Web-based Multi-Agent Medical and Control System

Moe Myint Myint, Dr. Soe Hay Mar
University of Computer Studies, Hpa-an
mss.moemyintmyint@gmail.com

Abstract

It is demonstrated as medical diagnosis agents of medical centers located in different areas. Agents are provided for users who want to know a related syndrome (disease) according to the users' input symptoms by using forward chaining method. Another approach is showed by using backward chaining method to help users for searching the doctors' information that fix with users' related syndrome (disease). The agents can be coordinate and cooperate with each other to search the related syndrome and the doctor information. As the expert knowledge required building the medical diagnosis multi-agent system, medical results are exploited. Examining the reasons of failure for paying the doctor information with the related syndrome, the system has realized that necessary measures should be taken in order to establish a semantic interoperability environment to be able to communicate with other medical diagnosis agents of other medical centers that may be located in different areas. Due to the nature of the problem which necessitates having autonomous entities dealing with heterogeneous distributed resources, the system has been built as a multi-agent system. The agent creation benchmark operates with basic Java Agent Development Framework (JADE) agents using rule based reasoning.

Keywords: Multi-agent, Rule Based Reasoning, Forward Chaining, Backward Chaining.

1. Introduction

The system has been developed as rule-based multi-agent system that provides medical services to users through a remote terminal connected to Internet (e.g. a portable PC, a PDA or a mobile phone) [3]. Each agent contains knowledge as rules that can predict the user's felling syndrome. When the user have been chosen symptoms that described on the user interface, the agent search the related syndrome using forward chaining methods. If the agent cannot search the syndrome that related with the user's choosing symptoms, it sends it's symptoms to other agents to search specific syndrome. Then it provides to connect other medical centers and the user. Then the system approaches to backward chaining method. In backward chaining method, the agent has been decided about what syndrome is occurred in its user. The agent requests to the user to insert the facts of the doctor that the user wants (e.g. name, location, time). Then the agent searches the doctor information that fix with the end-user desires, the end-user syndrome and who is good for paying treatment in the field of the end-user syndrome (specialty) in its knowledge repository. If the agent cannot search in its knowledge repository, it connects to other medical diagnosis agents to complete its goal (the doctor information that is the user's want).

The aim of this work was to develop a framework capable of supporting the decision making process in complex real-world domains, such as environmental, industrial or medical domains using a Multi-Agent approach with Rule-Based Reasoning. Medical Multi-Agent Software Systems are needed to reduce error in diagnosis and treatments. In this concern, Medical Multi-Agent System has been intending to develop JADE for sharing information through the web to be used in different medical centers' agent. In this system, Medical diagnosis multi-agent has been represented JADE (Java Agent Development Framework), a software framework to write the agent applications in compliance with the FIPA specifications for interoperable multi-agent systems [1].

2. Related Work

Intelligent Agents and Multi-agent systems are one of the most important emerging technologies in computer science today [5] [1]. The advent of Intelligent Multi-Agent Systems has brought together many disciplines in an effort to build distributed, intelligent and robust applications [7]. Intelligent Multi-Agent Systems have given us a new way to look at distributed systems and provided a path to more robust intelligent applications [5].

3. Rule-Based Reasoning (RBR)

Rule-Based Reasoning is an AI technique which tries to emulate the human reasoning and problem solving capabilities. They model how a human expert analyzes a particular situation by applying rules to the facts in order to reach a

conclusion. A RBR system represents information and searches for patterns in that information. Fact patterns are analyzed until either the goal succeeds or all of the rules are processed and the goal fails.

A rule-based description of metabolism can also be extremely fast and highly interactive. Truth maintenance mechanisms, which automatically deduce and retract conclusions when the underlying fact base changes, make the reasoning processes involved in rule-based simulations more robust and efficient. There are two main directions in Rule base reasoning: Forward and Backward Chaining Algorithms. In MAS, and an agent should be share the work between forward and backward reasoning, limiting forward reasoning to the generation of facts that will be solved by backward chaining.

The system can be decided syndrome by matching the end-user symptoms as rules. In the agent's knowledge repository, knowledge is represented as rules. There are many diseases (syndromes) which have many symptoms or only one symptom. The system defines symptoms that can be occurred syndromes as rules. So, symptoms can occur a little different or more different in syndromes. The system searches in its knowledge repository by matching rules using forward changing and backward chaining.

