Context Repository Architecture for Smart Collaborative Environment

S M Faisal †, Mohammad Rezwanul Huq †, A. S. M. Ashique Mahmood †

† Islamic University of Technology (IUT), Board Bazar, Gazipur 1704, Bangladesh

Summary

Due to the advancement of technology, smart sensors and smart objects are becoming popular now-a-days. These smart sensors and objects have some computation power so that depending on the change of the environment they can take actions accordingly. Since the use of portable devices like PDA, cellular phone is increasing; users also would like to interact with the environment through these. Moreover, for performing complex operations, we need to have context middleware that can analyze the contexts and provide necessary services to users. Therefore, it is obvious that for all of these cases, a repository of contexts is badly needed to store the generated contexts as well as to retrieve those contexts whenever necessary. In this paper, we propose novel context repository architecture for smart collaborative environment where we have a bunch of sensors deployed, smart objects, handheld devices as well as our backend middleware. We design our context repository architecture considering all of these issues. Our context repository architecture contains solution for the repository in smart sensors, smart objects, handheld devices as well as backend middleware. We design our repository architecture in such a way so that it will be lightweight for smart sensors, objects and handheld devices since these elements will perform simple operations. In the backend middleware, we store all the generated contexts to support complex reasoning and analysis which will be carried out for performing complex operations. Furthermore, our context repository also provides some extended functionalities which can be used by the middleware to provide context-aware services to users. Depending on our design, we will try to implement our prototype in future which will demonstrate the viability of our architecture surely.

Key words:
Context, Repository, Smart sensors, Smart Objects, Collaborative Environment.

1. Introduction

There exists numerous low-level context and high-level context in Ubiquitous Environment. Context Repository is a persistent storage of the contexts and support query to applications. The overall context repository can be composed of multiple levels of repository for each type of entities like smart sensors, smart objects, handheld devices as well as the central middleware. Centralized Repository is placed at backend server which has enough computational power and large capability of storage. On the other hand, for smart sensors, objects and handheld devices, we need a repository with small computational power and storage space. The repository must be light weight and the contexts should be stored in a distributed space because of the characteristics of handheld devices, sensors and objects in SCO environment.

Figure 1 shows a typical example of smart collaborative object (SCO) environment. There are some passive sensors connected to active sensors. These active sensors can co-operate with each other. The term 'co-operate' means that they can communicate with each other by means of sharing information. These smart co-operative sensors can also communicate with other smart objects and handheld devices. Again, smart object and handheld devices can communicate with each other. The contexts generated from users' activities will be stored in the backend server as well as some important contexts will be stored in smart objects and handheld devices. These contexts contain useful information regarding users or these are frequently shared contexts.

1.1 Major Challenges

The design and development of a context repository for smart collaborative object environment faces a number of challenges. These challenges are:

1. Low power and limited memory of handheld devices: The main goal of smart collaborative object environment is to provide services to the users in a context-sensitive way. It is very much expected that user will have handheld devices like PDA, mobile phones those have limited power resource and memory. Therefore, the context storage should be light weight so that the most essential information and context should be stored in users' handheld devices.
2. Location-centric: The entities in SCO environment are mobile in nature. This mobility adds location as a new dimension to applications that does not typically play a role in stationary scenarios. Consider a system that can answer questions such as "find the drugstores within 2 miles of my current location". Such a system must track

