Modeling ebXML Registry Service Architecture

Ivan Magdalenic, Ivo Pejakovic, Zoran Skocir,
Mihaela Sokic
Faculty of Electrical Engineering and Computing
University of Zagreb
Unska 3, HR-10000 Zagreb, Croatia

Marina Simunic
Agrokor Group
Trg Drazena Petrovica 3, HR-10000 Zagreb, Croatia

Abstract— **Department of Telecommunications, Faculty of Electrical Engineering and Computing, University of Zagreb works on the research project named "Networked Economy", supported by Croatian Ministry of Science and Technology. The project is also recognized by corporate domain and was made a part of an e-business initiative by the Agrokor group. The Agrokor's intention is to use the ebXML Registry/Repository layer as the central point of integration among the many Agrokor's affiliates and partners. We have considered different approaches toward ebXML specifications describing Registry/Repository layer, not only to conform to specifications, but also with vision that careful architecture modeling will enable modularity and flexibility of our solution. This could be vital, as we must bear in mind that specifications are surely going to be improved over certain period of time. This paper describes modeling ebXML Registry/Repository architecture and concentrates on its Registry Service part.**

Index terms -- *B*2B, ebXML, UML, XML, RIM, Registry Service.

I. Introduction

The ebXML architecture is only globally recognized framework that has the power and technical predispositions to become globally accepted e-Business platform. The rapid development of specifications and expertise of involved experts are combined with feedback from industry leaders. Implementations are also developing, but with much slower pace, considering its possibilities as a global standard.

An implementation is hosted at the Center for e-Commerce Infrastructure Development (CECID) in Hong Kong [1]. Its development was started by Sun Microsystems Inc. and continued as an open-source project hosted at Source Forge. Many other open-source and commercial implementations are undergoing, varying implementation details and platforms but conforming developed specifications.

A research project named "Networked Economy", supported by Croatian Ministry of Science and Technology is preformed on Department of Telecommunications, Faculty of Electrical Engineering and Computing, University of Zagreb, Croatia. In the course of research we are especially considering e-business solutions [2]. We are focused on e-business frameworks and conceptual prerequisites for building necessary e-business infrastructure, to ensure smooth transition of developing country economy towards e-business. Having acknowledged the importance of e-business, we are carrying out the research aimed primarily at defining the strategy for implementation of e-business in Croatia. In this sense, a series of projects are conducted, with ebXML standard adopted as a technical infrastructure. We have found that ebXML as a standard for electronic business offers the solutions to variety of recorded problems.

The e-business part of a "Networked Economy" project is also recognized by corporate domain and was made a part of an e-business initiative by the Agrokor group. The Agrokor group intention is to use the ebXML Registry/Repository as the central point of integration among the many Agrokor's affiliates and partners, which will lay the foundation needed to migrate the current business processes to an e-business platform. We have therefore considered different approaches toward ebXML specifications describing Registry/Repository layer. Our goal was not set only to achieve fully specifications conformance. We presume that specifications are going to be improved over certain period of time. So the main goal was shifted from meeting conformance requirements toward careful architecture modeling that will enable modularity and flexibility of our solution. The Registry Service architecture is presented in this paper. We consider that presented architecture of the ebXML Registry/Repository implementation will be our contribution to better performance of Registry Service.

Section II describes the role of ebXML Registry/Repository specification. Section III describes implementation of the Registry Information Model (RIM). Section IV describes implementation of the security in the ebXML Repository. Section V describes Registry Services implementation. Conclusion in section VI is followed by a list of references.

II. Role of ebXML Registry/Repository in ebXML Infrastructure

E-business has given companies the prospect of eliminating paper documents, reducing costs, and improving efficiency by exchanging business information in electronic form. For over 25 years Electronic Data Interchange (EDI) is used for conducting highly structured inter-organization exchanges, such as for making purchases or initiating loan requests [3]. Many companies, however, find EDI expensive and difficult to implement.

Large-scale adoption of e-business practices in small countries with Small and Medium-Size Enterprises (SME) prevailing environments challenges standard bodies and industry groups.

In the last few years, the Extensible Markup Language (XML) has rapidly become the first choice for defining data interchange formats over the Internet. XML is an open and freely available document from the World Wide Web Consortium and has the support of the world's leading technology companies [4]. XML also supports Unicode that enables the display and exchange of most of the world's written languages.

ebXML, build on the experience and strengths of existing EDI knowledge, recognized emerging XML efforts and adopted XML as basis.