E.g. Enteric Fever and Vital Infection is the same in the symptom of fever is more than four weeks and Toxic symptom. But Dry coated tongue symptom, furred symptom, slow pulse rate symptom and Epistaxis symptom can occur Enteric Fever. Tachyardia symptom can occur Vital Infection. The rule for Enteric Fever is fever is more than four weeks symptom, Toxic symptom, Dry coated tongue symptom, furred symptom, slow pulse rate symptom and Epistaxis symptom. The rule for Vial Infection is fever is more than four weeks symptom, Toxic symptom and Tachycardia symptom. But the end-user chooses only the symptom of the fever is more than four weeks, the system decide the end-user's syndrome can be both Enteric Fever and Vital Infection. The system's decision is depend on the end-user's choosing symptoms.

3.1 Forward-Chaining (FW)

Forward Chaining is an example of the general concept of data-driven reasoning that is, reasoning in which the focus of the attention starts with the known data. It can be used within an agent to derive conclusion from incoming percepts, often without a specific query in mind. New facts can be added to the agenda to initiate new inferences. An example of this is a medical diagnosis in which the problem is to diagnose the underlying syndrome based on a set of symptoms (the working memory). A problem of this nature is solved using a forward-chaining, data-driven, system that compares data in the working memory against the conditions (IF parts) of the rules and determines which rules to fire [4].

In this system, forward chaining method is used when the system searching syndrome that is related with the user's choosing symptoms. In the case of searching doctor, the end user's criteria is little or the end user' criteria is not give to the system. On the above condition, the system may has many doctor information fix with the user's criteria or the user's syndrome (the user's criteria is not involved and the system decide doctors' information depend on the user's syndrome). The system is also used forward chaining. If the end user's wanted doctor is not exist in the user's requested agent, the system approaches to the backward chaining method for firing rule and completing the agents' work.

3.1.1 Example for Forward-Chaining (FW)

1. Data in working memory

User input = Choosing symptoms
(Data in knowledge-base **If** hills **then** Malariae Malaria, Falciparum Malaria
If hills and rigor **then** Malariae Malaria, Falciparum Malaria
If hills and rigor and on every fourth day **then** Malariae Malaria)
If the system is no rule to fire, it approaches to multi-agent technology for firing rule.
Trigger rule: Confirm for treatment
Conflict rule: rule Confirm for treatment (It is inserted by the user).
Fire rule: Confirm for treatment

2. Data in working memory

User input = Choosing symptoms
Syndrome = Decided from the system (Malariae Malaria)
Confirm treatment = yes
(Data in knowledge-base **If** syndrome = Malariae Malaria and treatment = yes **then** deciding doctor who can pay treatment to the end-user syndrome)
Trigger rule: Deciding Doctor (the system can get many doctors' information that can pay treatment to the end-user's syndrome)
Conflict set: rule Deciding Doctor by asking the end-user's criteria
Fire rule: Deciding Doctor

3. Data in working memory

User input = Choosing symptoms
Syndrome = Decided from the system (Malariae Malaria)
Confirm treatment = yes
Deciding Doctor = Doctors' information that fix with the user's criteria (e.g. Doctor's name = U Than Aye, location = Yangon hospital, start time for paying treatment = 10 am, end time for paying treatment = 3 pm, specialty = Malaria (specialty is

included as knowledge about Dr. U Than Aye is good for paying treatment in Malaria))

(Data in knowledge-base **If** syndrome = Malariae Malaria and treatment = yes and Doctor's name = U Than Aye and location = Yangon hospital and start time for paying treatment = 10 am and end time for paying treatment = 3 pm and specialty = Malaria **then** Finding Doctor)

If the system is no rule to fire, it approaches to backward chaining method for firing rule.
Trigger rule: Finding Doctor
Conflict set: rule Finding Doctor
Fire rule: Finding Doctor

4. Data in working memory

Doctor = Decision making from the system
Action = reply to the user
Trigger rule: Notify to user
Not added to Conflict set
Fire rule: Finding Doctor

3.1 Backward-Chaining (BW)

In other problem, a goal is specified and the AI must find a way to achieve that specified goal. For example, if there is an epidemic of a certain disease, this AI could presume a given individual had the syndrome and attempt to determine if its diagnosis is correct based on available information. A backward chaining, a goal-driven, system accomplishes this. To do this, the system looks for the action in the THEN clause of the rules that matches the specified goal. In the other words, it looks for the rules that can produce this goal. If a rule is found and fired, it takes each of that rule's conditions as goals and continues until either the available data satisfies all of the goals or there are no more rules that match [4].