Manuscript received December 5, 2008
Manuscript revised December 20, 2008

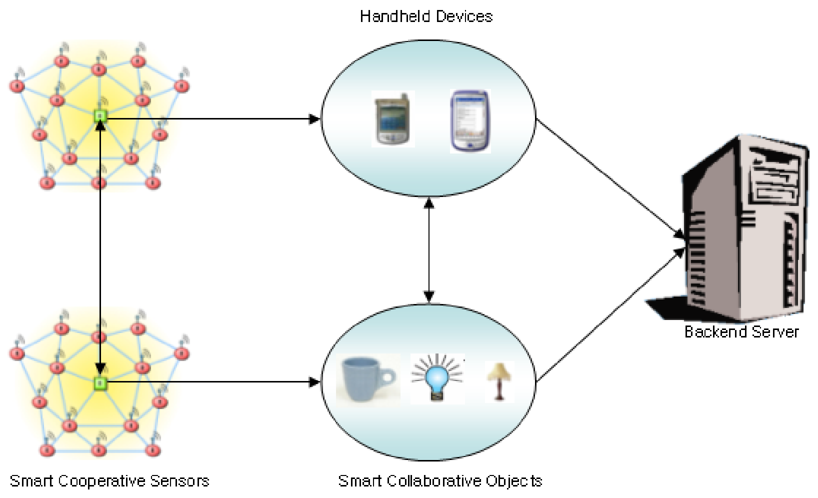

Figure 1: Smart Collaborative Environment

the location of the current user and be able to access information based on relative locations and distances.

3. Machine learning: Whether the system obtains its context information from sensors, user input, PIM (personal information management) applications, or some combination of these, it must perform a good deal of processing over the data in order to be able to accurately assess the state of the environment and the intensions of the user. Thus, context-aware applications impose demanding requirements for machine learning techniques.
4. Context reasoning: This is very much necessary to get high level context from low level context for effective context-sensitive information in SCO environment. The provision of context reasoning will be extremely efficient in order to be able to interact with the user in a useful and unobtrusive manner.
5. Diversity of contexts: In SCO environment, there should be a lot of heterogeneous devices. Those devices will produce a number of contexts which will be different in nature. For designing and developing a successful context repository in SCO environment we need to identify some common framework to store that context information. This common framework should be understandable by all the devices and agents interacting in the system.
6. Objects are distributed: Smart collaborative objects will be distributed over a particular region. Users will be also roaming around over that region. Therefore centralized storage system will be in no use in this environment. We must have to design the context repository in distributed manner to satisfy the queries issued by the distributed objects and users.

1.2 Our Objectives

Our main concern is to build a context repository for the SCO environment. In the course of building this context repository we need to meet the following objectives too. These are listed below.

1. Providing a means for context storing in smart objects and smart sensors using tuple space.
2. Ontology driven, lightweight context storing mechanism in users' handheld devices with agent based communication.
3. Build a central context repository in backend server to support complex reasoning and querying.

The rest of the paper is organized as follows. In section 2, we have discussed the possible existing approaches for context modeling and representation. In section 3, we have reviewed the existing methods for ontology-based database. In section 4, we propose our context repository architecture for smart collaborative environment. Finally, we conclude in section 5 with some hints of our future work.

2. Context Modeling and Representation

Context Modeling is one of the most important jobs in designing an application for a smart space. The most relevant modeling approaches classified by the scheme of data structures which are used to ex-change contextual information have been described below.

1. Key-value Models: The simplest data structure that represents the context information by a value provided via an environment variable to an application.

2. Markup scheme Models: All markup based models use a hierarchical data structure consisting of markup tags with attributes and content.
3. Graphical Models: Context is modeled by using a very well know modeling tool Unified Modeling Language due to its generic structure. Various approaches exist where contextual aspects are modeled in by using UML, e.g. Bauer in [1].
4. Object oriented Models: It uses the power of object-orientation (e.g., encapsulation, reusability, inheritance) to represent different context types (such as temperature, location, etc.) and encapsulate the details of context processing and representation on an object level.
5. Logic based Models: This model defines context as fact, expression together with rules on which new concluding expressions or facts can be derived from a set of existing facts and expressions. Contextual information adding, updating and deleting are done from logic based system in terms of facts or inferred from rules in the system.

Ontology based Models: Ontologies represent a description of the concepts and relationships. Therefore, ontologies are a very promising instrument for modeling contextual information due to their high and formal expressiveness and the possibilities for applying ontology reasoning techniques. Various context-aware frameworks use ontologies as underlying context models.
In our solution, we have used ontology based models for context modeling purpose. Therefore, we need some means to represent this ontology-based model. Next, we provide the illustration for ontology based context representation.

