Impact of Multi-Agents in Hospital Environment

Mijal Mistry
Assistant Professor
Institute of Science & Technology for Advanced studies and Research,
Sardar Patel University

Dr. Dipti Shah
Associate Professor
G.H.Patel Post Gradutate Department of Computer Science & Technology
Sardar Patel University

ABSTRACT
This paper presents the architecture of the hospital system with the help of the Jade platform. It gives the idea about different agents used in hospital and how the communication occurred between them and how to manage different agents. Multi Agent System (MAS) provides an efficient way for communicating agents and it is decentralized. Prototype model is developed for the study purpose and shows the impact of the agent's platform and how it is used to solve the complex problems arise into the hospital domain. The paper discusses prototype model of Multi Agents in hospital environment and its impact.

Keywords
Hospital Domain, Jade, Multi Agents, Health Care Domain

1. INTRODUCTION
The term 'agent', or software agent, has found its way into a number of technologies and has been widely used, for example, in artificial intelligence, databases, operating systems and computer networks literature. Therefore, an agent is autonomous, because it operates without the direct intervention of humans or others and has control over its actions and internal state. An agent is social, because it cooperates with humans or other agents in order to achieve its tasks. An agent is reactive, because it perceives its environment and responds in a timely fashion to changes that occur in the environment. An agent is proactive, because it does not simply act in response to its environment but is able to exhibit goal-directed behavior by taking initiative.[14]

2. AGENT MANAGEMENT
The agent management reference model consists of the components depicted in Figure 1.[14]

Agent Platform (AP): This provides the physical infrastructure in which agents are deployed. The AP consists of the machines, operating systems, FIPA (Foundation for Intelligent Physical Agents) agent management components (described below), the agents themselves and any additional support software.

The specific internal design of an AP is left to the developers of an agent system and is not a subject of FIPA standardization beyond the components discussed here. As a single AP may be spread across multiple computers, resident agents do not have to be co-located on the same host.

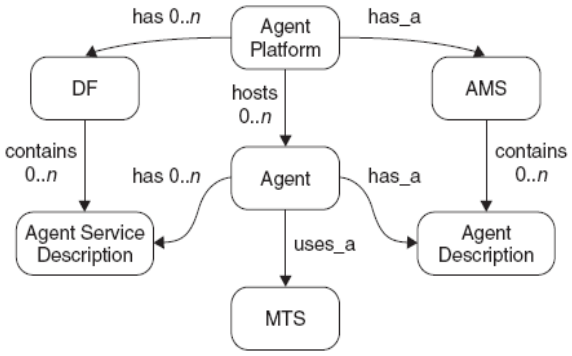


Figure 1 Agent Management Reference Model

The specific internal design of an AP is left to the developers of an agent system and is not a subject of FIPA standardization beyond the components discussed here. As a single AP may be spread across multiple computers, resident agents do not have to be co-located on the same host.

Agent: An agent is a computational process that inhabits an AP and typically offers one or more computational services that can be published as a service description. The particular design of these services, otherwise known as capabilities, is not the concern of FIPA, which only mandates the structure and encoding of messages used to exchange information between agents (and other third party technologies if FIPA compliant).

Directory Facilitator (DF): The DF is an optional component of an AP providing yellow pages services to other agents. It maintains an accurate, complete and timely list of agents and must provide the most current information about agents in its directory on a non-discriminatory basis to all authorized agents. An AP may support any number of DFs which may register with one another to form federations.[14]

Agent Management System (AMS): The AMS is a mandatory component of an AP and is responsible for managing the operation of an AP, such as the creation and deletion of agents, and overseeing the migration of agents to and from the AP. Each agent must register with an AMS in order to obtain an AID which is then retained by the AMS as a directory of all agents present within the AP and their current state (e.g. active, suspended or waiting). Agent descriptions can be later modified under restriction of authorization by the AMS. The life of an agent with an AP terminates with its deregistration from the AMS.[14]

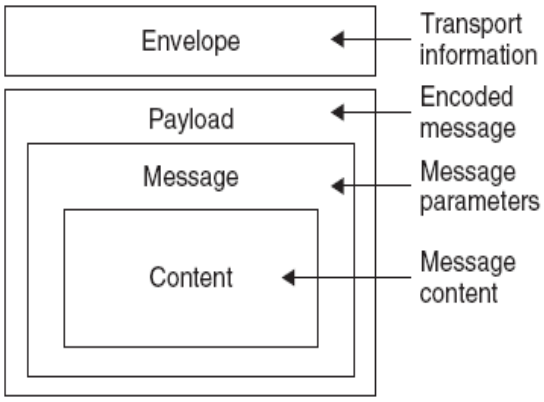


Figure 2 FIPA message structure

After deregistration, the AID of that agent can be removed by the directory and can be made available to other agents who should request it. Agent descriptions can also be *searched* for within the AMS, and the AMS is the custodian of the AP description that can be retrieved by requesting the action get-description.

