The xml files that were presented in [4] are used by the application in order to make it more general and configurable. An example of the xml files used in this version of the application is shown below.

```
 <?xml version="1.0" encoding="UTF-8" standalone="no"?>
 <conditional_schema>
 <iobjs>
 i1
 i2
 i3
 i4
 </iobjs>
```

```
<aobjs>
a1
a2
a3
</aobjs>
<CS>
s1
s2
</CS>
<ER>
r1
r2
r3
r4
</ER>
<Attributes>
at1
at2
at4
</Attributes>
<Values>
v1
v2
v3
v4
</Values>
```

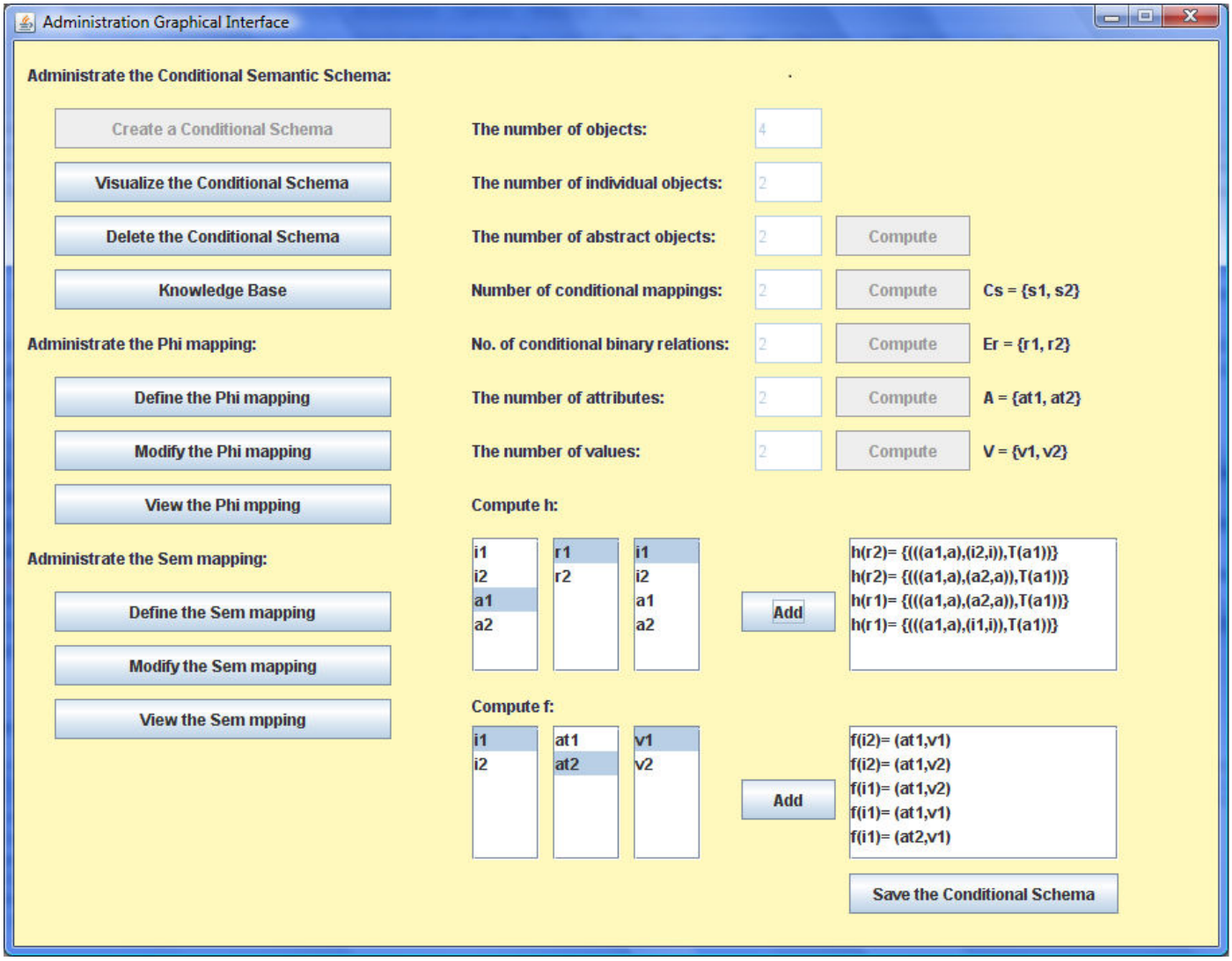

Fig. 3 – DiaSys – Administration Graphical Interface

```
<h_mapping>
h(r1)={(((i1,i),(i2,i)),T(i1))}
h(r2)={(((i1,i),(i3,i)),T(i1))}
h(r2)={(((i2,i),(i4,i)),T(i2))}
h(r3)={(((i3,i),(a2,a)),s1)}
h(r3)={(((i4,i),(a3,a)),s2)}
h(r4)={(((i1,i),(a1,a)),T(i1))}
</h_mapping>
<f_mapping>
f(i1)=(at1,v1)
f(i1)=(at1,v2)
f(i1)=(at2,v1)
f(i1)=(at2,v2)
f(i2)=(at1,v1)
f(i2)=(at2,v2)
f(i2)=(at2,v1)
f(i2)=(at1,v2)
</f_mapping>
</conditional_schema>
```

IV. DiaSys client server

Because of the fact that the DiaSys is intended to be used in health consultancy and other complex systems it is only normal for the application to evolve into a client-server system.

The system was already structured to be used by two types of users that interact differently with the application.

A client server application uses a server and one or more clients. Fig. 4 represents such a case based on network communication.

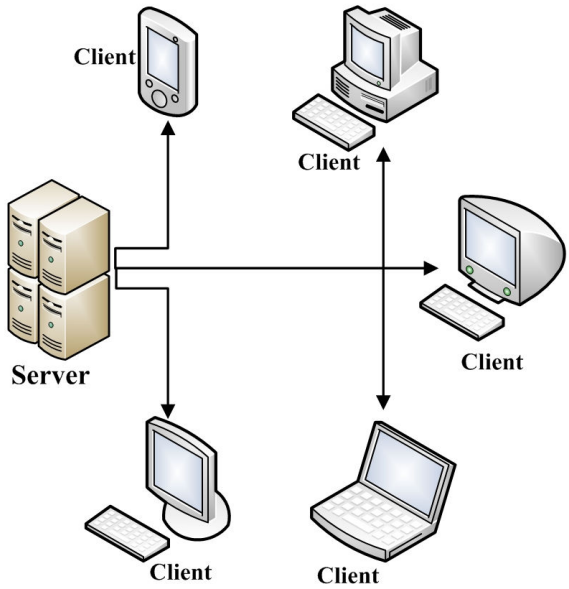

Fig. 4 – Client-server architecture

In a client-server application the client makes a service request and the server fulfills that request. This is very similar with how the application already functions: the user requests an answer and the system interrogates the knowledge base to provide the answer.

In the following sections we will describe the server and the client functions separately.

A. Server side application

In this section we describe the actions of the server. The communication between the server and the client is done using sockets. A socket is one endpoint of a two-way communication link between two programs running on the network. A socket is bound to a port number so that the TCP layer can identify the application that data is destined to be sent to.

An example of the variables used to work with the server is:

```
...
private ServerSocket listenSocket;
private   Socket   connection;   //the
socket type variable
private InputStream inStream;
private DataInputStream inDataStream;
private OutputStream outStream;
private                DataOutputStream
dataOutputStream;
private String message;
private int defaultPort;
private String host;
private  int  port;  //the  port  trough
which the messages are sent
private Connection con;
private boolean connected;
...
```

The actions performed by the server are described in several steps:

Step 1 – Starting the server

The server is started by double clicking the server application icon. The constructor of the server is called, the interface for the server is constructed and the server begins to check for one or more available knowledge bases. It then checks for an existing knowledge base. If the knowledge base is not found the server displays an error message and cannot start connections with the clients.

The error message is:

"The application could not find a knowledge base file. Please configure a knowledge base and try again."

When the knowledge base is found the server moves on to step 2;

```
public   DiaSysServer(String   host,
String portString){
...
boolean exists = false;
ReadXML cxml = new ReadXML();
Vector            vect            =
cxml.read("KBFile.xml",
   "knowledge_base");
if (vect.size() > 0)
   exists = true;
if (exists == false){
   serverMessageArea.append("The
   application could not find a
   knowledge base file. Please
   configure a knowledge base and
   try again");
   stopServer();
   }
```

Step 2 – Starting a connection: The server program creates a new ServerSocket object on a specific port (for example 9090). The socket object is considered to be successfully created only if the server binds to the port. This port can be configured by the administrator so that we can assure that the port is always available for the server only. The server now accepts a connection from the client. The server assigns a thread to each client that attempts to connect to it;

```
listenSocket = new ServerSocket();
listenSocket.setReuseAddress(true);
listenSocket.bind(new
InetSocketAddress(port));
```

```
serverMessageArea.append(    "Server
running on " + host + ", port " +
port+"\n" );
```

