An Architecture of a Multi Agent Enterprise Knowledge Management System Based on Service Oriented Architecture

Pooja Jain[1], Deepak Dahiya[1]
[1]Jaypee University of Information Technology,
Waknaghat, Solan

Abstract. An enterprise system has many issues which are solved by incorporating the concepts of knowledge management into it. An effective enterprise knowledge management system (EKMS) can be created by using a network of multi agents. An EKMS is a distributed system and distributed systems can be best designed using Service oriented architecture (SOA). These agents act as service providers. The applications can search the repository of services and then select an agent providing the desired service.
Based on the knowledge management, software agent technology and service-oriented architecture (SOA), the agent-based knowledge service-oriented system framework is designed to reflect the distributed, flexible and hierarchical characteristics of an enterprise system. Many other issues of an enterprise system are solved as discussed in the paper.

Keywords: **Multi agent systems, service oriented architecture, web services, enterprise knowledge management system**

1. Introduction

Knowledge management is the second revolution of enterprise management, and is an important means to improve enterprise competition power [11]. The enterprises implementing knowledge management generally include knowledge accumulation, knowledge collection and management, knowledge sharing and knowledge innovation [2]. Enterprise knowledge management systems (EKMS) can be best realized using multiple agents coordinating, collaborating and cooperating with each other to perform the tasks which are not possible by monolithic systems. SOA is an architectural style for building software applications that use services available in a network such as the web. It promotes loose coupling between software components so that they can be reused. A service is an implementation of well-defined business functionality, and such services can then be consumed by clients in different applications or business processes. Applications in SOA are built based on services. An EKMS can be designed using SOA which will bridge the gaps of multi agent systems and SOA.
Based on the knowledge management, software agent technology and service-oriented architecture (SOA), the agent-based knowledge service-oriented system framework is designed to reflect the distributed, flexible and hierarchical characteristics of an enterprise system.
The next section talks about the related work done in the field of multi agent EKMS and SOA. It also gives

a brief introduction of all the concepts used in this paper. The third section talks about the importance of knowledge management in an enterprise. It also outlines the design of an EKMS. The knowledge creation layer is discussed in detail in this section. The fourth section deals with the importance of service oriented architecture. It also talks about the reason of choosing SOA for the design of a multi agent EKMS. The next section details the architecture of the proposed multi agent EKMS based on SOA. The sixth section describes an application working on the proposed architecture. The final section concludes and tells the future work that can be done in this direction.

2. Related work

Multi agent systems are open and extensible systems that allow for the deployment of autonomous and proactive software components [7]. The increasing complexity of the software systems has constantly led to the evolution of new programming paradigms: from functional, to object-oriented, to component oriented, to service–oriented [8]. An Enterprise Model (EM) is a hierarchical network of rules that enables an agent to explain, anticipate, and predict events and interaction patterns in the enterprise and in its environment [9]. Service-Oriented Architecture (SOA) is a standards-based; technology independent distributed computing paradigm and an architectural style which is especially suited to meet the today's demands of dynamic and flexible business applications. With the help of independent and loosely-coupled software services, SOA is able to provide a platform for an efficient and effective publication, discovery, binding, and assembly of these services. Intelligent agents can be regarded as autonomous, problem-solving computational entities with social abilities that are capable of effective proactive behavior in open and dynamic environments. If the term entity is replaced by service, the two concepts of MAS and SOA can be combined together to get a system which can bridge the gaps of the both the paradigms [10].

3. Importance of KM in an Enterprise

An enterprise system is a multi facet, complex large-scale system exhibiting various inherent inefficiencies. There may be technological as well as organizational deficiencies [1]. Enterprise systems are large-scale, integrated application-software packages that use the latest information technology concepts to support business processes. With the rapid development of knowledge economy, knowledge management has become the main impetus of enterprise development [3]. Knowledge management has become the core competence and key of sustainable development for organizations.

