Agent Based Bioinformatics Integration Using RETSINA

Kulathuran Shunmuganathan[1], Kumar Deepika[2], and Kumaradhas Deeba[2]
[1]Anna University, India.
[2]Noorul Islam College of Engineering, India

Abstract: *Vast amounts of life sciences data are scattered around the world in the form of a variety of heterogeneous data sources. The need to be able to co-relate relevant information is fundamental to increase the overall knowledge and understanding of a specific subject. Bioinformaticians aspire to find ways to integrate biological data sources for this purpose and system integration is a very important research topic. The purpose of this paper is to provide an overview of important integration issues that should be considered when designing a bioinformatics integration system. The currently prevailing approach for integration is presented with examples of bioinformatics information systems together with their main characteristics. Here, we introduce agent technology and we argue why it provides an appropriate solution for designing bioinformatics integration systems.*

Keywords: *Wrapper, system integration, agent technology, mediator, RETSINA.*

Received December 16, 2006; accepted April 23, 2007

1. Introduction

System integration is a challenging research topic, important for bioinformatics. Agent technology has been successfully applied in the past to system integration. In the following paper we introduce agent technology and argue that it is appropriate for bioinformatics systems integration. Section 2 represents the integration of bioinformatics system. Agent technology is discussed in section 3 and section 4 summarizes the bioinformatics integration and agent technology.

2. Bioinformatics System Integration

System integration is a challenging research in Bioinformatics systems because of the inherent complexity of the domain in which: (1) most rules have exceptions; (2) there is a rich variety in data demanding vast amounts of storage capacity; (3) complex relationships between structures; (4) variation in curation and quality control standards [9]; (5) multiple sources of similar data or interpreted versions of the same data; and (6) uncertainty, natural variation, experimental error, interpretation error, computational error. In this section we will introduce principal aspects of system integration with a focus on bioinformatics systems.

2.1. Fundamental Aspects of Integration

The main goal of integration is to provide mechanisms that can unify a number of computer systems. We can describe request of data from various resources and then combine the results to get more useful information using integrated systems as a number of steps: (1) the user makes a request (query) to the integrated system; (2) the integration system processes the request and decides how to split it into sub-requests specific to data sources; (3) the sub-requests are made and all individual results are returned to the integration system; and (4) the results are combined to a coherent answer which is returned to the user. Three important aspects of system integration are *distribution*, *autonomy* and *heterogeneity*.

2.2. Heterogeneity

Heterogeneity has two major categories [10]:

- *Technical*: such differences can occur because of different hardware platforms, operating systems, database management systems (query languages, data models), access protocols, transport formats, and programming languages.
- *Semantic*: conceptual differences occur in the data models/schemas of the data sources, i.e., the organisation of data and the relationships between such data.

To bridge schema heterogeneity we usually define a common schema expressed in a Common Data Model (CDM). Each local data model is mapped to the CDM thereby resolving semantic heterogeneity. Integrated systems that aim to create a CDM and a federated schema are called federated systems.

2.3. Federated Systems

Federated systems can be classified in terms of their degree of federation and instantiation. The first refers to how autonomous-independent from the integration system-the data sources are; autonomy indirectly influences the precision of the schema integration. We can have a tight federation, which involves non-autonomous data sources-potentially very precise matching of the local schemas-, and capability to allow reliable read-write access to the integrated system. Alternatively, a loose federation means completely autonomous data sources-constraint matching of the local schemas-, and only read-only reliable access to the data sources. The second, the degree of instantiation, refers to where the physical data reside. We can have a virtual federation, which means that the actual data reside in the respective data sources, and the integration system provides just a unified view of these data sources, or a materialized federation-also called warehousing-in which the integrated system consists of a global physical repository, which includes all the data sources' data. Although a materialized solution is more efficient computationally, in general the virtual approach is preferred as it does not involve data replication-which introduces data update and synchronization problems-and it is much easier to maintain [2].