Step 3 – Process the request received from the client: A ConnectionHandler object is created. This object will process the requests sent and received from the client. The server thread that is assigned to the client will transmit the query to the Analyze module of the DiaSys Application;

```
try {
   ServerSocket server =
   new ServerSocket(port);
   while(true){
       System.out.println("Waiting
       for clients on port ...");
       Socket client =
       server.accept();
       ConnectionHandler handler =
       new ConnectionHandler(client);
       handler.start();
   }
}
catch(Exception ex) {
...
}
```

Step 4 – Compute an answer – The DiaSys Application computes an answer to the client request. The answer is transmitted from the "Answer Generation Module" to the server;

Step 5 – Send an answer to the client – The server sends the client a message that contains the answer to the question that the client requested.

Step 6 – Terminate the connection;

```
if(conected == true){
    try{
        dataOutputStream.close();
         inDataStream.close();
         connection.close();
         listenSocket.close();
         listenSocket = null;

    }
    catch(IOException e){
 System.err.print(e.getMessage());
     }
     conected = false;
 }
 ...
```

The administration of the conditional schema and of the knowledge pieces have remained on the server and the graphical interfaces did not change. Additionally, a graphical interface was created for the server. The interface is shown in Fig. 5.

B. Client side application

In this section we describe the actions of the client. We "moved" all the end user functions onto a client application.

An example of the variables used to work with the client is:

```
 private Socket connection;
 private InputStream inStream;
 private DataInputStream inDataStream;
 private OutputStream outStream;
 private DataOutputStream
dataOutputStream;
 private String defaultHost;
 private int defaultPort;
 private String host;
 private int port;
 private String message;
```

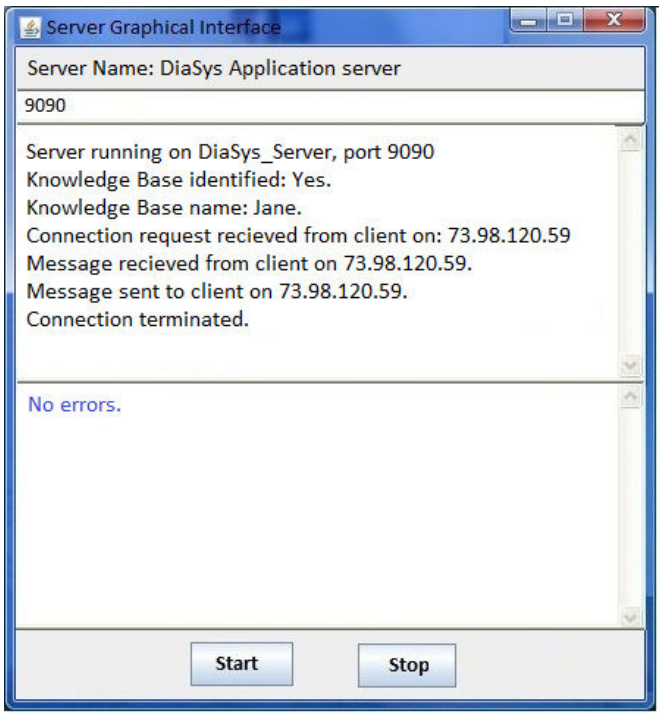

Fig. 5 – Server Graphical Interface

The actions performed by the client are described in several steps:

Step 1 – Initiation of the client application: The client is started by the end user by accessing the end user graphical interface which is installed on the station that the user owns;

```
public DiaSysClient(String
server, String portString){
   ...
   }
```

Step 2 – Establish a connection with the server: the client sends a connection request to the server. The server allocates a socket that the client will use;

```
try {
   InetAddress here =
   InetAddress.getLocalHost();
   host = here.getHostName ();
}
```

```
catch (UnknownHostException e) {}
port = defaultPort;
try {
   connection = new Socket(
   InetAddress.getByName(host),po
   rt );
   if (connection.isConnected()==
   false)
        return false;
   outStream =
   connection.getOutputStream();
   dataOutputStream= new
   DataOutputStream(outStream);
   inStream =
   connection.getInputStream();
   inDataStream = new
   DataInputStream(inStream);
}
catch(IOException e){
   return false;
}
return true;
```

Step 3 – Communicate with the server: The client sends a message to the server. Part of this message is the question that the end user has asked. The server will compute the answer to the end user inquiry and transmit a message that contains it to the client.

```
...
try{
    if (dataOutputStream == null)
    return;
    dataOutputStream.writeUTF(quer
    y);
    }
catch(NullPointerException ex){
    System.out.println(ex.getMessa
    ge());
    return;
}
catch(IOException e){
    System.out.println(e.getMessag
    e());
     return;
}
...
```