In this system, backward chaining is started in the following conditions. One, the doctor information was already known by the end-user (e.g. the end user give detail doctor information to the system for searching). Two, the end user asked the system about the end user's doctor information (the end user may give to the system about the doctor name, the doctor location, the doctor time or the doctor's specialty). Three, the user's wanted doctor was not exist in the user's requested agent. The system use backward chaining method to find the end-user wanted doctor on the above conditions.

3.1.1 Example for Backward-Chaining (BW)

Query: Doctor = U Than Aye (inserted by the user)

1. Working memory = empty

Goal stack: (Malaria (specialty already exist in the system as knowledge), U Than Aye)

Antecedent: Treatment = yes

2. Working memory

Syndrome = Malaria (Decided from the system)
Treatment = yes
Antecedent: Finding Doctor = U Than Aye (inserted by the user)
Finding Doctor = Doctors' information that fix with the user's criteria (e.g. Doctor's name = U Than Aye, location = Yangon hospital, start time for paying treatment = 10 am, end time for paying treatment = 3 pm, specialty = Malaria (specialty is included as knowledge about Dr. U Than Aye is good for paying treatment in Malaria)
Goal stack: (Treatment, Finding Doctor (cooperating and negotiating with other agents to search the doctor information))

(Malaria, (name = U Than Aye, Location = Yangon Hospital, start time = 10 am, end time = 3pm, specialty = decided syndrome)

3. Working memory

Syndrome = Malaria (Decided from the system)
Finding Doctor = Doctors' information that fix with the user's criteria (e.g. Doctor's name = U Than Aye, location = Yangon hospital, start time for paying treatment = 10 am, end time for paying treatment = 3 pm, specialty = Malaria (specialty is included as knowledge about Dr. U Than Aye is good for paying treatment in Malaria)
Treatment = yes
Antecedent: Deciding Doctor = Doctor's name = U Than Aye, location = Yangon hospital, start time for paying treatment = 10 am, end time for paying treatment = 3 pm, specialty = Syndrome (Malaria)
This antecedent condition may be including other doctors' information that is closely related with the end-user's wants. (E.g. Dr. Hla Hla is good for paying treatment in Malaria and exists in Yangon hospital. Dr. Mya Mya is good for paying treatment in Malaria but she stays in Mandalay. The system records these doctors' information with priority. The system records firstly Dr. Hla Hla's information and then records Dr.Mya Mya information.)

4. Working Memory

Syndrome = Malaria
Treatment = yes
Deciding Doctor = Doctor Information that fix with the end-user's criteria and records in the system
User Criteria = Fire and pop off the rule Find-Doctor
User can get Doctor Information that already record in the system.

4. Multi-Agent System (MAS)

An intelligent agent may be defined as a computational process that can perform tasks

autonomously. It inhabits a complex and dynamic environment with which it may interact to accomplish a given set of goals [5]. A set of agents that communicate among themselves to solve problems by using cooperation, coordination and negotiation techniques composes a multi-agent system (MAS).

Multi-agent Systems (MAS) may be seen as a new methodology in distributed problem-solving via theorem proving, i.e. agent-based computing has been hailed as a distributed problem solving and/or a new revolution in software development and analysis. Indeed, agents are the focus of intense interest on many sub-fields of Computer Science, being used in a wide variety of applications, ranging from small systems to large, open and complex. Agents are not only a very promising technology, but are emerging as a new way of thinking , a conceptual paradigm for analyzing problems and for designing systems, for dealing with complexity, distribution and interactivity. It may even be seen as a new form of computing and intelligence. Many approaches consist of multiple agents [5].

5. Implementation of the Multi-Agent System

Multi-agent systems offer an implementation alternative that certainly fits user needs, because they have the following interesting properties:

-Modularity: Agents can have medical knowledge and many kinds of doctor information to pay treatments with different property and predict syndrome with related symptoms [5].