Message Transport Service (MTS): The MTS is a service provided by an AP to transport (Foundation for Intelligent Physical Agents) FIPA ACL (Agent Communication Language) messages between agents on any given AP and between agents on different APs. The general structure of a FIPA-compliant message is depicted in Figure 2.

Agents in a multi-agent system (MAS) must be able to interact and communicate with each other. This usually requires a common language, an Agent Communication Language, or ACL. Much work has been done in developing ACLs that are declarative, syntactically simple, and readable by people. KQML [1] and FIPA-ACL [2] are two of the most widely used ACLs in multi-agent systems. These languages have been very successful in facilitating the communication and coordination of software agents in a variety of domains including organizational decision making [3], [4]; financial management [5]; and even aircraft maintenance [6].

3. COMMUNICATION WITHIN AGENT BASED ARCHITECTURE

The agents need to correspond between themselves, as well as with the services of their environment. There are numerous respond mechanisms are available like: message swapping and transferring, methods invocation. There are also standardized inter-agent communication languages like KQML (Knowledge Query Management Language), ACL (Agent Communication Language) is also used for further communication.

Another language called ACL (Agent Communication Language) [8] is a successor of KQML and it is full with rich semantics. This language was proposed by FIPA [8], ACL aims to standardize communication between agents.

In our approach, we use ACL for communication between agents and the contents of the messages we use XPDL. The use of ACL and XPDL in agent communication allows getting interoperability by eliminating the problem of various exchanges among the different doctors in the agent platform environment.

The exchanged messages
There is standard syntax for messages, which is supplied bi FIPA [8]. These messages are based on the theory of the act do dialogue, which is the result of the linguistic study of human communication [8]. Basically, this is used for achieving the result from the language. In the FIPA-ACL, There are no specific language exist for the description of the values of the messages. Few languages can be used for the description of the contents of the message such as KIF (Knowledge Interchange Format), Semantic language (SL), prologue and XML (Extensible Mark-up Language) XPDL (based in XML language) etc.[13] This language is also used for the specification and understanding of the contents of messages between agents. The messages swapping in our architecture are described in FIPA-ACL/XPDL. [5]

The use for XPDL for the contents of communications among agents gives the possibility to display of messages into different Web browsers and also allows to integrate the system with other web-based applications.

4. IMPLEMENT MULTI AGENT SYSTEM WITH JADE

JADE (Java Agent Development framework) is the multi-agent platform; developed in Java by CSELT (Research Groups Telecom, Italy). It provides a FIPA (Foundation for Intelligent Physical Agents) compliant environment and implementation of multi agent system [9]. JADE includes two basic parts: an agent platform and software package. It also provides several features which are mentioned following [9]:

A scattered agent platform: the agent platform can be spilt among several hosts (provided they can be connected via RMI). In JADE, agents can be implemented as threads and the life of that agent is within Agent containers which provides runtime support for the agent implementation.[16]

A JADE platform is composed of agent containers that can be distributed over the network. Agents live in containers which are the Java process that provides the JADE run-time and all the services needed for hosting and executing agents. There is a special container, called the *main container*, which represents the bootstrap point of a platform: it is the first container to be launched and all other containers must join to a main container by registering with it. The UML diagram in Figure 3 schematizes the relationships between the main architectural elements of JADE.

In Jade, an agent is an instance of the Java class defined by the programmer. This class itself is an extension of the basic Agent class (included jade.core). It implies the inheritance of the set of basic methods to implement the personalized behavior of the agent. The agent is implemented using multitasking; where the tasks (behaviors) are executed concurrently. Every functionality supplied by an agent must be implemented in one or several behaviors. [16]

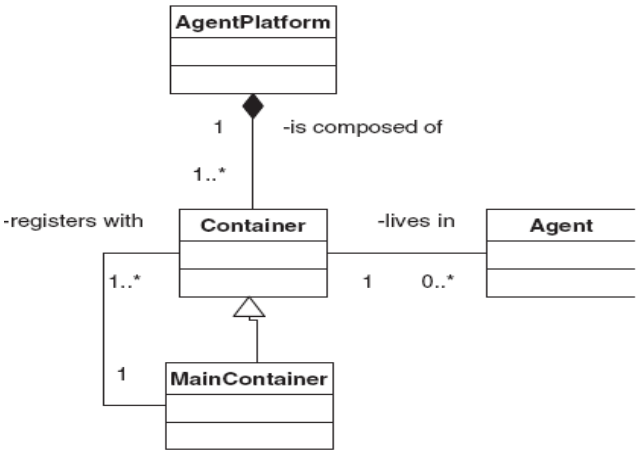


Figure 3 Relationship between the main architectural elements

5. PROTOTYPE MODEL

To study the real impact of the multi agents a prototype model has been developed. Following are the different classes which are used for implementing the prototype model.