KMS architecture model should contain several basic conditions:

1. Analysis of the enterprise's values and operating environment of the system.
2. Integration and diffusion of knowledge management concept.
3. Design and optimize business operations across the enterprise
4. Teach the right knowledge to the right people at the right time

Multi agent systems are used to solve the problems which a monolithic system can't solve. Moreover the enterprise systems are distributive and in such case multi agent approach is the best one. According to the KMS functional structure, infrastructure is the back bone of the knowledge management. It includes database, knowledge base, multi-database coordination systems, networks and various communication channels between people [2].

The enterprises implementing knowledge management generally include knowledge creation, knowledge acquisition, knowledge accumulation, knowledge collection and management, knowledge sharing and knowledge transfer and knowledge retrieval. The fig 1 depicts the phases in an enterprise knowledge management system.

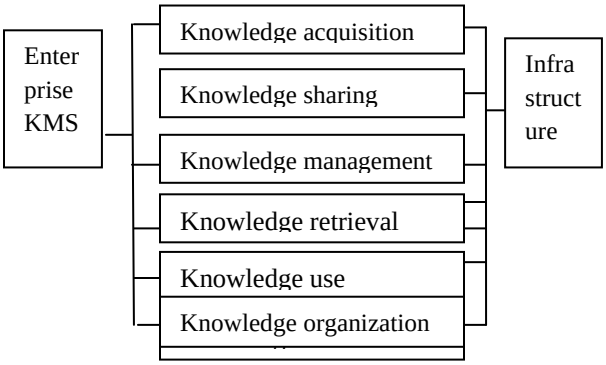

Fig 1: Enterprise knowledge management system

Discussing about all the phases of KM is out of the scope of this paper. To, give a brief idea, the Knowledge creation module is discussed in detail.

Knowledge creation is an important module of an Enterprise Knowledge Management System. Knowledge can be created based on ongoing experience in a specific domain and then using this newly created knowledge in combination with the

existing, to come up with updated knowledge for knowledge sharing. The knowledge creation layer can be viewed as consisting of many agents collaborating, cooperating and coordinating together to create knowledge in various ways. The design is depicted in the Figure 2 below.

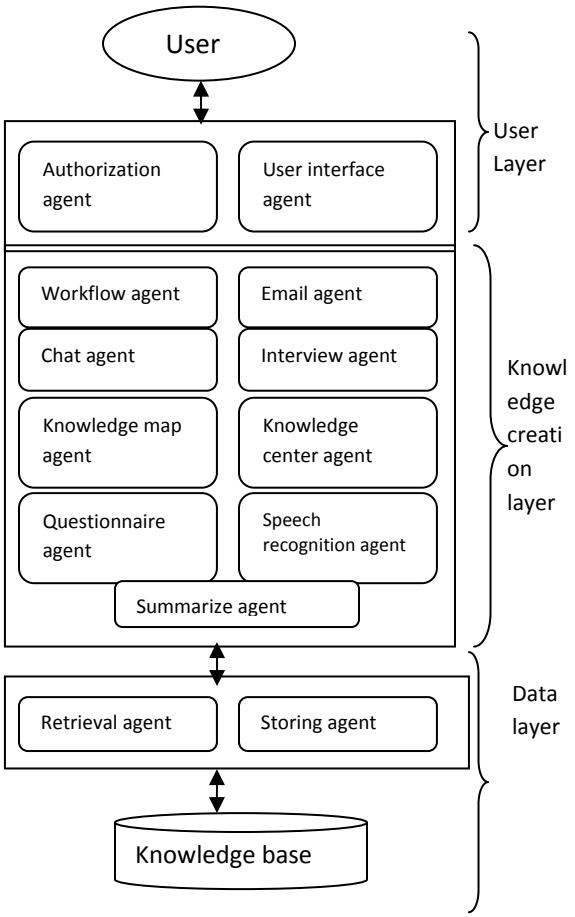

Figure 2. Multi agent knowledge creation layer

The *authorization agent* and the user interface agent lie in the user layer which is directly accessible by the user. The user needs to first authenticate himself before using the system. Once the authentication is done, the *user interface agent* gives the appropriate options to the user, which the user can select depending upon his requirement. There is one to one correspondence between the agents and the choices given to the user. For example, if the user wants to chat with an expert, then the *chat agent* is made active by the user through the *user interface agent*.

