Journal of Software Engineering and Applications, 2012, 5, 33-40
http://dx.doi.org/10.4236/jsea.2012.51006 Published Online January 2012 (http://www.SciRP.org/journal/jsea)

Multi Agent Knowledge Management Architecture

Sheikh Amanur Rahman, Dipti Yadav, Prerna Agarwal, Pankaj Singh Bisht

Department of Computer Science & Engineering, Galgotias College of Engineering & Technology, Greater Noida, India.
Email: {sheikhamanur, diptiyadav2206, prerna115, pankaj.bisht33}@gmail.com

Received November 12th, 2011; revised December 13th, 2011; accepted December 29th, 2011

ABSTRACT

Nowadays, knowledge in Public Sector environment becomes very vast and increasing day by day at speedy pace. So, to handle and manage the knowledge becomes a tedious job, resulting into degrading the overall affectivity and productivity of the system. Hence, the need of effective architecture arises, which can increase the performance of disseminating knowledge in public sector. This results the implementation of knowledge management (KM) using Multi Agents (MA). Using Multi Agents reduces the time overhead for serving relevant knowledge to end users. The objective of this paper is to propose KM architecture using MA which will be helpful and effective in circulating knowledge to public sectors in a much better and easier manner, due to which it enhances the productivity and performance. The paper firstly, gives the understanding of literature on various knowledge management frameworks and tools for implementing Multi Agents. Then it proposes a MA enterprise knowledge management architecture (MAEKM), stating that how knowledge circulation will be done. At the end, using JADE framework, paper implements MAEKM architecture for public sector. The paper describes the necessity of implementing this architecture and its usefulness in disseminating knowledge in public sectors.

Keywords: Knowledge Management; Multi-Agent System; MAEKMA; Enterprise; Public Sector Unit (PSU); JADE

1. Introduction

Knowledge Management (KM) [1,2] is defined as to provide relevant information to the right people at right time. We are using Multi-Agent methodology using JADE (Java Agent Development) for implementing KM. An agent can be defined as a software and/or hardware component of system who accomplishes tasks on behalf of its user. Agents are reactive, autonomous and co-operative in nature. They have the ability of knowledge based reasoning.

1.1. KM Life Cycle

Nissen [3] proposed KM lifecycle model consisting of six phases: create, formalize, organize, distribute, apply and evolve. It shows as **Figure l**.

The knowledge management lifecycle starts with the create phase where new knowledge is created. Then the newly created knowledge is formalized and therefore stored in knowledge base according to the knowledge organizing mechanism. When knowledge is required, it can be searched and use relevant knowledge by accessing knowledge base. Finally knowledge is applied and further evolved into new knowledge. It also leads to further knowledge creation and completing the lifecycle.

1.2. Knowledge Management in Industries

Engineering can be considered as a knowledge-oriented industry. Even low weighted projects need thought, knowledge and skills from many sources which may include electronic media, documents and people. Various Engineering firms have been managing knowledge informally for years but the industry who is facing challenges today mean that most of the organizations now required a more structured and consistent approach to knowledge management [4]. Industries implementing projects are usually organized into stages with deadlines and there are various teams assigned to those projects. The problem is that capturing and reusing of knowledge learning from projects is measured to be difficult because teams are of dynamic in nature and may be dismissed before finishing point of the project and therefore gathered to the next project. These types of problems generally bound the information flow and can create obstacle to knowledge learning.

This paper outlines an approach to assessing and implementing Knowledge Management using Multi-Agents. It highlights the importance of aligning KM initiatives to the business goals.

The rest of the paper is organized as follows. Section 2 provides the literature review and current techniques and

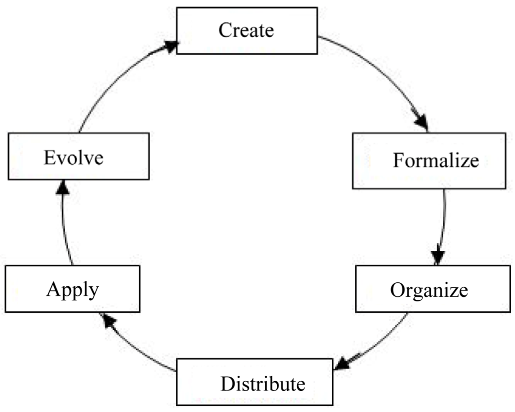


Figure 1. Knowledge management lifecycle.

research related to implementing multi-agent KMS. This section also discusses the various tools for implementing multi-agents and then shows the features of JADE when compared with others. Section 3 discusses the implementation of new multi-agent enterprise knowledge management (EKM) architecture for PSU which we have proposed and its specialty. It also discusses the motive of implementing the proposed architecture. In Section 4, we have provided the implementation details of the proposed architecture. In Section 5, we have shown the result and observations after implementing the proposed architecture. Finally, Section 6 summarizes the conclusion and future work anticipated in this direction.