- Agent_Schedule
- Agent_Doctor
- Agent_Search

The Agent_Schedule class is used to make the appointment with the doctor. The Agent_Doctor class is used to retrieve the available doctors for the appointment. It gives the list of doctors who are ready to take appointments with patients. The Agent_Search class searches into the already made appointments and check the free slots for schedule.

The following Java code represents the implementation of the Schedule class and the Search agent class in hospital environment. These classes are extensions of the basic Agent class respectively of the Schedule Agent class , Doctor agent and Search agent.

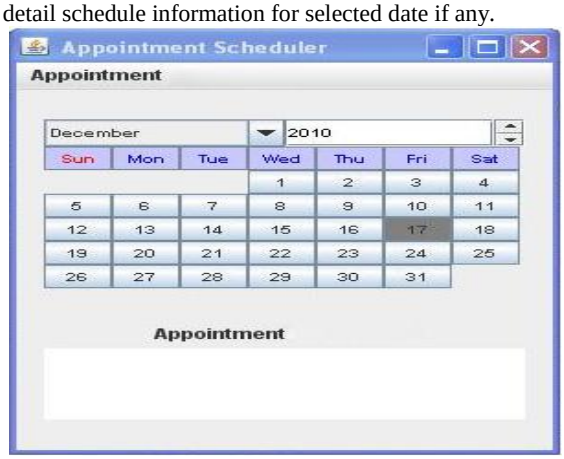
```
public class Agent_Doctor extends Agent
{
protected void setup ()
{
addBehaviour (new Simple Behaviour (this))
{
// Processing
}
}

public class Agent_Schedule extends Agent
{
protected void setup ()
{
addBehaviour (new Simple Behaviour (this))
{
// Processing
}
}

public class Agent_Search extends Agent
{
class reception extends simpleBehaviour {
// Processing
}
Public reception (Agent a) {super (a);}
Protected void setup()
{
reception mybehaviour = new reception(this);
addBehaviour (mybehaviour);
}
}
```

The Agent_Schedule class shown above, represents a type of agent exactly as a normal Java class represents a type of object. Several instances of the Agent_Schedule class can be launched at run-time. Unlike normal Java objects, which are handled by their references, an agent is always instantiated by the JADE run-time and its reference is never disclosed outside the agent itself (unless of course the agent does that explicitly). Agents never interact through method calls but rather by exchanging asynchronous messages. Agent_Doctor class provides the available doctors. It shows the available doctors and how many appointments left for the doctor. Agent_Search class will also use the Agent_Doctor class for searching the available doctors and showing them to the user.

The setup() method is intended to include agent initializations. Examples of typical operations that an agent performs in its setup() method are: showing a GUI, opening a connection to a database, registering the services and starting the initial behaviors. It is good practice not to define any constructor in an agent class and to perform all initializations inside the setup() method. This is because at construction time the agent is not yet linked to the underlying JADE run-time and thus some of the methods inherited from the Agent class may not work properly.

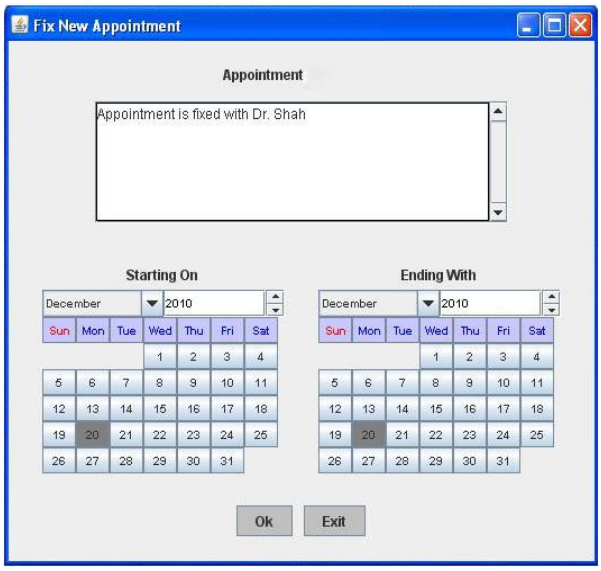
From the above code snippet we can get the idea that we have developed classes for Schedule, Doctor and Search which are extended from Agent class. All classes can communicate with each other. Fig. 4 shows how it can be done. From the Fig. 4, one can get the idea that the system is hosted into different clients and they are inter-connected using network.