Step 4 – Display the message received from the server: The client extracts the answer from the message sent by the server. The message is displayed in the graphical end user interface.

```
try{
    answer = inDataStream.readUTF();
    return;
}
catch(IOException e){

System.out.println(e.getMessage());
 return;
}
```

Step 5 – Terminate the connection with the server: The connection will be terminated when the end user closes the interface or after 5 minutes of inactivity (the end user does not use the client application for 5 minutes);

```
try{
    dataOutputStream.close();
    inDataStream.close();
    connection.close();
}
catch(IOException e){
    System.err.println("Error on
    terminsting the connection
    with the server " +
    e.getMessage());
}
```

The interface of the client application is exactly the one that was used by the end user prior to the evolution of the system.

The user must enter the question in the text field and press the "Send" button. The client will send the question to the server while in the end user interface the "Send" button will be deactivated. After receiving the answer to the question the client will display the answer in the text area. To ask another question the user must press the "Reset" button.

An example of the end user interface is shown below in Fig. 6.

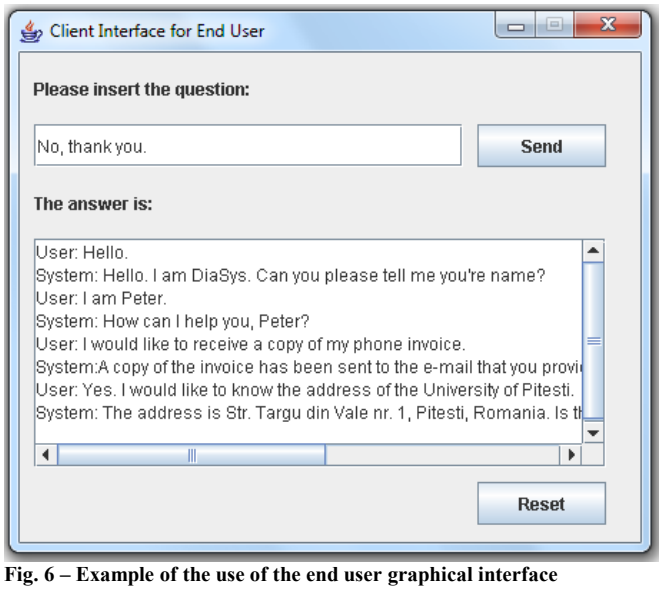

Fig. 6 – Example of the use of the end user graphical interface

V. Conclusion and Future Work

In this paper we described in a few words the functions of a Dialogue System Application. The inference engine and the knowledge base are built taking into account a knowledge representation method named conditional knowledge. The user can interrogate the system only by sentences concerning the declarative facts. In this article the DiaSys is adapted to a client-server system. We presented step by step the client and the server tasks. The server and the client are presented step by step.

An interesting extension of this study refers to the case when the user asks the system in the area of procedural

knowledge. In this way we are led to the idea to add a way to explain how the system obtained the conclusion. This can be named *Module of Explanation*. In this way our system can be used more successfully in automatic training. We intend to extend the dialogue system based on conditional knowledge. In a future work we exemplify the use of such a system in automatic training, to build health systems consultancy and systems for diagnosing the malfunctions of a device.

ACKNOWLEDGMENT

The author Cristina Zamfir has produced this article under the project "Supporting young Ph.D students with frequency by providing doctoral fellowships", co-financed from the EUROPEAN SOCIAL FUND through the Sectorial Operational Program Development of Human Resources.

REFERENCES

[1] N.Tandareanu, M.Colhon, Conditional graphs generated by conditional schemas, Annals of the University of Craiova, Mathematics and Computer Science Series, Vol.36, No.1, p.1-11, 2009

[2] N.Tandareanu, M.Colhon, The Inference Mechanism in Conditional Schemas, Annals of the University of Craiova, Mathematics and Computer Science Series, Vol.37, No.1, 2010

[3] N.Tandareanu, M.Colhon, C.Zamfir, A Spoken Question Answering System Based on Conditional Knowledge, The 15th WSEAS International Conference on COMPUTERS, Julay 15-17 2010, Corfu Island, Greece, p220-225, ISSN: 1792-4251

[4] DiaSys - A Dialogue System based on Conditional Knowledge, Annals of the University of Craiova, Mathematics and Computer Science Series, Vol.37, p.78-91, 2010