Languages like XML that define structure of a document, but lacks semantic model, are not enough for describing ontologies- intuitively an XML document may be clear, but computers lack the intuition. In recent years ontology languages based on Web technologies have been introduced. DAML+OIL [2], which is based on RDF Schema [3], is one such language. It provides a basic infrastructure that allows machines to make simple inferences. Recently, DAML+OIL language was adopted by World Wide Web Consortium (W3C), which is developing a Web Ontology Language (OWL) [4] based on DAML+OIL. Like DAML+OIL, OWL is based on RDF Schema [3], but both of these languages provide additional vocabulary—for example relations between classes, cardinality, equality, richer typing of properties, characteristics of properties, and enumerated classes—along with a formal semantic to facilitate greater machine readability. The OWL language has a quite strong industry support, and therefore it is expected to become a dominant ontology language for the Semantic Web. OWL is built on the top of RDF & RDFS. It is much more expressive than RDF & RDFS with hierarchies and relationships between re-sources. It has many predefined classes and properties for ontologies that can be reused. It supports a wide variety of development tools.

3. Ontology based Database

Ontology is a data model that represents a domain and is used to reason about the objects in that domain and the relations between them. Ontologies are used in artificial intelligence, the semantic web, software engineering and information architecture as a form of knowledge representation about the world or some part of it. It is a data model used for implementing semantic web that is a vision of web pages that are understandable by computers, so that they can search websites and perform actions in a standardized way. Semantic web provides a common framework that allows data to be shared and reused across appli-cation, enterprise, and community boundaries. The motivation of using ontology driven approach for context repository is three folds.

- Knowledge sharing: Ontology provides a common framework for information sharing. We used to define standard vocabularies at the time of ontology modelling. These set of standard vocabularies could be easily understood by different smart collaborative objects present in the environment. Therefore, the sharing of knowledge could be served in a meaningful way.

- Knowledge reuse: we can build large-scale context ontology by reusing knowledge of several domain specific ontologies. We need not to start from scratch to build up the large-scale context ontology. Objects in the environment can also reuse the knowledge as it is providing a common framework to define contexts.

- Logic inference: Systems built on the top of SCO environment need inference mechanism. Generating high level context from low level context is very much important for smart collaborative object environment. This inferring mechanism helps the proper execution of the system that gives necessary context-sensitive information to the user thus providing necessary services to users.

The database that stores data and the ontologies describing their meanings in the same database is known as Ontology-based database (OBDBs). Figure 2 shows different schemes used so far. Ontology-based data represents ontology individuals. As for example, instance of ontology classes. To ensure a high performance of queries on top of OBDBs efficient representation of

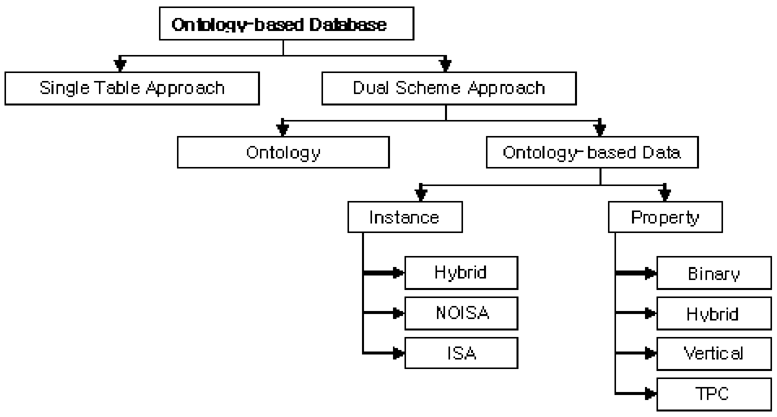

Figure 3: Different Schemes for Ontology based Database

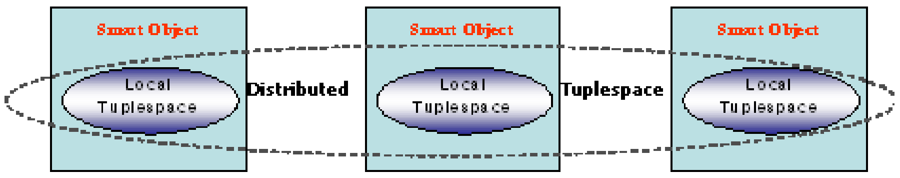

Figure 2. Distributed Tuplespace for Smart sensors and smart objects

ontology-based data is needed. Two main representation schemes have been proposed. They are:

- Single table approach
- Dual schemes approach

In the single table approach [5-8], the description of classes, properties and their instances are stored in a single table called vertical table [5]. Dual scheme approach consists in storing separately ontologies and instance data in two different structures, called ontology and data, respectively [9-11]. In the dual scheme approach, instances and their properties values are also stored separately. Another new technique of storing data has been proposed named as table per class representation approach [12]. This schema consists of all the class applicable properties that are used at least by one instance of the class.