This paper shows the idea of scheduling the appointment with doctor by creating the agents and passing necessary information to the agents. Following figure shows the calendar for displaying the detail schedule information for selected date if any.

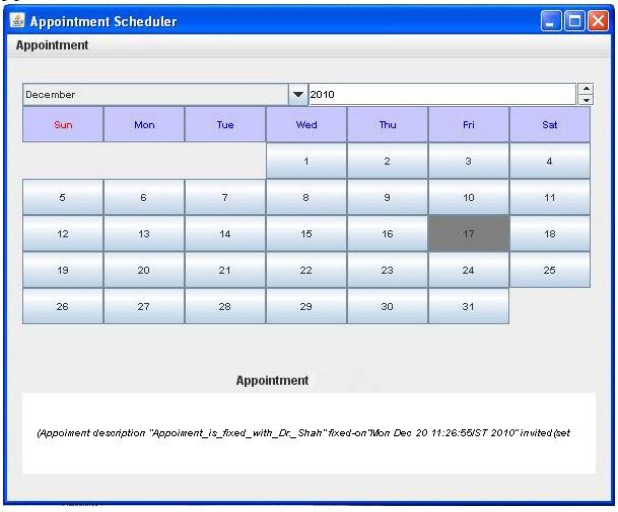


Here when user changes the date it will also change the description for the selected date and information into the appointment space.

By clicking on the Appointment button at the top of the form it will open another form which allows us to book the appointment with the doctor.



Here user can enter the necessary information for appointment. If we want to add some remarks or notes then also we can add using the appointment description. Once we done with description then select the appointment date, by clicking on OK it will assign the appointment. If you want to cancel the appointment just click on exit button and it will cancel the appointment and come out from it. Once appointment is created then this information will be shown on selected date from the main form. Here, we have used several inbuilt java apis to make the work smoother and faster and also jade platform for inheriting the different agents apis. For some getting good look and feel of the form we used java swing. Following fig. shows the information of the schedule appointment.



For eg. on the Fix Appointment form when user click on the OK button it will create the object of the Appointment class using following code:

```
Appointment a = new Appointment();
a.setDescription(textArea1.getText());
c = calendar1.getCalendar();
d = c.getTime();
a.setStartingOn(c.getTime());
c = calendar2.getCalendar();
a.setEndingWith(c.getTime());
```

There are several properties into the Appoinment class like setDescription, SetDate etc. One of the property is setDescription. Whatever information we have added into the information part will be assigned to the setDescription property of the Appointment class. Once all the statements above stated have been executed then the string will stored with modification of the message.

The Appointment class is used to store the information regarding the schedule and it will also display schedule information as showing in appointment schedule screen shot. It shows the information after fixing the appointment with the doctor. For storing and retrieving this information Appointment class has been used.

Agents have the potential to assist in a wide range of activities in hospital environments. They can maintain the autonomy of the collaborating participants, integrate disparate operating environments, coordinate distributed data, such as patient records held in different departments within a hospital or in several hospitals, clinics and surgeries [10, 11], and other organizations involved in health care such as insurance companies and government organizations [12]. In this context agent based approach is very useful for tackling such issues.

6. IMPACT OF MULTI AGENTS

By implementing the prototype system, we came across several interesting points for multi agents and its usefulness into developing the system into hospital environment. Here are the points which we came across while implementing the system:

- enhances overall system performance
- provides computational efficiency
- reliability
- extensibility
- robustness
- maintainability
- responsiveness
- flexibility
- reuse

[15] A multi-agent system (MAS) is a loosely coupled network of software agents that interact to solve problems that are beyond the individual capacities or knowledge of each problem solver.[15]