The *questionnaire agent* is responsible of making the questionnaire and sending them to the people through the email agent. Questionnaire is the way of asking questions from the people and then coming to a conclusion based on their answers. The questions chosen generally refer to

- the use of recorded information from past projects; the benchmarking of recorded information across projects
- the involvement of people in projects based on records of relevant expertise (as obtained from the knowledge map)
- the extent to which learning across projects occurred through either formal or informal channels, and the use of informal contacts to involve people in projects
- the extent to which lack of knowledge and information was seen as the critical constraint on developing projects whether the concept of knowledge management was recognized as an important issue in respondent's firms [6].

Once the responses of the questionnaire are obtained, the *questionnaire agent* analyses them to reach a conclusion as desired by the user and then finally stores it in the knowledge base through the storing agent. In this way the tacit knowledge of the employees of the organization is converted into explicit knowledge.

Interview agent takes care of the interviews conducted in an enterprise for a particular project. Manual knowledge elicitation process consists of interviewing knowledgeable individuals and eliciting their knowledge via several question-and-answer sessions. An Interview is a face-to-face discussion between the:

- Subject matter experts (SMEs) who possess the domain knowledge
- Knowledge Engineers (KE) who ask questions, observe the expert solving problems.

There are basically three types of interviews:-

- Structured: Questions and responses are definitive. Used when specific information is sought. Basically used when the user knows what exactly has to be asked.
- Semi-structured: Predefined questions are asked but expert has some freedom in expressing the answers
- Unstructured: Neither the questions nor their responses specified in advance. Used when exploring an issue. The user doesn't know exactly the questions to be asked. He takes the interview to gain some knowledge about a particular domain

Any type of interview, as explained above can be carried out to create knowledge. The interview agent

sets the stage and establishes rapport, properly phrases the questions, listens closely and avoids the arguments and finally evaluates the session outcomes.

Chat agent is responsible of recording the chat between a SME and a KE. Usually the KE chats with a domain expert to gain knowledge about the project and is not able to record the chat between the two. Moreover, there may be hundreds of such chats and it becomes very tough for the KE to re-visit the saved chat he wants, at the later point of time. The chat agent records the chat and analyses and summarizes the chat for the KE. It removes the stop words and records only the useful information. It also keeps track of the date, time and the topic of the chat. So, if the KE wants to see a particular chat in future, he can do so easily.

Knowledge map agent is responsible for making the knowledge map. It's like the yellow pages that contain the name of the experts and their field of expertise. Yellow pages help the people to get the right information at the right time from the right person. It helps the employee of an enterprise to reach to the correct person without wasting the time in finding out "who knows what".

The *Knowledge center agent* identifies the experts in different domains. There may be many people working in a domain but there will only a few experts in that domain. The *knowledge center agent* identifies such experts and calls them as knowledge centers. Whenever a user needs the knowledge about a particular domain, then he can directly contact the knowledge center through the email agent.

Email agent acts in collaboration with other agents. Whenever a person wants to create knowledge or get some knowledge from another person, he can drop a mail to that person using this agent easily. For example, an employee of an enterprise wants to know about the HR policy related to maternity leave. Then the *knowledge map agent* will help him to know the person to be contacted. Then he can email that person directly using the *email agent* in collaboration with the *knowledge map agent*.

Speech recognition agent converts tacit knowledge into explicit. The lectures, tutorials given by the experts are converted into documents that can be referred to in the future. This type of knowledge conversion is of utmost importance as the tacit knowledge lies with the people and when they leave an organization, knowledge also goes along with them. There is a need to capture this knowledge so that the organization doesn't suffer even if the employee leaves.

Summarize agent summarizes the documents present in the database of an enterprise. An organization has a huge amount of data. A new employee finds it almost impossible to go through all the documents and then extract the information required for his project work. The *summarize agent* condenses the documents so that the required knowledge is obtained easily and in minimum time.