4. Our Proposed Architecture

For building a successful context repository, we must have to provide a means to interchange contextual information among smart sensors, smart objects. Furthermore, we have to ensure that some user-oriented context will be stored in users' handheld devices as well as each and every context will be stored in the backend server. In the coming portion, we will talk about our solution involving distributed tuple space for smart objects and sensors, ontology-based lightweight database for users' handhelds and context repository for backend server.

4.1 Distributed Tuple Space for Smart Sensors and Smart Objects

Distributed tuplespace (Figure 3) provides a platform for storing and exchanging sensory data and con-textual information. It distributes the originally centralized tuplespace structure among different nodes. In our approach, we will distribute tuplespace among smart objects.

Here are some more details on distributed tuplespace.

- Tuple spaces are a realization of the associative memory aka the blackboard architecture for storage [13].
- A data structure shared by all the objects cooperating with each other.
- Each node contributes a portion of its local memory to the shared data space.
- Operations on shared data space reflect to all local spaces.

4.2 Ontology-based Lightweight Database for users' handhelds

The main component of our ontology driven context repository is entity. User, meeting room, class room,

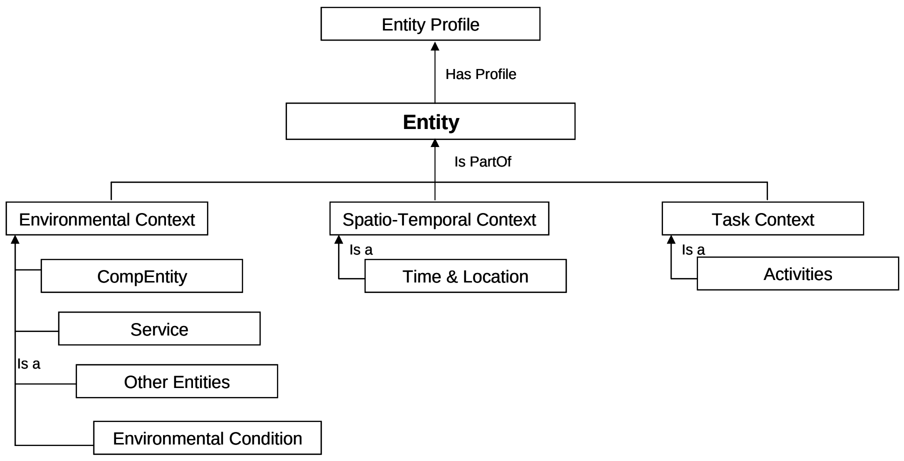

Figure 4: Concepts of ontology for an Entity

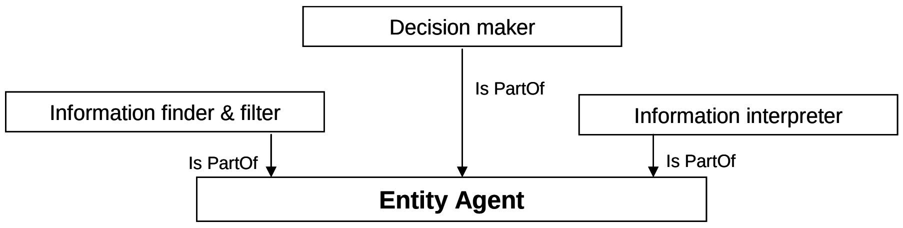

Figure 5: Block diagram of an Entity Agent

restaurant etc. is examples of different kinds of entities. In our definition of an entity computer or any other computational devices are not an entity itself rather it belongs to the environmental context of that particular entity. The contextual information for a particular entity can be roughly subdivided into the following categories.