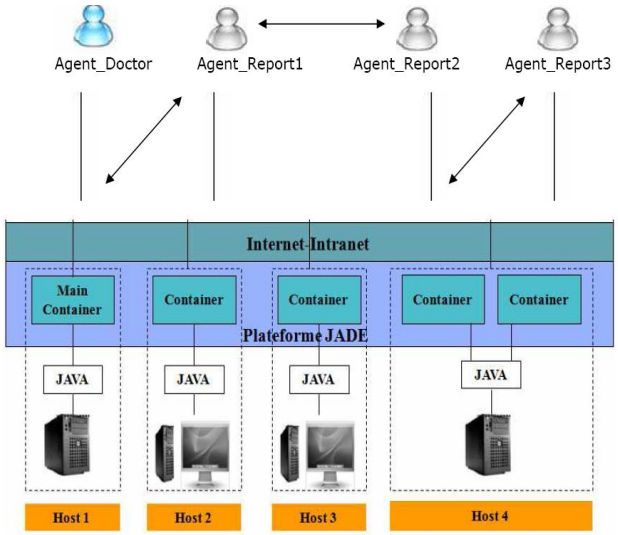


Figure 4 Implementation of MAS architecture in JADE

There are several reasons why using jade. JADE is a software platform that provides basic middleware-layer functionalities which are independent of the specific application and which simplify the realization of distributed applications that exploit the software agent abstraction (Wooldridge and Jennings, 1995). Jade has several easy to use customize core functionalities like:

- Full compliance with the FIPA specifications.
- A simple, yet effective, agent life-cycle management.
- Support for ontologies and content languages.
- An in-process interface for launching/controlling a platform and its distributed components from an external application.
- Integration with various Web-based technologies including JSP, Servlets, applets and Web service technology.
- Efficient transport of asynchronous messages via a location-transparent API.
- A set of graphical tools to support programmers when debugging and monitoring.

7. CONCLUSION

We have used the Multi Agents for implementing the system in JADE environment. Our domain is HealthCare and we have implemented three agents. One Agent is for booking the schedule, Second Agent will search the booked appointments, how many appointments have been scheduled, and Another Agent is doctor agent which gives the information regarding the doctors and its specialties. There are several other researches have been made into different domain using multi agents. We have seen several articles regarding workflow environments, in robotics, multi agents systems into academic environment, several knowledge base system using multi agents.

In this paper the attempt is made to make the system more flexible by using the MAS paradigm. It provides the more stability in the process. Moreover, its concepts are adequate to taking in charge adaptability in a workflow environment. Right now we have used three agents for communication for hospital. Right now it books the appointment after checking the availability of the doctors. Our future work will involve to achieve communication between more complex agents using MAS and others participants in external environment applications.

8. REFERENCES

[1] T. Finin, R. Fritzson, and R. McEntire, .KQML as an agent communication language,. in Proceedings of the 3rd International Conference on Information and Knowledge Management, November 1994.

[2] Foundation for intelligent physical agents. [Online]. Available: http://www._pa.org, 1997.

[3] K. Sycara, K. Decker, A. Pannu, M. Williamson, and D. Zeng, .Distributed intelligent agents,. IEEE Expert, December 1996.

[4] H. Chalupsky et al., .Electric elves: Agent technology for supporting human organizations,. AI Magazine, Summer 2002.

[5] K. Decker, K. Sycara, and D. Zeng, .Designing a multi-agent portfolio management system,. in Proceedings of the AAAI Workshop on Internet Information Systems, 1996.

[6] O. Shehory, G. Sukthankar, and K. Sycara, .Agent aided aircraft maintenance,. In Proceedings of Autonomous Agents '99, May 1999, pp. 306.312.

[7]. Ferber J, "les systeme multiagents: Vers une intelligence collective" InterEdition, 1997, 239-245

[8] FIPA; 1999, http://www.fipa.org/spec/FIPA98.html

[9] Fabio Bellifemine, Giovanni Caire, Tiziana Trucco ,« Jade Programmer's Guide », university of Parma 200-2003, http://jade.cselt.it.

[10] Fensel, D., Ontologies (2003), "Silver Bullet for Knowledge Management and Electronic Commerce", 2nd edition, Springer-Verlag, Berlin.

[11] Jonathan M. DiLeo, B.S. Thesis on ontological engineering and mapping in multiagent systems developement retrieved from www.stormingmedia.us/84/8408/A840804.html on 20th April 2009.

[12] The Application of Agent Technology to Health Care , AgentCities Working Group on Health Care By John L Nealon and Antonio Moreno

[13] The Workflow Management Coalition, Workflow Management Coalition Workflow Standard," Process Definition Interface -- XML Process Definition Language", Document Number WFMC-TC-1025, October 3, 2005, http://www.wfmc.org/standards/XPDL.htm

[14] Wiley.Developing.Multi.Agent.Systems.with.JADE by Fabio Bellifemine, Giovanni Caire, Dominic Greenwood

[15] Multi-Agent Systems - Carnegie Mellon University, http://www.cs.cmu.edu/~softagents/multi.html

[16] Implementation of Multi Agents System to Control Adaptability in Workflow Environment by Hamdane Mohmaed El-Kamel, Lezzar Fouzi, Boufenar Chaouki, Mili SeifEddine.