Workflow agent is responsible for the overall flow of the information. It takes care of the collaboration and the coordination between the different agents. It takes care of the flow of knowledge from one agent to another.

The *retrieval agent* and the *storing agent* directly interact with the knowledge base. The *retrieval agent* is responsible of giving the right information to the knowledge create layer at the right time in the right form. The *storing agent* stores the data in the knowledge base as and when required.

4. Advantages of SOA

SOA is emerging as the premier integration and architecture framework in today's complex and heterogeneous computing environment.SOA is an architectural style for building software applications that use services available in a network such as the web. The fundamental organizing concept in SOA is service. Many sys can be viewed as collection of services. A service is defined as a collection of capabilities. A capability is an atomic unit of work that can be performed in a service [4]. A well designed service has a set of related capabilities.

There are certain of qualities of SOA, which has made it very popular in recent times, and the reason behind why it has been chosen in this paper. Some of them are:-

1. **Modularity**:-In order to promote modularity, the designer isolates the code into units that deal with limited functionality. An additional goal of SOA is the less use of repeated or duplicated code. Services break a single application into a set of services that are separate code units. Thus, services can be reused in multiple applications

and the functionality of the service is implemented once

2. **Easy to modify the code**:-Unlike a stand-alone application, where the code is deployed after test, services are independent units of code and can be changed independent of applications that use the service.
3. **Loose coupling**:-Loose coupling is reducing dependencies between the service contract, its implementation, and its service consumers. Loose coupling ensures that the function performed by the service does not depend on the services that call them or that may call. Services deploy as standalone code units with a defined and known function. A loosely coupled architecture allows replacing or modifying the components, without making any changes in the other components of the system. Thus an enterprise can change its business systems as and when required, without modifying the entire system as a whole. Due to loose coupling, the recovery is also easy in case of any failure, since there is no dependency between the different components of the system.
4. **Technological independence**: - The developers can choose the right desired technology for a job without concerning themselves with technical dependencies. Thus he can make a module in J2EE to work with other components written in C, C++ or COBOL.
5. **Service abstraction**: - This property arises from the fact that the concept of service could fit well in object oriented programming, in which a service is a kind of object and the capabilities are analogous to the methods of the object. As in the case of objects, the methods, functions, properties and attributes are hidden from the rest of the application, the details of the service are also hidden from the rest of the system. But to implement service discoverability, some abstraction needs to be ignored. So, a careful balance is required to meet proper abstraction and still enable discovery
6. **Service autonomy**:-It's a quality common to multi agent systems. A service like an agent operates autonomously. The service controls its own operating environment and at the same time just like an intelligent agent, it learns from the environment.
7. **Service discoverability**:-the ultimate in discovery is an automated sys that dynamically identifies and locates the required services. A service registry stores and publishes information about all the available services. It's like the yellow pages. Whenever a service is required by an application, it checks in the registry and will get the details of a fitting service implementation, and can connect.
8. **Reusability**. Modularity leads to reusability. Developers can take the code developed for existing business applications, and then reuse it to meet new business requirements. Reusing existing functionality outside or inside an enterprise instead of re-developing the same code can result in a huge savings in terms of cost and time. In SOA, the only thing that a requesting application needs to know is the public interface of the service.
9. **Interoperability**. The services can communicate and understand with each other no matter what platform they run on. This is achieved by employing a standard way of communication, a protocol that's consistent across platforms, systems, and languages. Web services is able to solve this problem effectively. Web services consist of a set of protocols and technologies that are widely accepted and used, and that are platform, system, and language independent.
10. **Scalability**. Since services in SOA are loosely coupled, applications using these services tend to scale more easily than applications in a more tightly-coupled environment. That's because there are very few dependencies between the service provider and the service user. Services in a web services-based SOA are coarse-grained. Coarse-grained services offer a set of related business functions rather than a single function. Whereas a fine-grained service might handle only one operation in one particular application.
11. **Cost Efficiency**. Tightly-coupled systems are costly to maintain and extend because changes in one component require changes in other components of the application as well. Services in SOA are loosely-coupled, so the applications that use these services are less costly to maintain and easier to extend than customized solutions. SOA is about reuse of business functions. This is potentially the biggest cost saving of all [5].