2. Related Work

2.1. Skills of KM in Organization

Individual knowledge plays the important role in an organization. Individual knowledge management should be supported by the organizations by monitoring at knowledge worker's actions and move towards to manage their personal knowledge. Encouragement must be there for efficient KM behaviors in the organization. KM tool should help in enhancing the efficiency at work and should be able to help in managing time. KM tools consist of communities of practice by which people create networks of personal significant knowledge. KM within an organization can be empowered with coordination, collaboration, and cooperation.

2.2. Current Techniques Implementing KMS

Different techniques have been used to implement KMS. Intelligent agents are one of them [5]. Agents are proactive in nature as they can take the initiative at their own and complete their own goals. The autonomous behavior of the agents meets the required goal of this research because it can minimize the amount of work done by the employees when using a KM system. Another important issue is that agents can learn from their own experience. Consequently, agent systems are required to be more efficient because the agents get experience from their previous knowledge.

Different agent-based architectures have been proposed to support activities related to KM [6-10]. Some architecture has been designed for the development of KMS. However, most of them focus on a particular domain and can only be used under specific circumstances.

A lot of research and commercial organizations are involved in the realization of agent applications and a considerable number of agent construction tools have been realized [7]. Some of the most interesting are Cougaar, JACK, 3APL, and Agent Factory, JADE.

Features of Cougaar [12]:

- Cougaar framework is based on Java language which is used for distributed applications of agents.
- Multi-tier interaction model is used in Cougaar.
- Components interacts within agents with the help of local publish-subscribe mechanism.
- Message passing mechanism is used for communication between the agents.
- For loading the components dynamically, Cougaar uses a flexible component model.
- Agent relationship is dynamically negotiated by the method of hierarchical service discovery.
- Cougaar also enables the agents to be organized into communities.

Features of JACK [11]:

- JACK is a commercialized framework which is based on BDI agent realization.
- While dealing with agent plans and beliefs, JACK extends the Java by using syntactic constructs.
- JACK does not use language which is based on logic.
- Modularization is done well here through agent planning and agent team concept.

There are some important identifying features of JADE when comparing with other tools:

- JADE is identified with FIPA specification by its own because JADE is totally FIPA specification based.
- JADE provides special functionalities to make simple when developing multi-agent systems. There are very less restrictions on the user code. There is no need to understand any BDI architecture or use it by the user. Users just have to write Java code without required to learn any new special construct.
- JEE, JSE, and JME devices can be deployed by the JADE and therefore provides a homogeneous set of APIs in its runtime environment which does not depend on the underlying network and Java technology.

2.3. Jade Framework

JADE (Java Agent Development Environment) [3] is a software framework which is used for interoperable intelligent multi-agent systems by making the agent applications development easily in compliance with the FIPA specifications. List of features which are provided by the JADE to the agent programmer are as follows:

- JADE is an Agent Platform based on FIPA-compliance. It consists of AMS (Agent Management System), the default DF (Directory Facilitator), and the ACC (Agent Communication Channel). All these three agents are automatically activated at the agent platform start-up.
- Distributed agent platform: The agent platform can be split on several hosts. On each host only one Java Virtual Machine, is executed. Agents are implemented as one Java thread and for effective communication between agents, Java events are used on the same host. An agent can execute parallel tasks, and JADE schedules these tasks in a cooperative way.
- Several DFs (Directory Facilitator) can be started at run time in order to build multi-agent environments. Common facilitator advertised their services.
- Java API to send/receive messages to/from other agents; ACL messages are represented as ordinary Java objects.
- Lightweight transport of ACL messages inside the same agent platform, as messages are transferred encoded as Java objects, rather than strings, in order to avoid marshalling and unmarshalling procedures.
- Support for agent mobility within a JADE agent platform.
- Library to manage user-defined ontology and content languages.
- GUI is used to manage several agents and agent platforms from the same agent. The activity of each platform can be monitored and logged.

3. Proposed Work

3.1. Motive for Implementing MAEKM for Public Sector

Traditionally, in generalized public sectors, it is very difficult to handle and manage the organization's knowledge which results in:

- More time consuming for knowledge serving queries;
- Lack of appropriate knowledge;
- Degradation of overall quality;
- Lack of effective communication when dealing with customer knowledge queries.

Therefore we have decided to implement KM for public sector using Multi Agent System in a way such that to get rid of all the problems above and to produce the following result after implementation:

- Smooth communication between agents;
- Knowledge reusability by agent's past experience;
- Reducing time overhead for knowledge serving;
- Serve relevant knowledge at correct time;
- Provides best possible solutions.

We decided to take the concept of Multi Agent System instead of Single Agent System because:

- Single Agent has partial capabilities to solve a problem;
- Single Agent does the computation asynchronously.

3.2. MAEKM Architecture

The architecture consists of 3 layers as shown in **Figure 2**:

1) User GUI Layer: This is the top most layer where information is send and receive by the user. At this layer, user interacts with the system to retrieve, create, share, use the knowledge etc. Personal Assistant (PA) Agent serves to user and act as a interface between user and the system.