-Efficiency: The agent who is the user requested provides not only no doctor to pay treatment but also no knowledge of the syndrome for symptoms which choose from the user. It coordinates with other agents to find the doctor that fix the user's wants and the syndrome related with symptoms.

-Reliability: [5] A user can request to every agent. So every agent can be sender to request to other agents and they have their own knowledge to solve the user's health problem. When any agent is failed, the user can work with their requested agent.

-Flexibility: Agents can perform user's requirement and which way user choose (to take treatment or not). When the user requires taking treatment, the agent can search with other negotiated agents to satisfy the user's wants. Otherwise, the agent can eliminate its action [5].

-Existence of a standard: the FIPA (foundation for Intelligent Physical Agents) is a non-profit foundation based on Geneve (Switzerland). Its main mission is to establish the rules that have to govern the design and implementation of MAS in order to achieve interoperability among systems [5].

-Existence of development tools: JADE (Java Agent Development Environment) is a programming tool that contains a set of JAVA libraries that facilitate the development of the FIPA-compliant in the multi-agent system. A JADE plug-in that provides certain security mechanisms, called JADE, has been released [5].

5.1 Java Agent Development Environment (JADE)

JADE (Java Agent Development Environment) is a software framework to make easy the development of agent applications in compliance with the FIPA specifications for interoperable intelligent multi-agent systems. JADE is an Open Source project, and the complete system can be downloaded from JADE Home Page [11]. The goal of JADE is to simplify development while ensuring standard compliance through a comprehensive set of system services and agents.

JADE (Java Agent Development Framework) is a software framework to make easy the development of multi-agent applications in compliance with the FIPA specifications. JADE can then be considered a middle-ware that implements an efficient agent platform and supports the development of multi agent systems. JADE agent platform tries to keep high the performance of a distributed agent system implemented with the Java language. In particular, its communication architecture tries to offer flexible and efficient messaging, transparently choosing the best transport available and leveraging state-of-the-art distributed object technology embedded within Java runtime environment. JADE uses an agent model and Java implementation that allow good runtime efficiency, software reuse, agent mobility and the realization of different agent architectures.

5.2 JADE Agents' communication

JADE agents have communication capabilities. The adopted paradigm is asynchronous message passing. Each agent has a sort of mailbox (the agent message queue). JADE posts into the mailbox the messages sent by other agents. Whenever a message is posted in the message queue the receiving agent is notified. The programmer decides when to get the message from the message queue.

Messages exchanged by JADE agents have a format specified by the FIPA ACL (Agent Communication Language) Instances of the jade.lang.acl.ACLMessage class.

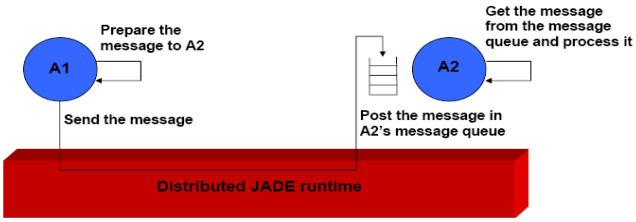

Figure 1: JADE Agent Communication Architecture

6. System Design

There are three different medical centers located in different areas. Each medical center has its own agent. Each agent has been created in web to pay medical services and doctor information to the end user. Each agent has its own knowledge to decide the user's syndrome. They are negotiated with each other not only when the user's desired doctor doesn't exist in the user's requested agent but also when the syndrome according with the related user's choice symptoms doesn't search in the requested agent. When a user chooses one of the agents, the user can work with its requested agent until to complete its goal.