In short, the important advantages of SOA are enhanced software architecture, improved software development time and cost, increased reuse through modularity.

Thus it's apparent that there is a one to one correspondence between the agents and the services. The similarity between SOA and multi agent architectures indicates that SOA could be a very good implementation vehicle for MAS. The properties of services are common to those of agents. So, a system based on service oriented architecture can be realized with the help of intelligent, multiple, software agents coordinating, collaborating and cooperating with each other to do the jobs which are not possible by a monolithic system. The next section talks about the architecture of such a proposed multi agent enterprise knowledge management system based on service oriented architecture.

5. Architecture

As discussed in the previous sections, a knowledge management system can be designed which is multi agent based on service oriented architecture. The multi agent paradigm is the one which is communication and cooperation oriented and therefore suitable for the internet and enterprise systems. When the intelligent software agents are applied to a distributed system based on SOA, then agents represent service users or service providers and act autonomously on behalf of the system. The problem of establishing and configuring agent based multi agent distributed enterprise systems is stated as follows:-

1. Define the set of agents comprising the distributed sys
2. Define the connectivity of agents through links
3. Define the elementary services that should be supported by each agent
4. Deploy the service resources, i.e. software components and data, that are required to execute services defined [4].

The system parameters of a formal model based on multi agent network can be represented by a double :{ A, S, L} where

A- Set of all the agents in the system

S- Set of all the services provided to user agents by provider agents

L-Set of all the links between the agents of the system

All the three combined together form a network of multi agents collaborating and cooperating with each other to perform the tasks which are not possible by monolithic systems. An agent represents a service user or service provider and can communicate with other agents through message passing. An agent is defined by a double:-

$$A_k = \{name_k, service_k\},$$

where $name_k$ is a unique agent identification and $service_k$ is the service provided or used by the agent. The service of an agent can be represented as:-

$$service_k = \{s_{k1}, s_{k2}, s_{k3} .... s_{kn}\}$$

where $s_{k1}, s_{k2}, s_{k3} .... s_{kn}$ are the different elementary services provided or used by the agent A_k having the service $service_k$.

The same elementary service can be provided by multiple agents or a single agent can provide multiple elementary services. Depending upon the complexity of the applications, agents perform the service request alone or by working in a team of other collaborating agents.

$L_{ij} = \{A_i, A_j\}$

Where L_{ij} is the link between the two agents A_i and A_j.

The network of the multi agents can be depicted in a manner as shown in the fig 3.

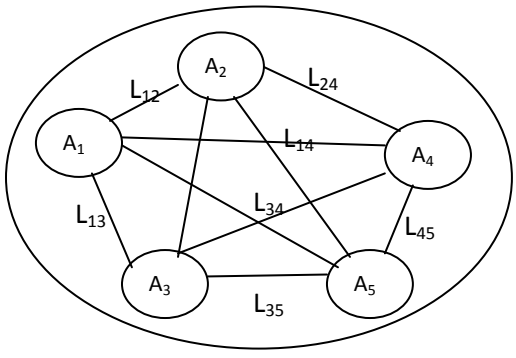

Fig 3: the network of agents in a MAS

The services performed by the different agents can be depicted as in fig 4

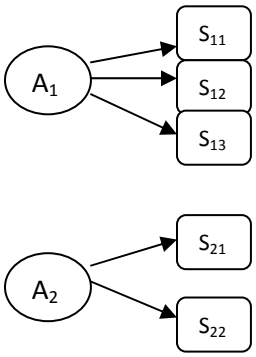

Fig 4: agents providing the services

The agent A_1 performs three elementary services s_{11} s_{12} and s_{13} to perform the service service$_1$ and agent A_2 performs two elementary services s_{11} and s_{12} to perform the service service$_2$

There are many reasons for an enterprise to take an SOA approach, and more specifically, a web services-based SOA approach. The advantages of SOA have already been discussed in the previous section. Web services require containers and monitoring systems for operations of the system but the interface to the service is common across any of the containers or frameworks and involves only the communication protocols and message formats. Web Services generally use either SOAP or XML-RPC as communication method for services, UDDI for service discovery and WSDL for service description.