2) Agent Inter Communication Layer: This is the main layer of the architecture. At this layer, Multi-Agents remains active all the time. Agents communicate to each other regarding knowledge Processing, retrieval, creation etc and produce the result in lesser time.

3) Knowledge Repository Layer: This is the lower most layer. It contains organization's overall knowledge where all the information is stored.

3.2.1. Multi-Agents Descriptions

- Head Agent (HA): It handles and manages the all other agents which take part in some KM activity.
- Personal Assistant (PA) Agent: This agent serves to the users and therefore handles the user's queries regarding the knowledge processing. PA Agent collaborates with the other agents and act according to the user's queries.
- Knowledge Agent (KA): This agent processed the required query and produce the relevant information for the given request. It also manages and updates the Knowledge Repository.
- Knowledge Filter Agent (KFA): This agent filters the irrelevant data from the useful data which results in producing the accurate knowledge for the related request query.

3.2.2. Overall Implementation of PSU Knowledge Management

Based on MAEKM architecture, we are implementing PSU Knowledge Management using Multi Agent System as shown in **Figure 3**, to exchange knowledge with each other, in a way that preserves the knowledge, and therefore reaches to the knowledge seeker in a just in time. One of the main tasks of this implementation is to support and encourage collaboration and knowledge sharing. This implementation defines various services like billing information, payment services, complaint services, tender filling services, complaint services, customer support

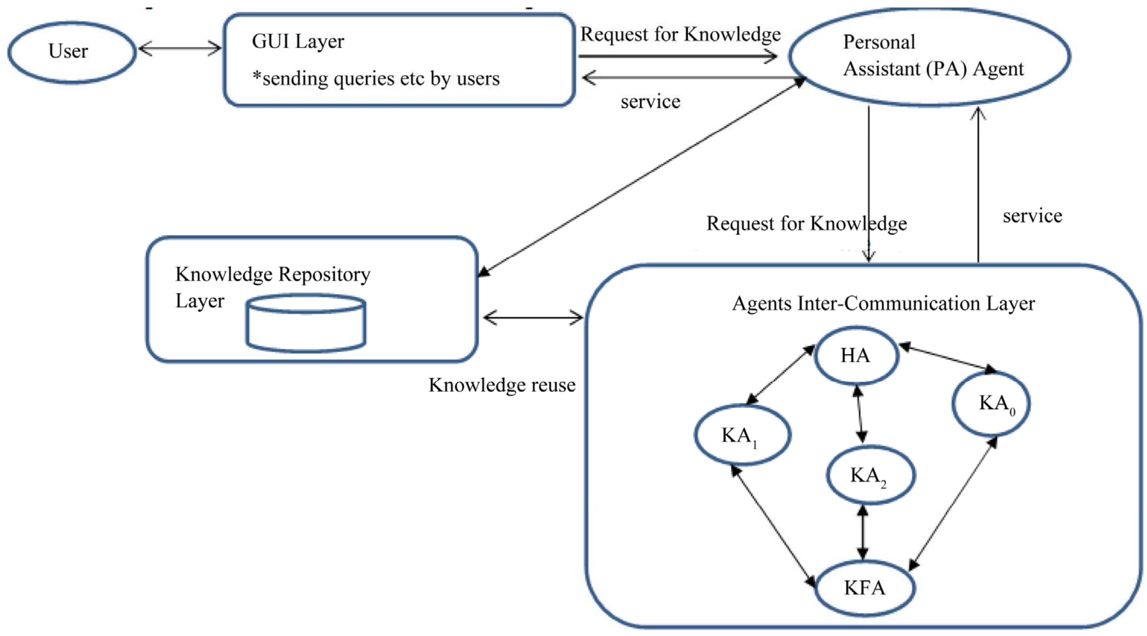

User
GUI Layer
*sending queries etc by users
Request for Knowledge
service
Personal Assistant (PA) Agent
Request for Knowledge
service
Knowledge Repository Layer
Knowledge reuse
Agents Inter-Communication Layer
HA
KA$_0$
KA$_1$
KA$_2$
KFA

Figure 2. General MAEKM architecture.