- Environmental context: This part captures the entity's surroundings like things, services, agents, environmental condition, users and information accessed by users.
- Spatio-temporal context: This type of context is concerned with attributes like time, location and movement.
- Task context: This describes what the entity is doing, it can describe the entity's goals, activities etc.

This contextual information can be efficiently managed by ontology. Ontology provides a common framework for understanding this information. Here we give an example of ontology with related concepts for an entity (Figure 4).

Each entity in our smart collaborative object environment will communicate with each other by means of its agent. The goal of agents is to reduce user work and information overload.

Three fundamental roles of agents (Figure 5) are essential to smart collaborative environments: information finder/filter, information interpreter, and decision maker.

1. An information finder and filter helps users to find out the requested information and filter out unnecessary information according to a specified user task. The agent will provide a reasonable number of choices to users or suggest an alternative option if the requested information items are not available.
2. An information interpreter can access and convey information from one side to the other. In distributed network environments, heterogeneous data models and systems can not communicate directly.
3. A decision maker can make decision autonomously based on its own knowledge and user-defined rules. An agent can collect and analyze information according to specific events, such as the migration and linkages of objects and components, and then make an optimal decision based on the rational rules defined by users or other agents.

Agent for a particular entity is responsible for knowledge sharing with other agents of the smart collaborative object environment. Actually the agent retrieves contextual information from the ontology based database of each entity. As our primary concern is to make the database a

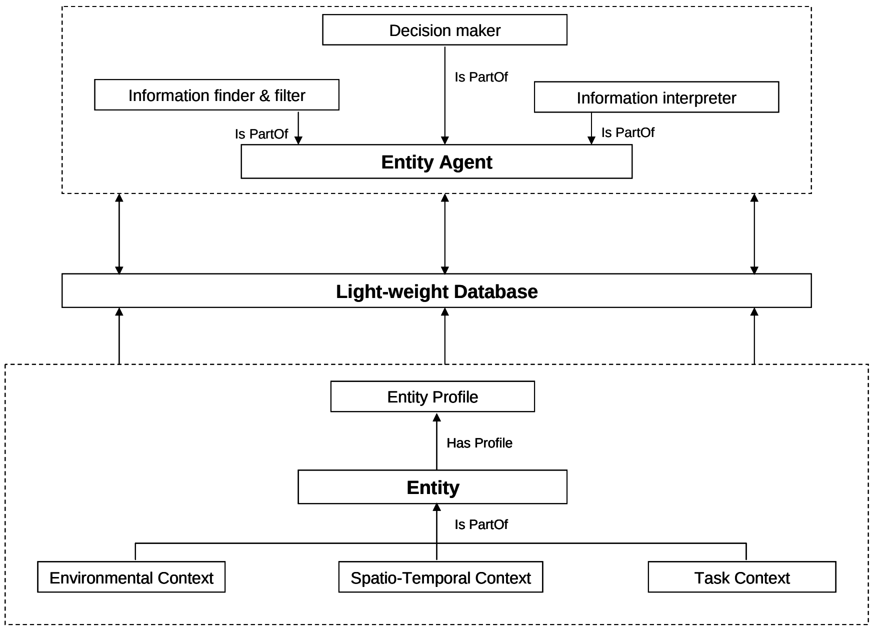

Figure 6: Interaction among Entity, Database and Agent

lighter one, we will use one of the light-weight database techniques like SQLite [14], HSQLDB [15], Mini SQL [16]. This light-weight database holds the ontology and ontology-based data for a particular entity. By means of agent of that entity, this data can be shared with other agents thus leads to the collaboration among different entities (Figure 6).

4.3 Context Repository in Backend Server

In the backend server, context will be collected in a timely manner from smart sensors, smart objects as well as users' handheld devices. Some principle features of backend server includes

- Support complex reasoning (e.g. Bayesian reasoning).
- Support prediction of user and object behavior.

Figure 7 depicts the context repository architecture in backend server. Context linker links data from different sources to the relevant information entity (e.g. Context location linker, user linker, device linker). Context merger integrates context from different sources, also identifies which parts of context is changed. Agent using context will subscribe the notification schedule to context middleware and will re-ceive notifications when relevant part of context is changed. While receiving new context, Agent sends it to reasoning engine to specify the tasks needed to automatically perform for users to adapt new context. Tasks will be sent to appropriate application agents' services.