2.4. Legacy Systems and Wrappers

Each data source has a query language that allows users to request data from that resource. This query language is designed to achieve mapping between the two. To deal with query language heterogeneity, integration systems use a global query language-also called internal or Common Query Language (CQL). This language is used as the common language between the heterogeneous data sources and it should be designed according to the common data model used. With a query formulated in the CQL the integrated system could use the federated schema to decompose, usually referred to as query decomposition and planning, the initial query to sub-queries that could be answered by individual resources. The sub-query, expressed using the CQL, is then translated to the data source-specific language. This task is accomplished by using software modules called wrappers. Wrappers encapsulate or 'wrap' the functionality of existing legacy systems. They are responsible for converting a request formulated in the CQL to the specific query language used by a data source and vice versa.

2.5. Mediation and Bioinformatics Integration Systems

One of the most common integration approaches in bioinformatics is mediation. Mediators were introduced with the argument that they "simplify, abstract, reduce, merge, and explain data" and their primary purpose is seamless integration of heterogeneous data sources. A mediator is a software module that exploits encoded knowledge about certain sets or subsets of data to create information for a higher layer of applications. Mediation is an abstract architecture that conceptualizes integration. In our integration overview we presented a more practical view of the integration procedure that is summarized in Figure 1.

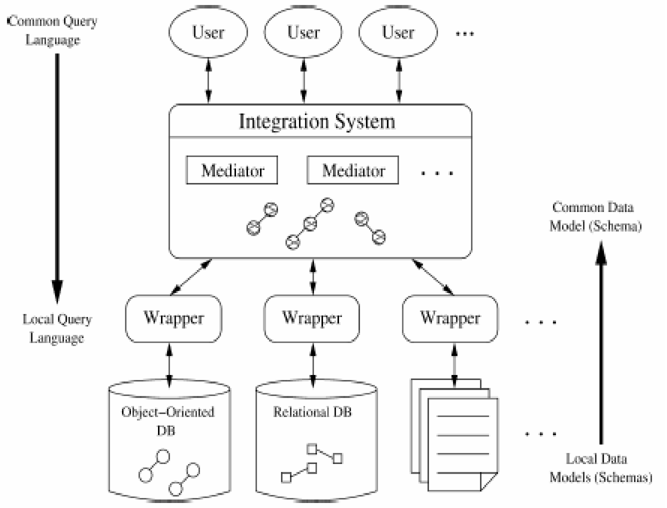

Figure 1. Overview of integration procedure.

We can now describe the integration steps in more detail:

- The user provides a query formulated in the common query language to the integration system-mediator(s).
- The integration system applies the query to the common data model. The query decomposition and planning module (part of the mediators) decomposes the initial query into sub-queries, again formulated in the common query language. The sub-queries are passed to the appropriate data sources via their respective wrappers. Each wrapper translates the sub-queries to the local query language used by the data source and then translates the results back to the CQL.
- The results are then returned to the integration system where they are combined to a coherent result, which is returned back to the user.

Systems that do not provide a conceptual model in their CDM, or not a CDM at all, cannot provide integration and location transparency. That means that the user has to define how the data sources' data will be combined and which data sources should be used; we refer to such systems as non-transparent. Bioinformatics integration systems follow the procedure illustrated in Figure1 and their functionality can be generally described with the integration steps mentioned above. Most of bioinformatics integration systems follow the system integration by providing a

CDM and a CQL as an intermediate layer; an integration system can dynamically answer any queries related to the integrated data sources as described by the CDM.

2.6. Confidence in Results' Quality

Transparency avoids the need for the user to know which data sources contain the information needed and how the resulting data should be combined to reach a final result. A transparent system incorporates integration reasoning automates the integration procedure. Allowing the user to intervene limits the system's transparency. Although, integration would be transparent the user should be able to adjust the level of transparency according to his/her needs.