[5] N. Tandareanu, Proving the existence of labeled Stratified Graphs, Annals of the University of Craiova, Mathematics and Computer Science Series, Vol. XXVII, (2000), 81-92

[6] N. Tandareanu, M. Ghindeanu, Image Synthesis from Natural Language Description, Research Notes in Artificial Intelligence and Digital Communications,Vol.103, 3nd Romanian Conference on Artificial Intelligence and Digital Communications, Craiova, June 2003, p.82-96

[7] N. Tandareanu, Collaborations between distinguished representatives for labeled stratified graphs, Annals of the University of Craiova, Mathematics and Computer Science Series, Vol. 30(2), 2003, p.184-192

[8] N. Tandareanu, Intuitive Aspects of the Semantic Computations in KBO, Research Notes in Artificial Intelligence and Digital Communications 101, 1st National Conference of RCAI, p.1-8, June 2001

[9] N. Tandareanu, Knowledge Representation by Labeled Stratified Graphs, The 8th World Multi-Conference on Systemics, Cybernetics and Informatics, July 18-21, 2004, Orlando, Florida, USA, Vol. V: Computer Science and Engineering, p.345-350, 2004

[10] N.Tandareanu, Semantic Schemas and Applications in Logical representation of Knowledge, Proceedings of the 10th International Conference on Cybernetics and Information Technologies, Systems and Applications (CITSA2004), July 21-25, Orlando, Florida, USA, Vol.III, 82-87, 2004

[11] N. Tandareanu and M. Ghindeanu, Hierarchical Reasoning Based on Stratified Graphs. Application in Image Synthesis., 15th International Workshop on Database and Expert Systems Applications, Proceedings of DEXA2004, Zaragoza, IEEE Computer Society, Los Alamitos California, p.498-502, 2004

[12] N. Tandareanu, Distinguished Representatives for Equivalent Labeled Stratified Graphs and Applications, Discrete Applied Mathematics, Vol. 144/1-2, p.183-208, 2004

[13] N. Tandareanu, An Overview of Stratified Graphs and Their Applications, Research Notes in Artificial Intelligence and Digital Communications, Vol.105, 5th International Conference on Artificial Intelligence and Digital Communications, (2005), 111-120

[14] N. Tandareanu, A recursive method to compute the layers of a stratified graph, Journal of Multi-Valued Logic and Soft Computing, Vol. 12, p.355-363, 2006

[15] Faraj A. El-Mouadib, Zakaria S. Zubi,Ahmed A. Almagrous, Irdess S. El-Feghi, Generic Interactive Natural Language Interface to Databases (GINLIDB), International Journa of Computers, Issue 3, Volume 3, 301-309, 2009

[16] S. Bringsjord, M. Clark, and J. Taylor, "Sophisticated Knowledge Representation and Reasoning Requires Philosophy", 2009

[17] E.Agichtei, L.Gravano, Snowball: Extracting Relations from Large Plain-Text Collections. Proceeding of the 5th ACM International Conference on Digital Libraries (DL'00). San Antonio, TX. June 2-7, 2000.

[18] E.Breck, J.Burger, L.Ferro, D.House, M.Light, Inderjeet Mani, A system called Qanda, Eighth Text REtrieval Conference (TREC-8), Gaithersburg, MD. November 17-19, 1999

[19] A.Fujii, T.Ishikawa, Organizing Encyclopedic Knowledge based on the Web and its Application to Question Answering, Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL-EACL 2001),Toulouse, France, July 6-11, 2001.

[20] S.Harabagiu, D.Moldovan, C.Clark, M.Bowden, A.Hickl, P.Wang, Employing Two Question Answering Systems in TREC-2005, In: Text Retrieval Conference (TREC-14),Gaithersburg, Maryland (2005)

[21] D.Moldovan, M.Bowden, M.Tatu, A Temporally-Enhanced PowerAnswer in TREC 2006, In: Text Retrieval Conference (TREC-15), Gaithersburg, Maryland (2006)

[22] D.I.Moldovan, C.Clark, S.M.Harabagiu, D.Hodges, COGEX: A semantically and contextually enriched logic prover for question answering. J. Applied Logic 5(1): 49-69, 2007

[23] Harksoo Kim, Kyungsun Kim, Gary Geunbae Lee, Jungyun Seo:-MAYA, a fast Question answering system based on a predictive answer indexer, Proceedings of the workshop on Open-domain question answering - Volume 12, Toulouse, France, p.1 - 8, 2001.

[24] B.Katz, Using English for Indexing and Retrieving, in Artificial Intelligence at MIT: Expanding Frontiers, v. 1, Cambridge, Massachusetts, 1990.