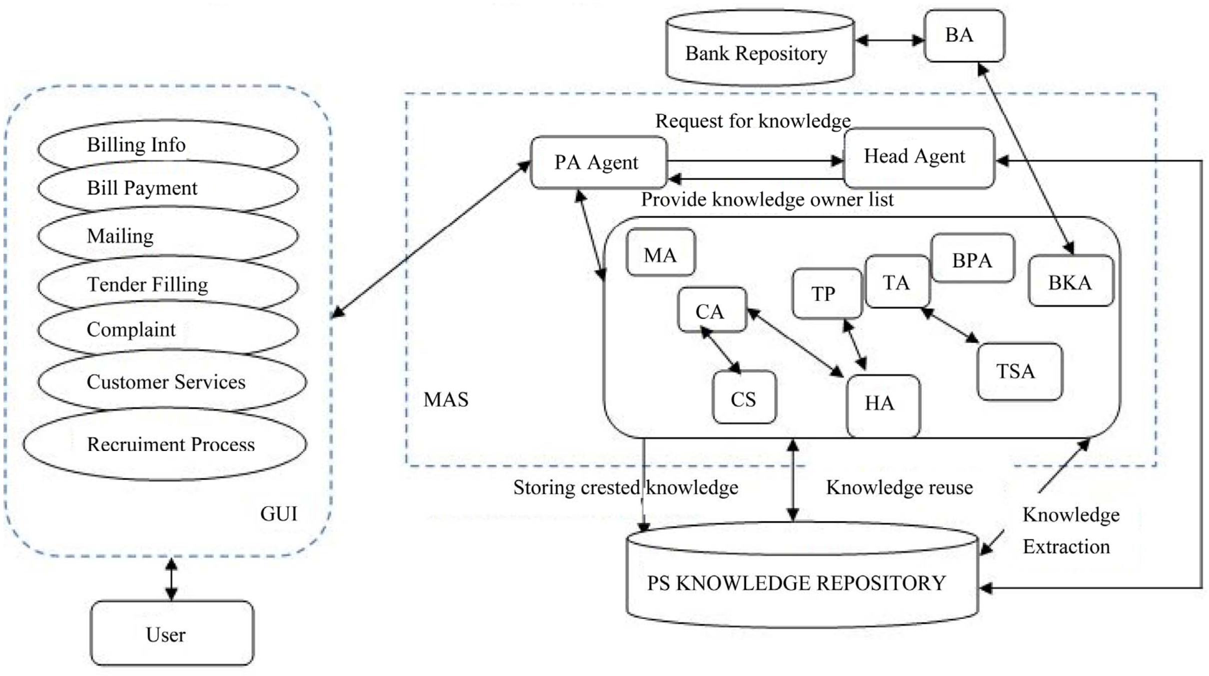


Figure 3. PSU multi agent KM.

services etc.

The following scenario illustrates the functionality of the knowledge market: A customer wants to apply for a new connection of electricity. Customer requires an analysis regarding the new connection schemes which are best suited according to his need. That is not directly available in the knowledge repository. Via the user interface, he communicates his needs and conditions to his Personal Assistant (PA) agent. The conditions set by the customer like type of connection etc. His assistant will serve the request and contacts the Head Agent (HA) in order to find out required agent. HA may use its own internal information about knowledge owners in the system, or possibly, referring to the knowledge repository to find out required agents matching the request and thus provide a list of required agents to the PA agent. Following its own strategy and the preferences specified by the customer, his personal assistant will then contact the required agent and try to get the best deal for his request.

3.2.2.1. Abbreviations Used

- BA: Bank Agent
- BKA: Bill Knowledge Agent
- BPA: Bill Payment Agent
- TA: Tender Agent
- HA: HRD Agent
- MA: Mailing Agent
- CSA: Customer Services Agent
- CA: Complaint Agent
- TSA: Tender Sanction Agent
- TPA: Training & Placement Cell
- PA: Personal Assistant

3.2.2.2. Inter Agent Communication

In **Figure 4**, interaction between Personal Assistant Agent, Head Agent and Customer Service Agent is depicted. This is actually an agent interaction model diagram. These types of diagram are very useful to see, at first glance, as agents interact with each other.

The above figure can be described as follows:

A customer wants to buy a new electricity connection. Then according to the figure:

- Firstly PA Agent interacts with Head Agent for serving query for new connection on behalf of its user.
- Head Agent then reply to PA agent with the list of Agents serving for the related query.
- PA Agent then interacts with Customer Service Agent (CSA) and request for the related query.
- Finally, CSA serves the required knowledge to the PA Agent.

3.2.2.3. Knowledge Creation

Figure 5 shows how creation of the knowledge is done in our implementation.

Knowledge is created by the Customer Service Agent by monitoring that which connection plan is preferred by most of the customers. Therefore by monitoring the user's activities about to purchase a new connection, Customer Service Agent stores the knowledge of most preferred connection plan its personal database.

3.2.2.4. Knowledge Re-Use

Figure 6 shows how creation of the knowledge is done in our implementation.

4. Implementation Details

4.1. Implementation Tools Used

We have implemented the PSU Multi Agent KM architecture using Java Agent Communication Language with JADE [1] in Eclipse Environment. The database we are using is Oracle 9i.

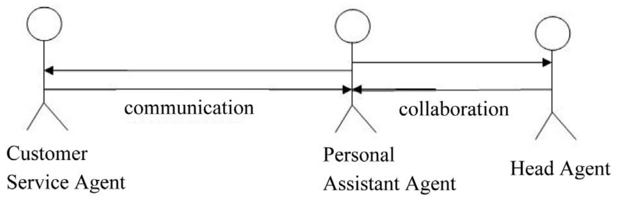


Figure 4. Interaction between agents.