2.7. Semantic Web Services, Semantic Grid, and Integration

With data integration, developers could not be certain of the purpose (semantics) of the service. The semantic web has been developed to provide a solution. The goal is to provide common meaning between concepts used in web pages and services. To this end: (1) a general-purpose data format has been designed (XML [1]); (2) it was extended to allow for metadata (RDF); (3) basic semantics for the data structures and values allowed have been specified (RDF Schemas), and recently; (4) fully developed ontology languages have been defined, e.g., the Web Ontology Language (OWL). Ontology is a group of concept definitions that describe an application domain. The very large-scale distributed computing and data required of particle physics coupled with the need to go beyond the limited stateless and insecure web service technology has led to the development of the grid [6]. The goal was to inter-connect a large amount of computing resources at a national or even worldwide scale to build 'cheap' virtual supercomputers. Other research communities, such as biology, earth science, and astronomy, have expressed interest in the grid. This change of focus made other extensions necessary; for example, to resolve heterogeneity of the disparate resources and to incorporate ontologies, led to what is now called semantic grid. In the bioinformatics literature, in the context of the semantic web and grid, 'system integration' is also used in a more general sense, i.e., that of mustering a large number of data sources and providing a framework for their discovery and execution. Two such notable systems are BioMOBY-for bioinformatics web services and myGRID-for a bioinformatics semantic grid.

2.8. Investigating Agent Technology

Biology domain contains a significant amount of ontologies. A consortium was formed, comprising collaborations between many bioinformatics data sources' curators, called Gene Ontology (GO). Because semantic heterogeneity is a fundamental part of interoperability, agent systems used ontologies. There are two primary reasons for agent systems to ideal. Firstly, biology's ontological work may exploit the potential of agent technology, in relation to semantic heterogeneity. Secondly, it has been argued that ''agent-oriented approaches are well suited for developing complex, distributed systems'', which applies to bioinformatics integration systems. Agent technology has been successfully applied in the past to system integration. However, in bioinformatics systems it has mainly been used for enhanced automation and thus far only a couple of bioinformatics integration systems are based on agent technology.

3. Agent Technology

Agent technology is a new concept derived from artificial intelligence. Agent technology has its roots in multiple research areas including distributed systems, social and economic. The agent is a computer system capable of autonomous action in some environment. For real world applications single agent is not enough. So we go for multi-agent. Systems with a number of co-operating agents are called Multi-Agent Systems (MAS). Agents that are part of MAS need to possess autonomy and communication. Agent Communication Language (ACL) achieves proper communication.

3.1. Agent Communication Languages

Communication between agents is modeled as the exchange of declarative statements. By including lexis in the semantics we can use sub-set of natural language characteristics to describe a language for agents. To be more precise it can be partitioned into three layers:

- *Pragmatics*: specifies the way that an entity will express its needs or/and the effect that it wants to pass to the receiver. This layer can be thought of as the specification for information exchange. Pragmatics is referred to as the ACL. In FIPA ACL in which speech acts are called communicative acts, and KQML in which speech acts are called performatives.
- *Syntax*: used to structure the information that will be sent. The content of the message contains words that are arranged according to a structure, defined by the syntax of the language.
- *Semantics*: semantics is used to give meaning to words. It ensures that the word is associated with the correct concept. Semantics for ACLs can comprise of multiple ontologies. This layered approach helps us to work on each one part of the language independently. FIPA specifications provide both formal and informal definitions for all the communication terms used in the ACL messages

exchanged, i.e., what are the speech acts, what is their semantic meaning, what kind of expressiveness does a content language need to provide, and so on.

3.2. ACLs and Bioinformatics Integration

The three-layered approach for communicating a message is a big step towards resolving heterogeneity-which was one of the main goals of MASs. More specifically:

- A common ACL with a pre-specified content language takes care of any potential technical heterogeneity as it provides a common intermediate representation of the exchanged data and
- A common ontology resolves any potential semantic heterogeneity.