To facilitate the process of conducting e-Business, potential trading partners need a mechanism to publish information about the business processes which they support along with specific technology implementation details about their capabilities for exchanging business information. In ebXML infrastructure, this is accomplished through the use of specific XML document which can be universally understood by ebXML compliant trading partners.

ebXML Registry and Repository are a central part of the ebXML architecture that provides [5]:

- A stable store where information submitted by a Submitting Organization is made persistent;
- Metadata about each repository item;
- A set of services that enable the sharing of information stored in Registry between Trading Partners.

The term repository item is used to refer to an object that has resides in a repository for storage and safekeeping (e.g., an XML document, DTD, UML models, information about parties or even software components). Every repository item is described in the Registry by a registry entry. The term "registry entry" is used to refer to an object that provides metadata about a repository item [6].

The ebXML Registry and Repository are separated. The reason for separation of Registry and Repository lays in intention of the ebXML technical committees to achieve maximum flexibility in description and query of the data. So, each item in the Repository is associated with a Registry object that represents it. This object is of the ExtrinsicObject class [6]. The Repository item is always retrieved via its ExtrinsicObject. Registry contains pointers to Repository items, which are metadata, not real instances of data (Fig. 1).

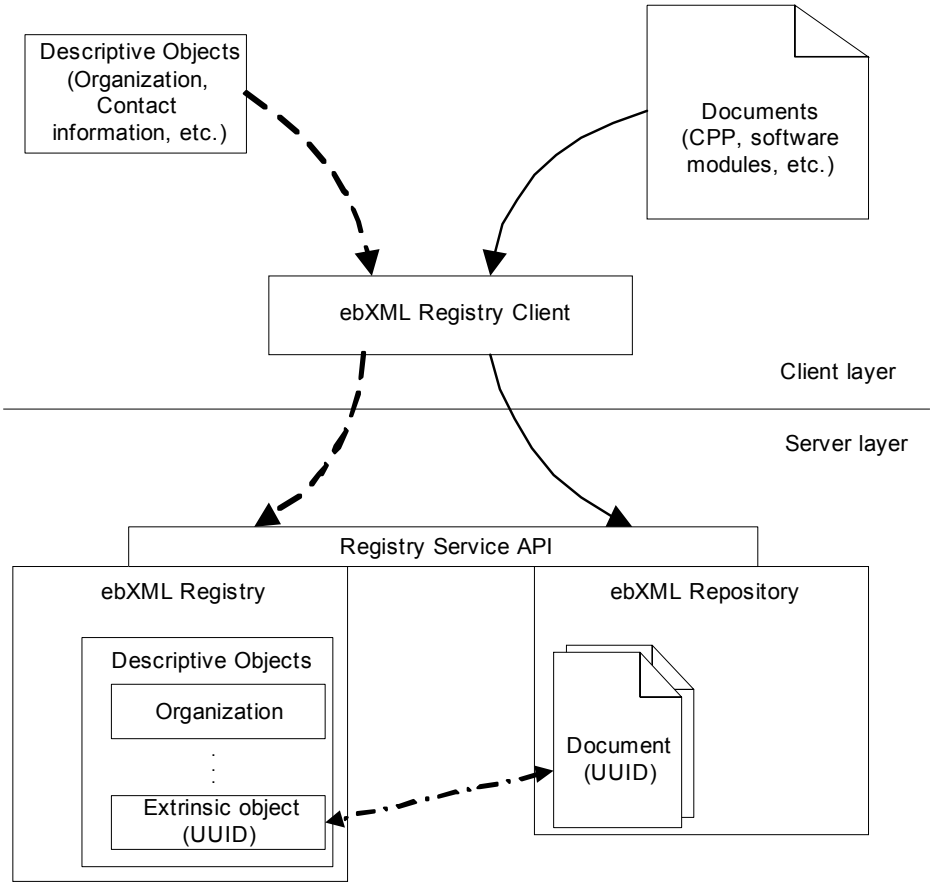

Figure 1. ebXML Registry/Repository relations

ConTEL 2003, ISBN: 953-184-052-0

Access to the ebXML Registry is provided through Interfaces (APIs) exposed by ebXML Registry Services. Registry Services exist to create, modify, and delete repository item and their metadata [7].

The information model for ebXML Registry – ebXML Registry Information Model (RIM) defines:

- What types of metadata are stored in the Registry;
- How stored objects are organized in the Registry (relationships among metadata Classes).