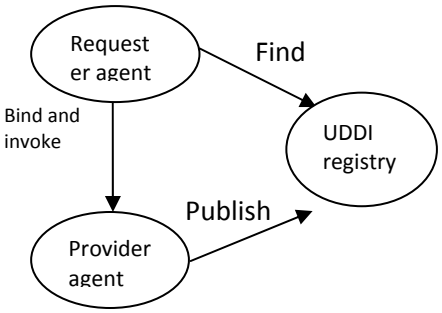

Fig 5: principle of SOA

Services can be described using WSDL and published in a UDDI registry so that they can be discovered and used by web service clients. Applications can use web services published by other providers, regardless of how they are implemented.

The complete picture of a multi agent enterprise knowledge management system based on SOA can be shown by the figure

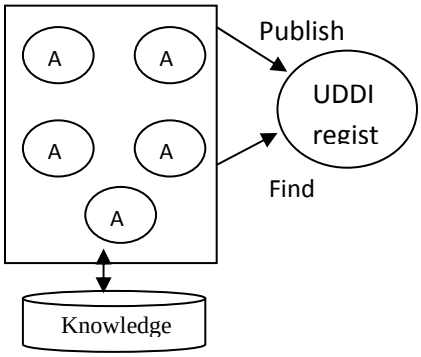

Fig 6: EKMS based on SOA

The entire set of agents A has published their services with the UDDI registry. In other words there is a description of all the services of the set S in the registry. If agent A_1 needs a service, it will search for it in the registry. If the service is found, the same would be notified to A_1. Suppose the service requested is provided by the agent A_2. Once the notification is sent, the agent A_1 will communicate with the agent A_2 through the link L_{12} using the SOAP protocol. The binding and invoking is done using WSDL and SOAP. The agents in the knowledge creation and acquisition layer, updates the knowledge base. The agents in the knowledge use and knowledge retrieval layer get the knowledge from the knowledge base.

The communication between the different agents is done through the protocol SOAP. The basis of transmission in SOAP is a SOAP message, which consists of a mandatory SOAP envelope, an optional header, and a mandatory body [5]. A SOAP message goes from the requester agent to the provider agent. It can move through the intermediate nodes if a direct path is not present. But the architecture, discussed here, assumes that there is a direct path between all the agents.

The SOAP's specification defines nothing more than a simple XML-based envelope for the information being transferred, and a set of rules for translating application and platform-specific data types into XML representations. SOAP's design makes it suitable for a wide variety of application messaging and integration patterns.

WSDL is XML grammar for specifying a public interface for a web service. WSDL describes four critical pieces of data:

- Interface information describing all publicly available functions
- Data type information for all message requests and message responses
- Binding information about the transport protocol to be used
- Address information for locating the specified service [12].

With the help of WSDL, an agent can locate a web service and invoke any of its publicly available functions. WSDL is an XML grammar with six major elements: definitions, types, message, portType, binding and service. It has two utility elements i.e. documentation and import.

The Universal Description, Discovery, and Integration (UDDI) specifications define how to publish and discover information about services in a UDDI-conforming registry. The specifications define a UDDI schema and a UDDI API. The UDDI schema identifies the types of XML data structures that comprise an entry in the registry for a service. UDDI registry can be considered as a "Yellow Pages" for web services. Like the Yellow Pages directory for phone numbers, a UDDI registry provides information about a service such as the name of the service, a brief description of what it does, an address where the service can be accessed, and a description of the interface for accessing the service.
SOA allows for the reuse of existing assets where new services can be created from an existing IT infrastructure of systems. In other words, it enables businesses to leverage existing investments by allowing them to reuse existing applications, and promises interoperability between heterogeneous applications and technologies.