Since the purpose of an ACL is the communication between agents, one can think of it as the CQL of an integration system.

3.3. KSE and FIPA

The goal of Knowledge Sharing Effort (KSE) was to develop techniques, methodologies, and software tools for knowledge sharing and reuse at the design, implementation, and execution stages. The KSE model was intended for information exchange between databases, expert systems and any other system that could be viewed as a virtual knowledge base. Nonetheless, the main focus was to share information, which implies communication, which in turn implies a common language for communication. This led to the concept of an ACL, as we use it today, and provided KQML as the means of communicating information. The KSE model consisted of KQML, KIF, and ontolingua for the pragmatic, syntactic, and semantic layers, respectively. The KQML [4] implementations made it impossible for different systems to interoperate. The companies joined to form a forum, the Foundation for Intelligent Physical Agents (FIPA), to discuss, design, and provide specifications for agent technology. FIPA's mission statement is: ''FIPA is an international organization that is dedicated to promoting the industry of intelligent agents by openly developing specifications supporting interoperability among agents and agent-based applications'' FIPA [12] succeeded in establishing its specifications as the accepted international standards in MAS interoperability. RETSINA is a software development framework aimed at developing multi-agent intelligent systems.

3.4. The RETSINA Multi-Agent Infrastructure

REusable Task-based System of Intelligent Networked Agents (RETSINA) [6] is a multi-agent infrastructure that was developed for information gathering and integration from web-based sources and decision support tasks. Each agent in RETSINA specializes in a specific class of tasks. When the agents execute tasks or plan for task execution, they organize themselves to avoid processing bottlenecks and form teams to deal with dynamic changes in information, tasks, number of agents and their capabilities. In RETSINA, the agents are distributed and execute on different machines. The RETSINA architecture is shown in Figure 2.

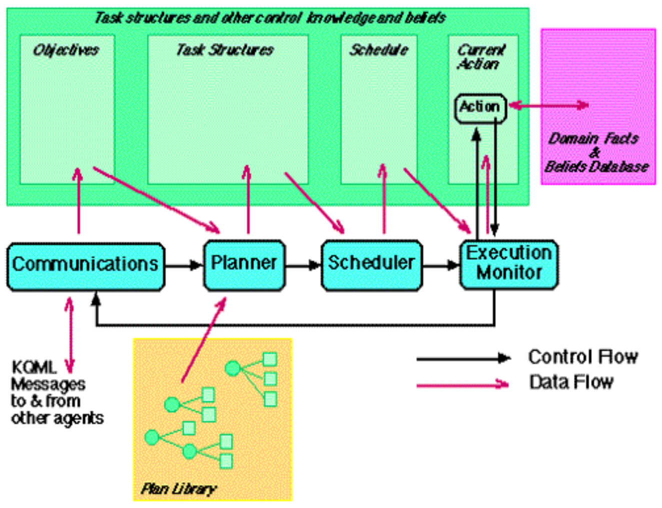

Figure 2. RETSINA architecture.

Based on models of users, agents and tasks, the agents decide how to decompose tasks and whether to pass them to others, what information is needed at each decision point, and when to cooperate with other agents. The agents communicate with each other to delegate tasks, request or provide information, find information sources, filter or integrate information, and negotiate to resolve inconsistencies in information and task models. The system consists of three major classes of agents: interface agents, task agents and information agents [11]. Interface agents interact with users receiving their specifications and delivering results. They acquire, model and utilize user preferences. The interface agents hide the underlying structural complexity of the agent system. For instance, there may be a hybrid of two types, such as interface+task agent. Task agents [2] formulate plans and carry them out. They have knowledge of the task domain, and which other task agents or information agents are relevant to performing various parts of the task. In addition, task agents have strategies for resolving conflicts and fusing information retrieved by information agents.

Information agents provide intelligent access to a heterogeneous collection of information sources. They have models of the information resources and strategies for source selection, information access, and conflict resolution and information fusion. Information agents can actively monitor information sources.