4.3.1 Interface with Sensors and Other Components in the System

Figure 8 depicts the interface among context repository in backend server and other components in the system like smart sensors, objects and different reasoning engine. Our backend repository collects con-texts through Data Acquisition Manager. It is the interface between repository and the smart sensors, ob-jects. In our repository, contexts, rules and codes for different reasoning engine will be stored. Depend-ing on the use of reasoning engine, the code for that particular reasoning engine will be dynamically instantiated. Moreover, rules are stored too for rule based reasoning mechanism. CRC (Contexts, Rules and Codes) Provider acts as an interface between repository and different reasoning mechanism used for a particular type of application. Our repository in backend server thus satisfies all the requirements for SCO environment.

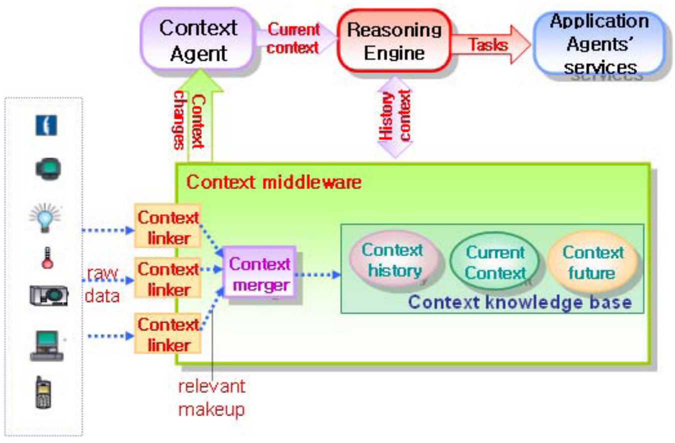

Figure 7: Context Repository Architecture in Backend Server

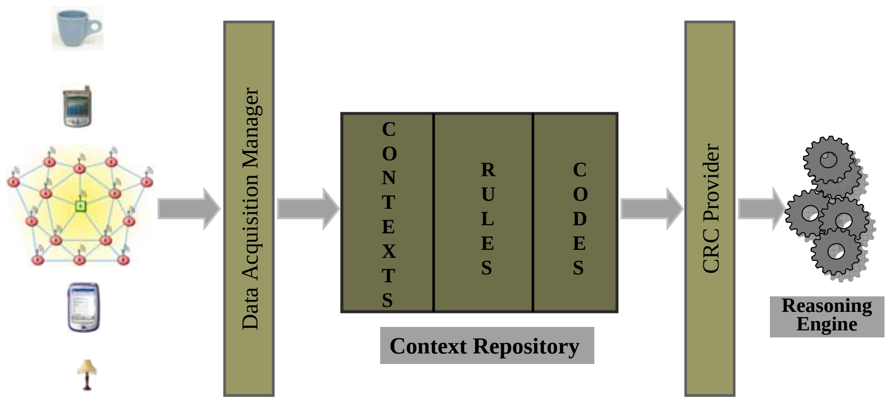

Figure 8: Interface among Context Repository in the Backend Server and Other Components

5 Conclusion

Though our proposed context repository scheme is still in an abstract level of thinking, it can satisfy the requirements for SCO environment. We are going to use one of the light-weight databases that will take out a huge overhead of computational power and memory. Moreover, each entity will hold the on-tology for itself that introduce the provision of distributed modular ontologies [17, 18]. Ontology pro-vides a data model that can serve as a common framework for context storing thus make knowledge shar-ing and reuse effective. Logic inference is very much possible by using the Semantic Web Rule Lan-guage (SWRL) [19] and a rule engine like Java Expert System Shell (JESS) [20]. Our next step is to further investigate the strengths and weaknesses of the proposed architecture for context storing and once finalize the architecture then, to implement the context repository for SCO environment.