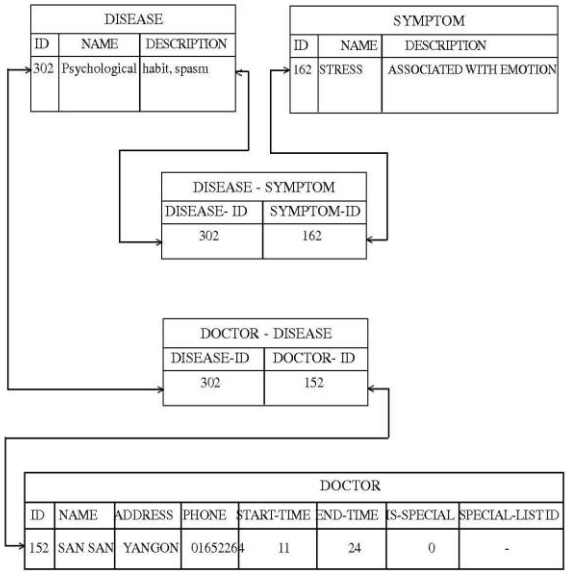

Figure 2: Table Design of the system in Oracle Database

There are three tables in Oracle Database table structure. The Symptom table includes medical names of Symptoms and description of these medical names. The Disease table includes medical names of Diseases and description of these medical names. The Doctor table includes NAME, ADDRESS, PHONE, START-TIME, END-TIME, SPECIALTY and SPECIAL-LIST ID. Firstly, Agent one accepts the user's symptoms. Then agent one predicts the disease using Forward chaining method but the related disease is not find in its database or knowledge repository. Agent one sends the user's symptoms to agent two using EJB and JADE techniques for completing the user's goals. Agent two searches the user's related disease and replies to agent one using EJB and JADE techniques.

When doctor information is not find in agent one, it sends syndrome to agent two to find doctors' information that pay a good treatment in this disease (specialty). Agent two searches doctors' information using backward chaining method in its database or knowledge repository. When agent two finds doctors' information, it sends results to agent one using EJB and JADE techniques.

The system is developed by using Java language (EJB-3.0, JSF-1.2 and JADE-1.4) and Oracle database that includes symptoms, related diseases (syndrome) and doctors' information to treat the syndrome. In Business Logic tier, the system developed on Web Logic Application Server is handling the data inserting, deleting and updating in the database using the enterprise java bean (EJB). In presentation tier, the JSF backing bean handle the request data from the user and response to the user uses Java Server Faces (JSF). When agent communication is started, java agent development environment (JADE) can do agents' communication.

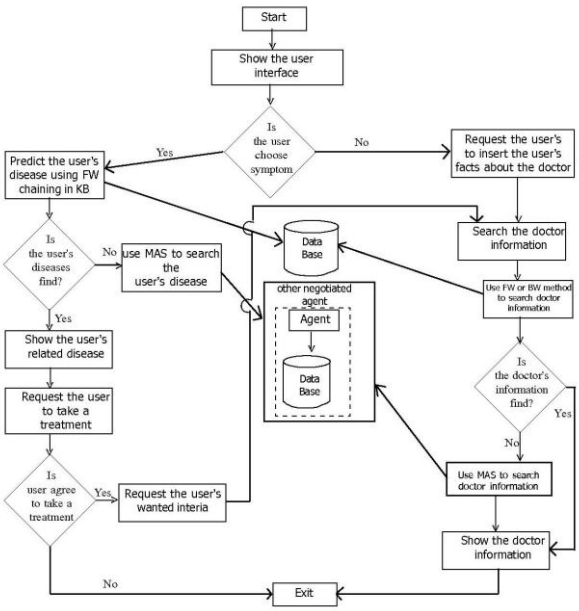

Figure 3: System Flow Diagram

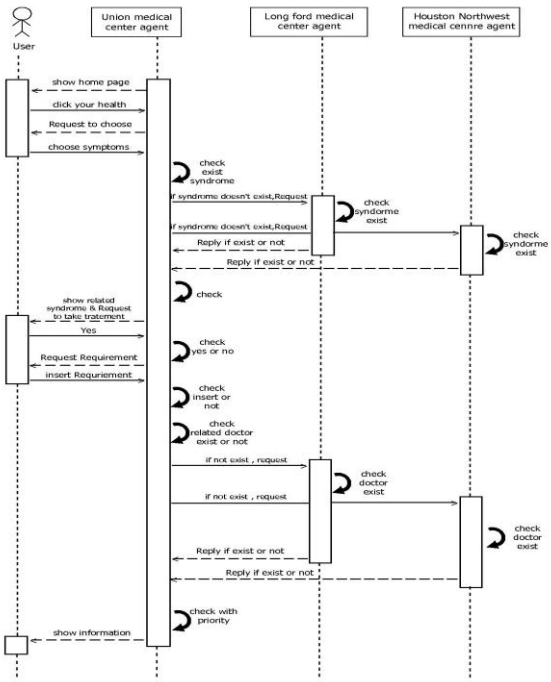

Figure 4: Sequence Diagram for Process

The system flow diagram refers to three agents' system flow. Each agent use forward chaining flow to predict syndrome and use backward chaining flow to search the user's desired doctor when the user's desired doctor does not exist in the users requested agent by negotiating with each other.