3.5. Standardisation and Bioinformatics Integration

Ideally, each data source provider would provide an interface that complies to a standard. It is here that the agent interoperability standardization efforts can be of great use. By embracing the FIPA standards data providers can just implement a FIPA-compliant agent that provides an interface to their data source.

3.6. Planning in Multi-Agent Systems

One of the popular techniques to distribute problem solving is by task sharing or task passing. Each agent tries to solve the given problem and when it reaches a task that it does not know how to handle it requests help from other agents. The basic steps in task sharing [3] are:

- *Task decomposition*: generate a set of tasks to be passed to other agents. This involves decomposing large tasks to sub-tasks that can be tackled by different agents.
- *Task allocation*: request from the appropriate agents to handle the sub-tasks.
- *Task accomplishment*: the appropriate agents each accomplish their sub-tasks-which may require further task decomposition and allocation.
- *Result synthesis*: when an agent completes a sub-task that it was responsible, it sends the result back to the requesting agent. The last will then synthesize the results into a solution, which could be a sub-solution and thus, in turn, needs to return the result to its requesting agent, until we reach the initial (root) agent that will compose an overall solution. In Figure 3 each one of the agents depicted acts as a planner-using traditional centralized planning-and co-operates with the rest to achieve a common goal.

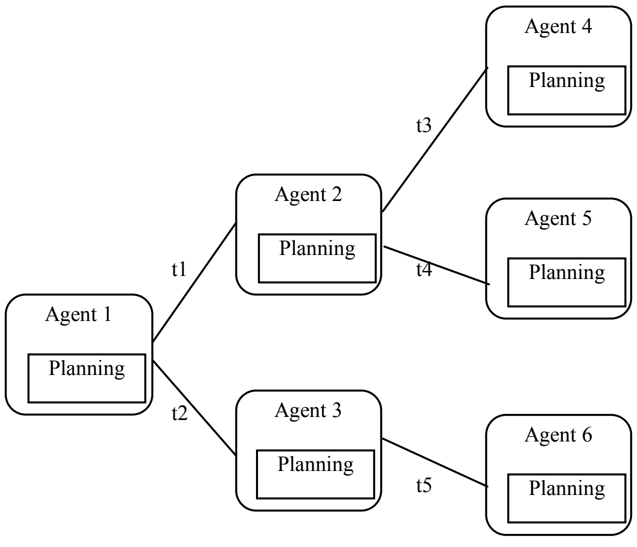

Figure 3. Example of multi-agent task sharing problem solving.

The agent that initially decomposed task acts as synchronization point for parallel execution of sub-tasks. Other types of distributed planning are: 'centralized planning for distributed plans,' 'distributed planning for centralized plans,' and 'distributed planning for distributed plans.'

3.7. Planning and Integration

The steps of task sharing are to the integration steps. If we consider a task to be a query, as expressed in the CQL, then the two procedures are identical. This implies that significant synthesis is possible between technologies developed for these activities.

3.8. Adjustable Autonomy

Adjustable autonomy means dynamically adjusting the level of autonomy of an agent depending on the situation. It is beneficial to have a mechanism that enables control over the behavior of a dynamic and complex distributed system so as not to feel uncertain about the quality of the results.

3.9. Adjustable Transparency

The issue of the results' quality is more important to the user. We already suggested that we need a way to adjust the integration system's transparency. Adjustable autonomy has as a result for the user to gain some control over the behavior of the agents. If the agents' functionality is to integrate then adjusting their autonomy is exactly what we need to better manage the system's transparency.

4. Agents and Integration

Most agent integration systems use the mediation approach. An agent acts as a mediator, usually called Mediator Agent (MA). This agent has access to the CDM, which could be represented using any representation language-including the content language of the ACL. Additionally, a number of agents will act as wrappers, usually called resource agents. The ACL could be used as the CQL.

4.1. Distribution, Autonomy, and Heterogeneity

Agents naturally cover the fundamental aspects of integration.