6. Application

The basic criterion for an application to work upon the proposed architecture is that it should be impossible for it work on a monolithic system. Or the performance of it in a monolithic system is so bad that one has to adopt a multi agent knowledge management approach. The application that's considered here is that of Environmental information systems (EIS). EIS performs one or more of the following tasks: environmental monitoring, data storage and access, disaster description and response, environmental reporting, planning and simulation, modeling and decision-making [13]. The basic responsibility of such a system is to monitor critical atmospheric variables and publish regularly their analysis results. On the basis of this result, any disastrous condition is identified and accordingly an alarm is produced. This paper talks about a system that monitors the sea bed with the help of a network of sensors. The information from the sensors goes to a multi agent knowledge management system which validates the data received, corrects the erroneous results, predicts the missing attributes and then prepares a report. Then the report is analyzed by the system and compared with the existing data available in the database. If there is a match then an alarm is produced indicating a disastrous condition, like Tsunami.

The system intervenes between the field sensors and human experts and undertakes several tasks in order to assist humans in their evaluations. The different services provided by the different agents are:-

1. Data_validation
2. Correct_data
3. Predict_missing_data
4. Check_sensor
5. Send_sensor_alarm
6. Connect_database
7. Analyze_data
8. Make_graph
9. Send_graph

The architecture of the system can be depicted as in figure 7.

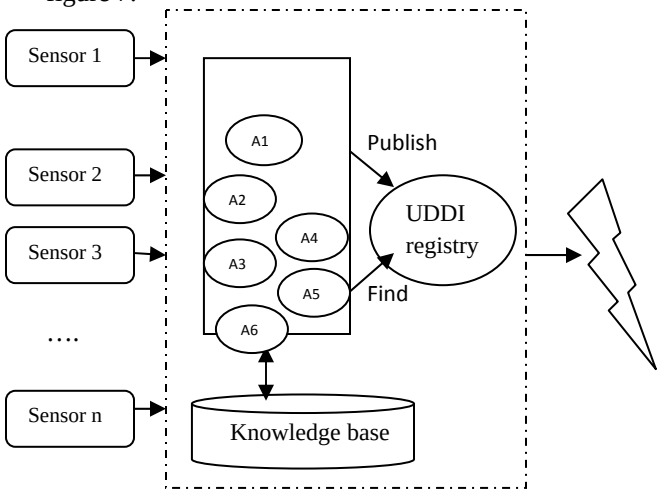

Fig 7: Environmental monitoring system

The agents depicted in the figure above, perform the service as listed earlier. These agents need to interact with each other to perform the various services. The agents are autonomous and have decision making powers depending upon the knowledge retrieved from the knowledge base and from the collaboration between other agents.
The knowledge base for such a system should be updated from time to time. All the modules/ phases of knowledge management are applied so that the data is most updated and the decision making capability of the agents can be effectively relied upon.
The responsibilities of the different agents are to carry out their services in coordination with other agents. The data which is received from the sensors is validated by the data_validation agent. The validity is determined by the pre-existing data available in the knowledge base. If the existing and received data are the same, then the data is validated. If there is any

discrepancy then the data validation agent takes the service of the correct data agent and the predict_missing_data agent. Once the data is corrected then it's analyzed by using the service of the analyze_data agent. If the agents are still not able to correct the data then the service of the check_sensor is taken. This agent checks for the sensor whether it's working or not. If the sensor is faulty then corresponding alarm is produced by the send_sensor_alarm agent. If the sensor is working fine then such a data is converted into a graph by the make_graph agent. Then this graph is sent to the user by the send_graph agent. If the user validates the graph, then this entire information is updated in the knowledge base by the connect_database agent. So that such a data is not termed incorrect in the future by the system. The service of the connect_database agent is taken from time to time by the different agents as required. Thus the principles of re-use and modularity are applied by the use of SOA. Some agents are easy to code in C++ and some in Java. So, different agents can be coded in different languages as desired by the developer. This freedom is given by SOA. By the use of SOA, the agents are loosely coupled, so in future, if any other functionality is to be added in the system or any change is to be made, it can be done without affecting the entire application as a whole.