References

[1] 1. BAUER, J. Identification and Modeling of Contexts for Different Information Scenarios in Air Traffic, Mar. 2003. Diplomarbeit.
[2] J. Hendler and D. L. McGuinness. The DARPA Agent Markup Language. IEEE Intelligent Systems, 15(6):67–73, 2000.
[3] D. Brickley and R.V. Guha. RDF Vocabulary Description Language 1.0: RDF Schema. 2003. W3C Working Draft, work in progress, available at: http://www.w3.org/TR/2004/REC-rdf-schema-20040210/.
[4] WAPFORUM. User Agent Profile (UAProf). http://www.wapforum.org.
[5] R. Agrawal, A. Somani, and Y. Xu. "Storage and querying of e-commerce data" In Proc. VLDB'01, pages 149–158, 2001.
[6] B.McBride. "Jena: Implementing the RDF model and syntax specification." In Proc. of the 2nd Intern. Workshop on the Semantic Web, 2001.

[7] S. Harris and N. Gibbins. “3store: Efficient bulk RDF storage.” In Proc. of the 1st Intern. Workshop on Practical and Scalable Semantic Systems (PSSS’03), 2003.
[8] L.Ma, Z. Su, Y. Pan, L. Zhang, and T. Liu. “Rstar: an RDF storage and query system for enterprise resource management.” thirteenth ACM international conference on Information and knowledge management, 2004:484 – 491.
[9] S. Alexaki, V. Christophides, G. Karvounarakis, D. Plexousakis, and K. Tolle. “On storing voluminous rdf descriptions: The case of web portal catalogs.” In Proc. ofWebDB’01 (co-located with ACM SIGMOD’01), 2001.
[10] J. Broekstra, A. Kampman, and F.V. Harmelen. “Sesame: A generic architecture for storing and querying rdf and RDF schema.” In Proc. of the First Inter. Semantic Web Conf., pages 54–68, 2002.
[11] Z. Pan and J. Heflin. “Dldb: Extending relational databases to support semantic web queries.” ISWC’2003, 2003.
[12] H. Dehainsala, G. Pierra, L. Bellatreche, “OntoDB : An ontology based database for data intensive applications”, DASFAA 2007.
[13] Wikipedia: http://en.wikipedia.org/wiki/Tuple_space.
[14] SQLite: http://www.sqlite.org/
[15] HSQL: http://hsqldb.org/
[16] Mini SQL 2.0: http://www.hughes.com.au/library/msql/manual_20/
[17] Y.Qu, Z.Gao, “Interpreting Distributed Ontologies”, WWW 2004, USA.
[18] Jie Bao, V.Honavar,”Adapt OWL as a Modular Ontology language (a position paper)”
[19] SWRL: A Semantic Web Rule Language Combining OWL and RuleML, http://www.w3.org/Submission/SWRL/
[20] Jess: The rule engine for the Java platform, http://herzberg.ca.sandia.gov/jess/

S M Faisal received his Bachelor of Science degree from Islamic University of Technology (IUT), Bangladesh on September 2005. He joined as Lecturer at the same university in the Computer Science & Information Technology (CIT) department in December 2005 and is still working at the same university. His research interest includes Collaborative Environment, Data Mining, Artificial Intelligence and Information Security. He is a member of the Institution of Engineers, Bangladesh (IEB).

Mohammad Rezwanul Huq received his Master of Computer Engineering degree from Kyung Hee University, Korea on February 2008. Currently, he is serving as Lecturer, CIT Dept. in Islamic University of Technology (IUT), Bangladesh. Earlier he completed his B.Sc. in CIT from Islamic University of Technology (IUT), Bangladesh on September 2004. Later, he joined as Lecturer at the same university in Computer Science & IT department from December 2004 and still serves at the same university. His research interest includes Collaborative Environment, Ubiquitous Computing, Semantic Web and Data Mining. He is a member of the Institution of Engineers, Bangladesh (IEB). He also served as a program committee member for the 2008 Semantic Web and Web Services Conference held in Las Vegas, USA on July 2008.

A. S. M. Ashique Mahmood received his Bachelor of Science degree from Islamic University of Technology (IUT), Bangladesh on September 2006. He is working as a lecturer in the Dept. of Computer science and Information Technology(CIT) at Islamic University of Technology (IUT) since his joining in December 2006. He worked in the development project of Banglapedia (digital version), the encyclopedia of Bangladesh. He is mainly focused in areas such as Parallel & Distributed Computing, Collaborative Environment, Artificial Intelligence and Databases.