- *Distribution*: a MAS is naturally distributed.
- *Autonomy*: agents are designed with the assumption that software entities are autonomous. Also, FIPA's interoperability standardization efforts will ensure communication between agents created from many different vendors, organizations, or research groups.
- *Heterogeneity*: a common communication language and a common message content language deal with technical heterogeneity while sharing ontology handles the semantic differences.

4.2. Legacy Systems and Wrappers

Nonagent systems could be made a part of an agent community if they were able to communicate using an ACL and ACL acts as the CQL, with wrappers translating from the ACL to the local query language and vice versa. Figure 4 shows three possible ways to do this-called agentification process [7].

A transducer is an agent that knows how to translate requests from an agent system-other agents-to the non-agent system's interface and vice versa. In system integration, the term 'wrapper' we mean either the transducer or the wrapper approach. Finally, one could also rewrite the non-agent system according to an agent paradigm. That amounts to a lot of programming work but one could potentially enhance the system's efficiency and capabilities. FIPA defined an agent software integration specification which is concerned with how agents can connect to and make use of external software an system, that is systems that are external to and independent of an agents execution model-the transducer approach to wrapping. In agent terminology, wrappers are usually called Resource Agents (RAs).

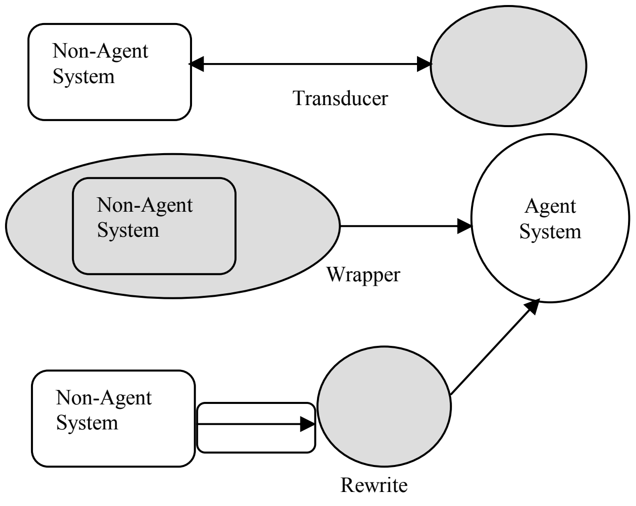

Figure 4. Three approaches to agentification.

4.3. Adding Data Sources

In accordance with FIPA specifications all (nonagent) software systems (data sources) should be described by software descriptions to list the properties of the software system. FIPA supports another agent role: an agent that brokers a set of software descriptions to interested agents. New data sources can be added dynamically to the system just by providing a software description for the resource to the request broker.

4.4. Bioinformatics Integration and Agents

Agent technology is appropriate for the complex integration systems, particularly in bioinformatics. In summary:

- The layered approach of an ACL provides a flexible common medium to represent knowledge among agents.
- The ACL and the ontologies deal with the technical and semantic heterogeneities, respectively.
- RAs can wrap data sources. In addition using the FIPA 'agent software integration' specification new data sources can be added dynamically to the system.
- The adoption of agent (FIPA) standards help in making the first steps towards solving the sociological problem.
- The most popular multi-agent planning technique, task sharing, is almost identical to integration using mediation, and thus they could easily be combined.
- Adjustable Autonomy provides a potential solution to the problem of the confidence of the results in bioinformatics integration.
- Bioinformatics integration systems are complex distributed systems, which makes use of agent technology a good choice [8].

4.5. Agents, Web Services, and the Grid

Making web services autonomous and able to reason to accomplish their goals makes web services more like agents and rendering the latter widely available to the public, via the internet, makes agents more like web services. Similarly, grid technology has a lot to gain from the high-level abstractions that the agent paradigm has to offer; use of agent protocols, negotiation, etc. All three technologies provide service-oriented functionality, which implies similarities, but each one can contribute its more unique attributes:

- Grid technology offers a large-scale distributed infrastructure,
- Web technology provides formatting standards to represent data and knowledge,
- Agent technology offers autonomy and advanced communication between peers, bridging the three technologies.