Sometimes it may happen that an alarm is produced in a normal situation, which may make the system unreliable. Such conditions are stored in the knowledge base by the knowledge acquisition layer. So that when a similar situation arises, then the system doesn't produce an alarm. The knowledge creation layer interacts with the human user from time to time to get his feedback or some other data pertaining to the disastrous conditions. Such details are also stored in the knowledge base to be used in the future. The knowledge organization layer takes care of the organization of the data in such a way that the right data is available at the right time. Such environmental monitoring systems are real time systems. When a tsunami comes, then the system gets very less time to validate the data, analyze it and then infer a tsunami. Thus, the data should be very well organized so that the search time is minimal and the system gives the alarm at the right time without delay.

Thus it's seen that when a multi agent knowledge management system is designed using service oriented architecture, then the tasks are simplified. The system becomes less complex, less costly, modular, loosely coupled technologically independent.

7. Conclusion and future work

This paper talks about a framework of an agent-based knowledge service-oriented system that integrates knowledge management approaches, software agent technology and a service-oriented architecture (SOA). The basic performance measures relevant to agent based internet applications and enterprise systems are response time and throughput, i.e. the time required to handle a service request and the number of services completed in unit time. The basic aim of this paper is to increase the through and at the same time reduce the response time. By employing a multi agent knowledge management system for an enterprise based on service oriented architecture can effectively achieve these criteria. The future work corresponds to extending this architecture for inter-enterprise collaboration. If any service is not available with any one enterprise then it can use the service of provided some other enterprise. The authors would also like to implement the above architecture for an application using JADE and web services.

8. References

1. Corporate IT Knowledge Workbench : Case Study by Kemal A. Delic – Laurent Douillet at the Proceedings of the 15th International Workshop on Database and Expert Systems Applications (DEXA'04)
2. Enterprise Knowledge Management System Design and Implementation by HAN Xiao-cui at International conference on Computer Engineering and Technology (ICCET), 2010
3. Research on modeling and implementation of knowledge management system in virtue enterprise by MIN HUANG, BO SUN, Proceedings of the Eighth International Conference on Machine Learning and Cybernetics, Baoding, 12-15 July 2009
4. Agent and multi-agent technology for internet and enterprise systems by Anne Hakansson.
5. Service-Oriented Architecture and Web Services: Concepts,Technologies, and Tools by By Ed Ort, April 2005
6. Stuart Barnes (Edited). Knowledge management systems: theory and practice. Published by Thomson Learning, 2002.

7. A Multi Agent-Based System for Securing University Campus: Design and Architecture by Faisal Alkhateeb,Shadi Aljawarneh at 2010 International Conference on Intelligent Systems, Modelling and Simulation
8. A.Granebring and P. Révay, Creating sustainable retail processes with service-oriented architecture, MicroCAD-06 conference, Miskolc,Hungary 2006.
9. Enterprise Knowledge Management Modeling and Distributed Knowledge Management Systems By Joseph M. Firestone, Ph.D. January 3, 1999
10. Service-Oriented Architecture and (Multi-)Agent Systems Technology by Monique Calisti, Frank Dignum P., Ryszard Kowalczyk, Frank Leymann and Rainer Unland at Dagstuhl Seminar Proceedings 10021
11. An Evaluation Method for Enterprise Knowledge Management Performance Based on Linguistic Variable by Rong Chen, Peide Liu, Shukun Tang at 2008 International Seminar on Business and Information Management
12. Web Services Essentials Distributed Applications with XML-RPC, SOAP, UDDI & WSDL by Ethan Cerami
13. An agent-based intelligent environmental monitoring system by Ioannis N. Athanasiadis and Pericles A. Mitkas at Management of Environmental Quality, An International Journal vol.15,no.3,pp.238-49,2004.Emerald Publishers.
14. Predictive wireless sensor networks by Pooja Jain, M Radha Krishna held at Wireless Communication and Sensor networks (WCSN-2005), 5-6 March 2005 at Allahabad.