7. Experimental Result

This system successively starts 2000 agents (each agent started creates the next one) and records the execution time and memory consumption. The memory scale of the agent creation benchmark displays the range of 0MB to 263736 KB. The amount of agents is sufficiently large, thus memory is used due to the creation of agents, each of which incorporates an own knowledge base. Each agent uses memory between 5050 KB to 272356 KB. Agent who uses forward chaining method consumes time age about 1500 MS and 2000 MS locally. If the agent asks the other remote agents or other applications, it takes between 3000 MS and 5000 MS. Agent search using backward consume time about 16000 MS and 19000 MS locally but connection with other agents use 29000 MS and 4000 MS. In practice, a long time takes to use these agents can't reduce their performance for communication. More times to use this system more quickly to connect and search.

8. Conclusion

This paper presented an integrated framework for the rule-based medical multi-agent system by using the combination of multi-agent system and rule-based reasoning. The forward design and backward design complement each other to form an integrated design system. This thesis's presentation include establishing the combination of multi-agent and rule-based reasoning for the prediction of the end-user syndrome more correct than using only multi-agent or rule-based reasoning , establishing communication between agents provide user to get the most suitable doctor information for the end-user's syndrome and to know the end-user about possible related syndrome according to the end-user inserted symptoms and introducing the automated searching approach which is based on the forward and backward design chain by negotiating with other medical center agents. MAS can work their own task to complete. The rule-based MAS can work not only their own task to complete but also build a domain- independent framework for supporting decision-making in complex real-world domains. This system shows to extend a MAS tool capable of incorporating Rule-based Reasoning.

9. References

[1] B. Chen, H.C. Harry and P. Joe, *"Integrating mobile agent technology with multi-agent systems for distributed traffic detection and management systems"*, Department of Mechanical Engineering and Department of Electrical & Computer Engineering, Michigan Technological University, USA.

[2] Diplomarbeit, *"Evaluation and Implementation of Match Algorithms for Rule-based Multi-Agent Systems using the Example of Jadex"*, Master Thesis, Hamburg University,01agun@informatik.uni-hamburg.de, Studiengang Informatics, Matrikelnummer: 5318142, Fachsemester 16.

[3] F.Pascal, D.D Sophine, M.F Jean, G.Jonathan, *"Performance Analysis of Multi-Behavior Agents for Supply Chain Planning"*, CIRRELT-2008-43, Interuniversity Research Center on Enterprise Networks, Logistics and Transportation.

[4] F.H Janes, "*Methods of Rule-based System",* Rule based system and Identification Trees, AI article writing contest, http://ai-depot.com/contest.

[5] M.Antio, S.David, I.David, *"Security measures in a medical multi-agent system"*, Research Group on AI, Multi-Agent systems Group, Computer Science and Mathematics Department, School of Electrical and Computer Engineering, University of Rovirai Virgili.

[6] M.Antonio, *"Medical application of Multi-Agent Systems"*, Computer Science & Mathematics Department, University of Rovirai Virgili, Spain.

[7] N.K Kazuyoshi, Y.G daisuke, N.Y Fumiyo, T.H Muneo, *"The Medical Diagnosis Support System with Intelligent Multi-agent Techniques by Performance Differential Difference",* fifth International Workshop on Computational Intelligence & Applications, Toin University of Yokohama, email: nakano@intlab.toin.ac, jp.2009.

[8] P.S Mamimdar, "*Towards a formal theory of communication for multi-agent system"*, Austin TX 78712-1188, Department of computer science, university of Taxas, USA, 1992.

[9] S.Najib, *"Application of Backward Chaining Method to Computer Forensic"*, Communication of the IBIMA Volume 6, Hofstra University, Hempstead New York, najib.saylani@hofstra.edu, 2008.

[10] F. Bellifemine1, A. Poggi, G. Rimassa2, CSELT S.p.A, G. R. Romoli, *"Developing Multi-agent Systems with JADE"*, Dipartimento di Ingegneria dell'Informazione, University of Parma, Parco Area delle Scienze, Parma, Italy,