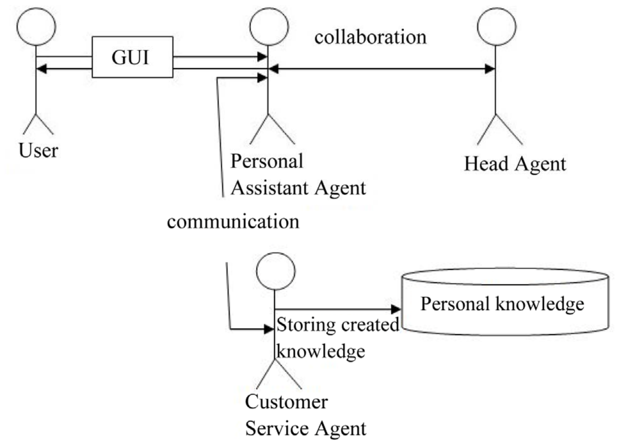


Figure 5. Knowledge creation.

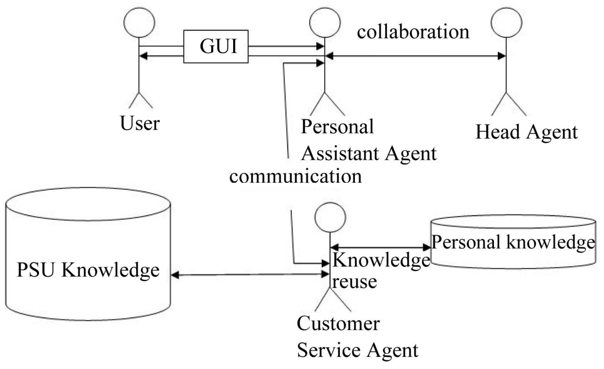


Figure 6. PSU knowledge re-use.

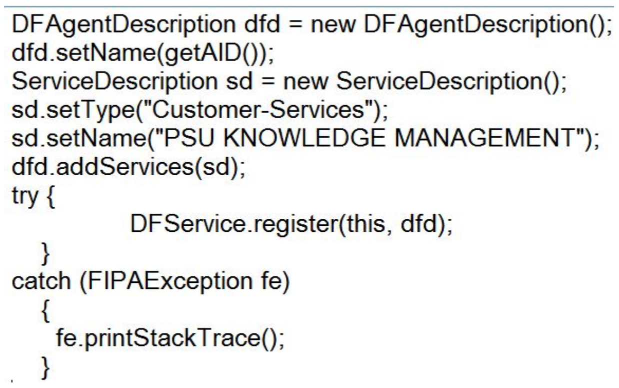
4.2. The Yellow Pages Service

A "yellow pages" service allows agents to publish descriptions of one or more services they provide in order that other agents can easily discover and exploit them. This is depicted in **Figure 7**. Any agent can both register (publish) services and search for (discover) services. Registrations, de-registrations, modifications and searches can be performed at any time during an agent's lifetime. Coding for CustomerServiceAgent publishing their services in yellow pages is given in **Figure 8**.

4.3. Implementation Snapshots

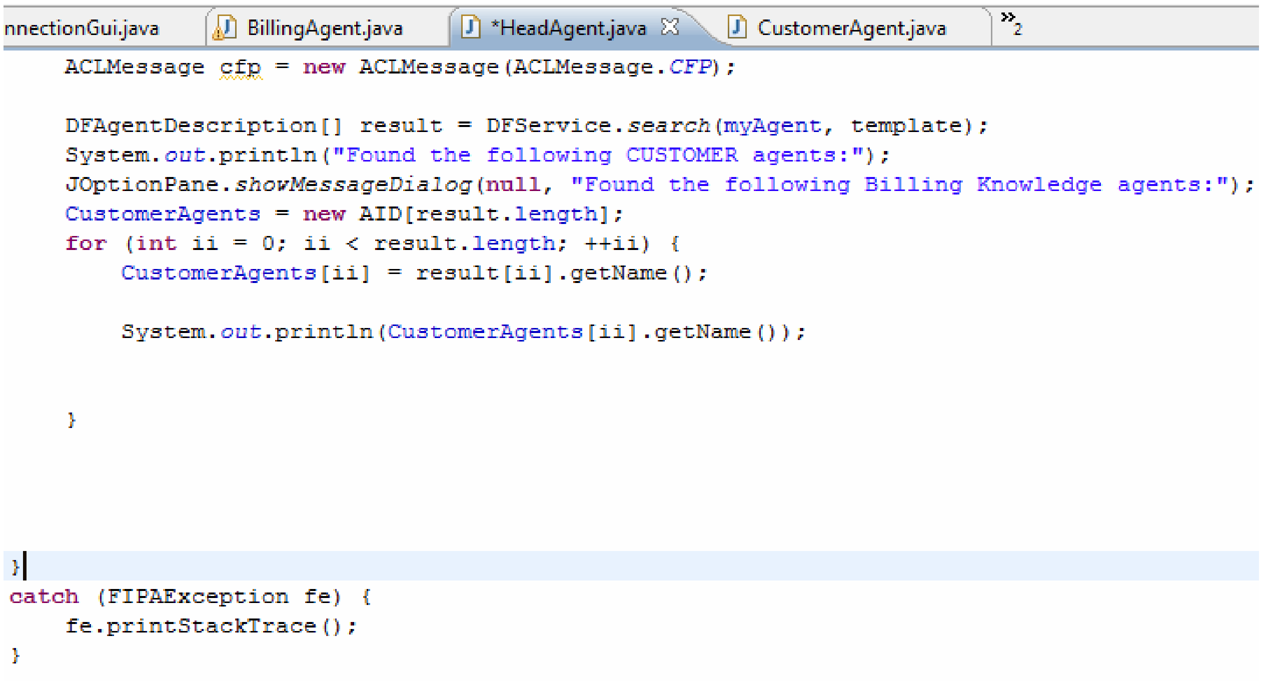
Head Agent Finding Customer Agent:

This code will be run by Head Agent which finds all



```
DFAgentDescription dfd = new DFAgentDescription();
dfd.setName(getAID());
ServiceDescription sd = new ServiceDescription();
sd.setType("Customer-Services");
sd.setName("PSU KNOWLEDGE MANAGEMENT");
dfd.addServices(sd);
try {
            DFService.register(this, dfd);
   }
catch (FIPAException fe)
   {
    fe.printStackTrace();
   }
```

Figure 7. Yellow page service.



```
    ACLMessage cfp = new ACLMessage(ACLMessage.CFP);

    DFAgentDescription[] result = DFService.search(myAgent, template);
    System.out.println("Found the following CUSTOMER agents:");
    JOptionPane.showMessageDialog(null, "Found the following Billing Knowledge agents:");
    CustomerAgents = new AID[result.length];
    for (int ii = 0; ii < result.length; ++ii) {
        CustomerAgents[ii] = result[ii].getName();

        System.out.println(CustomerAgents[ii].getName());


    }




}
catch (FIPAException fe) {
    fe.printStackTrace();
}
```

Figure 8. Coding of head agent finding customer agent.

the customer agents who have registered their services in yellow pages.

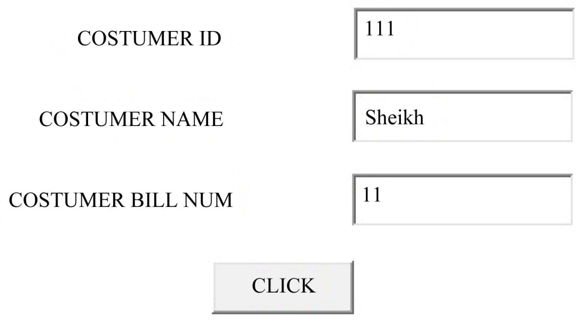
Snapshot of BILL INFO GUI shown in **Figure 9**:

Agents Inter-Communication Snapshots shown in **Figures 10** and **11**.

5. Results and Observations

We have partially implemented our proposed MAEKM Architecture in JADE Framework successfully. After implementation we observed that:

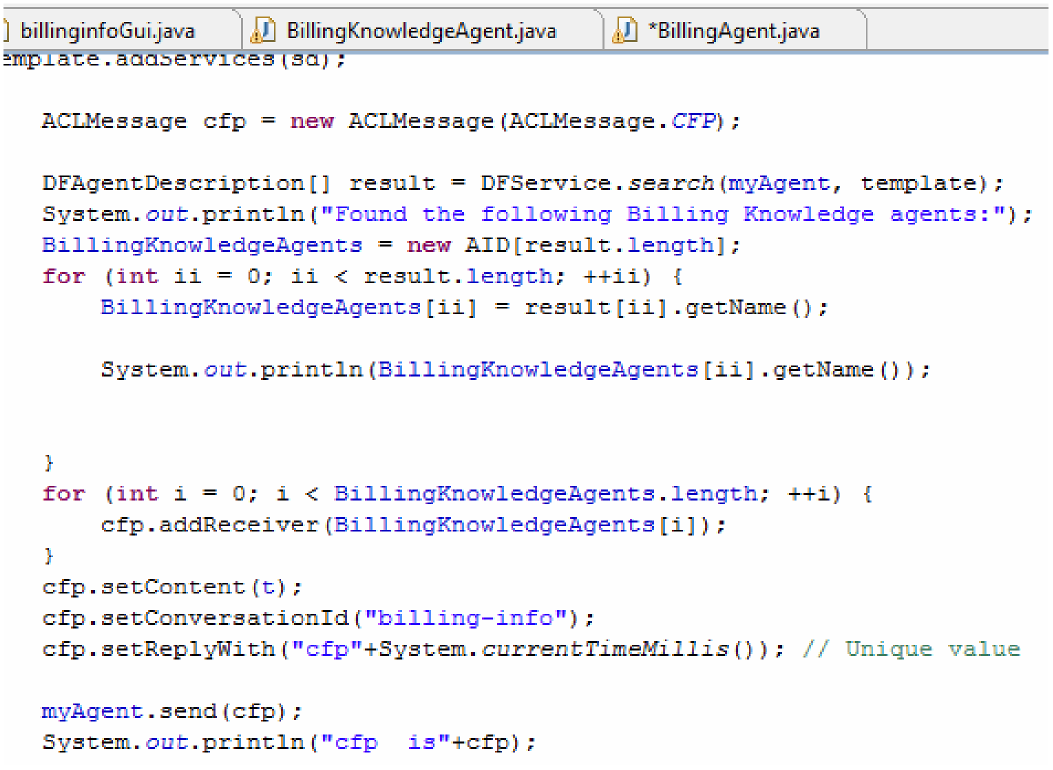
- Our architecture is creating, reusing, managing etc the knowledge in a good manner;
- Our architecture is providing smooth communication between agents;
- Knowledge is serving to the end users in a lesser time;