Currently only agent technology is self-sufficient enough to provide the advanced integration facilities. Agents are situated in a flexible and scalable distributed environment (Agent Platform [5]). Agents provide a common communication layer necessary for system integration. Also, they are good at addressing changes dynamically, which is an important asset for integration system applied in changing biology domain. The mutli-agent system demonstrating the above features is RETSINA, discussed above.

5. Conclusions

Biology is a knowledge-intensive science and a large number of data sources are publicly available. Data sources are integrated to enable higher-level questions to be answered by combining their data. Integration is a complex task aiming to provide a unified view of the underlying resources, while eliminating potential technical and semantic heterogeneity. The mediation approach to integration is widely used and most bioinformatics integration systems make use-in different degree-of it. Agent technology is a multi-disciplinary research field combining work from distributed systems, AI, social and economic sciences. Ever since its conception, its goal has been to develop techniques, methodologies, and software tools for knowledge sharing and reuse. Knowledge sharing is fundamental to integrating heterogeneous data sources, and as such, agent technology has much to offer to system integration. This becomes clearer after a detailed examination of agent properties and their usefulness for coping with bioinformatics integration challenges.

References

[1] Achard F., Vaysseix G., and Barillot E., "XML, Bioinformatics and Data Integration," *Bioinformatics*, vol. 17, no. 2, pp. 115-125, 2001.

[2] Davidson S., Overton C., and Buneman P., "Challenges in Integrating and Biological Data Sources," *Journal of Computational Biology*, vol. 2, no. 4, pp. 557-572, 1995.

[3] Durfee E., "Distributed Problem Solving and Planning," *in Weiss G. (eds.), Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, Cambridge, MIT Press, pp. 121-164, USA, 2000.

[4] Finin T., Weber J., Wiederhold G., Genesereth M., Fritzson R., and McGuire J. "Speci.cation of the KQML Agent Communication Language," *Technical Report*, DARPA Knowledge Sharing Initiative, External Interfaces Working Group, 1993.

[5] "FIPA Agent Management Speci.cation," available at: www.pa.org/specs, 2002.

[6] Foster I., Kesselman C., and Tuecke S., "The Anatomy of the Grid: Enabling Scalable Virtual Organizations," *International Journal of Supercomputing Applications*, vol. 15, no. 3, pp. 200-222, 2001.

[7] Genesereth M. and Ketchpel S., "Software Agents," *Communications of the ACM*, vol. 37, no. 7, pp. 48-53, 1994.

[8] Jennings N., "An Agent-Based Approach for Building Complex Software Systems," *Communication of the ACM*, vol. 44, no. 4, pp. 35-41, 2001.

[9] Karp P., Paley S., and Zhu J., "Database Verification Studies of SWISSPROT," *Journal of Biomedical Informatics*, vol. 37, no. 3, pp. 205-219, 2004.

[10] Sheth A. and Larson J., "Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases," *ACM Computing Survey*, vol. 22, no. 3, pp. 183-236, 1990.

[11] IEEE web site for Agents, www.fipa.org.

[12] The Intelligent Software Agents Lab Carnegie, Mellon University, www.intelligent-agents.com.

Kulathuran Shunmuganathan is a Phd student, completed MS from BITS Pilani, and has passed ME with distinction.. He is currently working as assistant professor in Noorul Islam College of Engineering.

Kumar Deepika is a prefinal year BE (CSE) student of Noorul Islam College of Engineering. Her topics of interest are artificial intelligence, DBMS, and image processing.

Kumaradhas Deeba is a prefinal year BE (CSE) student of Noorul Islam College of Engineering. Her topics of interest are artificial intelligence, DBMS, and image processing.