COSTUMER ID	111
COSTUMER NAME	Sheikh
COSTUMER BILL NUM	11

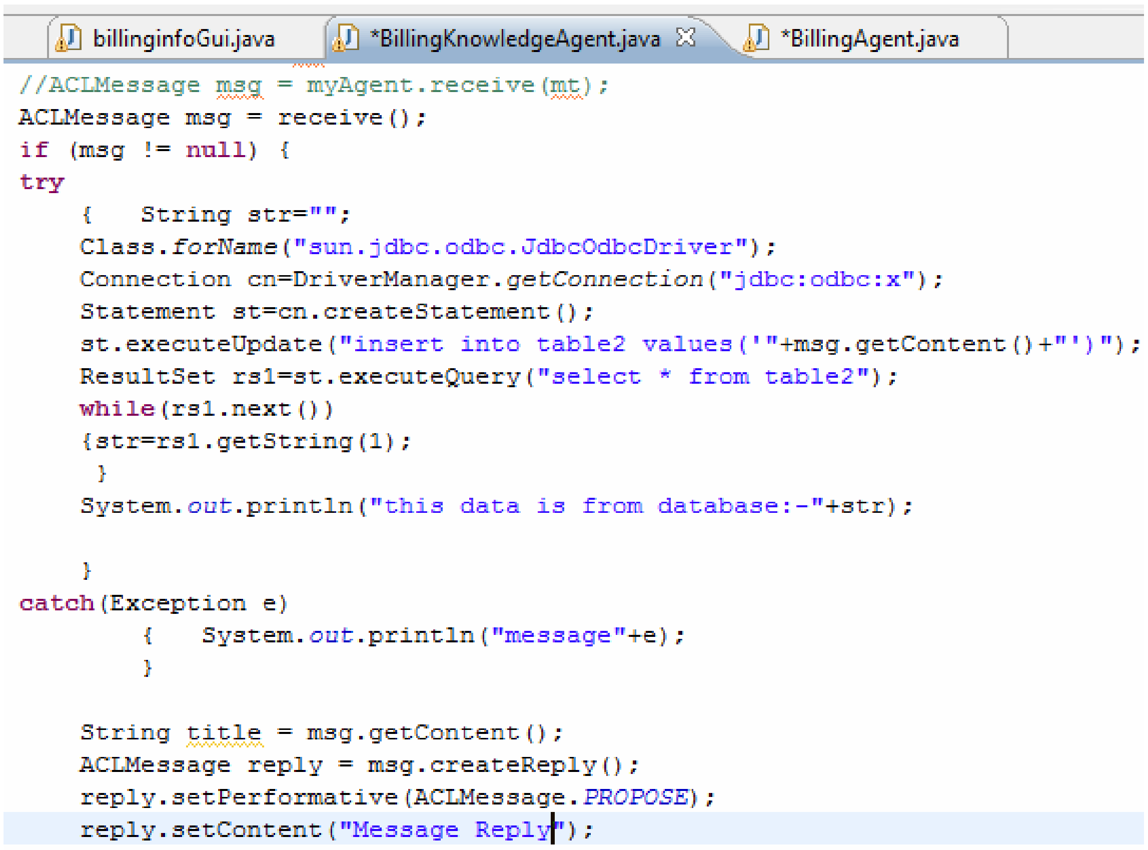
Figure 9. Bill info GUI.

- Relevant knowledge is serving to the users in correct time;
- Providing best possible solutions according to the related queries.



```
emplate.addServices(sd);

  ACLMessage cfp = new ACLMessage(ACLMessage.CFP);

  DFAgentDescription[] result = DFService.search(myAgent, template);
  System.out.println("Found the following Billing Knowledge agents:");
  BillingKnowledgeAgents = new AID[result.length];
  for (int ii = 0; ii < result.length; ++ii) {
      BillingKnowledgeAgents[ii] = result[ii].getName();

      System.out.println(BillingKnowledgeAgents[ii].getName());


  }
  for (int i = 0; i < BillingKnowledgeAgents.length; ++i) {
      cfp.addReceiver(BillingKnowledgeAgents[i]);
  }
  cfp.setContent(t);
  cfp.setConversationId("billing-info");
  cfp.setReplyWith("cfp"+System.currentTimeMillis()); // Unique value

  myAgent.send(cfp);
  System.out.println("cfp  is"+cfp);
```

Figure 10. Sending message from billing agent to billing knowledge agent.



```
//ACLMessage msg = myAgent.receive(mt);
ACLMessage msg = receive();
if (msg != null) {
try
    {   String str="";
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
    Connection cn=DriverManager.getConnection("jdbc:odbc:x");
    Statement st=cn.createStatement();
    st.executeUpdate("insert into table2 values('"+msg.getContent()+"')");
    ResultSet rs1=st.executeQuery("select * from table2");
    while(rs1.next())
    {str=rs1.getString(1);
     }
    System.out.println("this data is from database:-"+str);

    }
catch(Exception e)
        {   System.out.println("message"+e);
        }

    String title = msg.getContent();
    ACLMessage reply = msg.createReply();
    reply.setPerformative(ACLMessage.PROPOSE);
    reply.setContent("Message Reply");
```

Figure 11. Receiving message from billing knowledge agent and reply to billing agent.

6. Conclusions and Future Work

In this paper we have partially implemented the Multi Agent enterprise knowledge management (MAEKM) architecture using JADE with Eclipse for public sector unit. After successfully implementation it will be able to provide smooth communication between agents, Knowledge reusability will be done by using agent's past experience, reducing time overhead for knowledge serving, able to serve relevant knowledge at correct time, able to provide best possible solutions etc which subsequently enhances the system productivity and quality of services. Implementing EKM using Multi Agent, results in more reliability, fast processing etc.

The future work on this paper will incorporate complete implementation of this architecture to fulfill the primary goal of organizational productivity enhancement.

REFERENCES

[1] F. Bellifemine, G. Caire and D. Greenwood, "Developing Multi Agent System with JADE," Wiley Series, England, 2007. doi:10.1002/9780470058411

[2] J. Kephart and D. Chess, "The Vision of Autonomic Computing," *IEEE Computer*, Vol. 36, No. 1, 2003, pp. 41-50. doi:10.1109/MC.2003.1160055

[3] M. E. Nissen, M. N. Kamel and K. C. Sengupta, "Toward Integrating Knowledge Management, Processes and Systems: A Position Paper," *Proceedings of the AAAI Symposium on Bringing Knowledge to Business Processes*, Stanford, 2008.

[4] "Constructing Excellence, Demystifying Knowledge Management a Best Practice Guide for the Construction Industry," 17th July, 2004. www.constructingexcellence.org.uk/resourcecentre/publications/document.jsp?documentID=116179

[5] L. Van Elst, V. Dignum and A. Abecker, "Agent-Mediated Knowledge Management," *Proceedings of the Agent-Mediated Knowledge Management*, Standford, 2003, pp. 1-30.

[6] P. Maes, "Agents that Reduce Work and Information Overload," *Communications of the ACM*, Vol. 37, No. 7, 1994, pp. 31-40. doi:10.1145/176789.176792

[7] "Reticular Systems, Agent Construction Tools," 1999. http://www.agentbuilder.com

[8] "Reticular Systems, Agent Builder—An Integrated Toolkit for Constructing Intelligence Software Agents," 1999. http://www.agentbuilder.com

[9] S. A. DeLoach and M. Wood, "Developing Multiagent Systems with Agent Tool," In: C. Castelfranchi and Y. Lespérance, Eds., *Intelligent Agents* 7, *Agent Theories*, *Architectures*, *and Languages*—7*th International Workshop*, Springer-Verlag, Berlin, 2001.

[10] T. Kawamura, N. Yoshioka, T. Hasegawa, A. Ohsuga and S. Honiden, "Bee-Gent: Bonding and Encapsulation Enhancement Agent Framework for Development of Distributed Systems," *Proceedings of the* 6*th Asia-Pacific Software Engneering Conference*, Takamatsu, 1999.

[11] A. Helsinger, M. Thome and T. Wright, "Cougaar: A Scalable, Distributed Multi-Agent Architecture," *Proceedings of the IEEE Systems*, *Man and Cybernetics Conference*, Hague, 2004, pp. 1910-1917.

[12] M. Winikoff, "Intelligent Agents: An Industrial Strength Platform," In: R. H. Bordini, M. Dastani, J. Dix and A. El Fallah Seghrouchni, Eds., *Multi-Agent Programming*: *Languages*, *Platforms and Applications*, Springer, Berlin, 2005, pp. 175-193.