EbXML RIM is specified in [6]. This specification provides a blueprint or high-level schema for the ebXML Registry, particularly for implementers. UML diagrams are used as a way to concisely describe concepts. However, they are not intended to convey any specific implementation or methodology requirements.

Both centralized and decentralized repository models are acceptable. We are concerned with development of Registry Service that will enable our Registry to be included into schema of Registries, although its current usage is in form of a single centralized Registry. Typical items in the Repository include the CPP (Collaboration Protocol Profile) and CPA (Collaboration Protocol Agreement) documents, business process descriptions, service interface descriptions, program code for accessing companies' business interfaces, communication channel description etc. [7], [8].

III. Choosing Appropriate Technology for Implementing RIM

A. Object Oriented Database (OODB)

The UML model can be easily mapped into the ODMG (Object Data Management Group) data model because of its object oriented nature. A number of successful projects utilized OODBs, but OODBs have weaknesses that must be considered as well.

In many OODB implementations, the OODB is tightly coupled to the application and makes database layer no longer independent and replaceable.

OODBs are typically weaker for ad hoc queries against the database. Ad hoc query support seems to be an area where OODBs find it difficult to compete with relational database in both performance and features. OODB query optimization and functionality often legs behind major relational database products.

B. Native XML database

Native XML database has an XML document as its fundamental unit of logical storage, just as a relational database has a row in a table as its fundamental unit of logical storage [9]. Since XML documents are used to facilitate an ebXML-based B2B partnerships and transactions, there are good reasons to use native XML databases for storing these documents. Although XML native database technology is still not profound and powerful as relational database technology, there are solutions that uses native XML database to implement ebXML Registry [10]. We will give our comment on this technology and mention some of its advantages and disadvantages.

Native XML database is contextually aware of the structure of XML documents. This allows querying and updating stored documents using XML technologies (e.g. XUpdate, XPath). Using XUpdate we are able to update elements of a XML document without having to overwrite the whole document. When comparing retrieval speed to a relational database, native XML database is faster when retrieving fewer, larger fragments of XML documents. XPath query string has the same function in XML database as SQL query in RDBMS.

The nature of XML content should also be analyzed when deciding whether to use native XML database. XML content can be document-centric or data-centric. Payload and structure of a document-centric XML content are equally important. For data-centric XML content only payload is important and XML is used for transporting purpose. Document-centric XML content is better suited to native XML database.

According to ebXML specification [7], ebXML implementation must support query capabilities. Each query alternative is directed against a single class defined in the ebXML Registry Information Model. After reference to specific document has been found by searching through metadata, original document can be retrieved. Since metadata represents data-centric content, implementation in relational database would provide easier development and better search capabilities.

Native XML database can play significant role when it comes to searching through the content in XML documents, which is not stored as metadata. The best combination would be storing original XML documents in XML native database and all other data and metadata in relational database.

One of the native XML database implementation is Oracle Native XML database, which was used in our implementation of Registry. All Extrinsic_Object [6] are inspected to determinate whether are of XML type or not. That is achieved by trying to parse Extrinsic_Object into XML DOM Model. If no error is occurred during of parsing process it is assumed that Extrinsic_Object is well formed XML document and it is stored in Oracle XML Native database. If Extrinsic_Object is not of XML type it is stored as Oracle BLOB type. In Fig. 2 is presented design of database table used for storing Extrinsic_Object.

Column ID is used for storing UUID of Extrinsic_Object. Column CONTENT_TYPE describes whether content is of XMLTYPE or not. If

CONTENT_TYPE refers to XMLTYPE, Extrinsic_Object will be stored in CONTENT_XML column. Documents and data that are not of XMLTYPE are stored in CONTENT_BLOB column. Column XML_SIGNITURE is used for storing digital signature of Extrinsic_Object.

Column Name	Data type	Size
ID	VARCHAR2	64
CONTENT_TYPE	VARCHAR2	32
CONTENT_XML	XMLTYPE	
CONTENT_BLOB	BLOB	
XML_SIGNITURE	XMLTYPE	

Figure 2. Design of database table used for storing Extrinsic_Object

There are many advantages of this kind of storing Extrinsic_Object. XPath query, XUpdate and other XML technologies can be done against XML documents which increase query capabilities and retrieval speed.

C. *Relational Database Management System (RDBMS)*

The subset of UML describes the information model of the UML meta model mostly in terms of entities that participates in relationships and have attributes. This representation lends itself towards being mapped into a relational schema for a relational database.

In our implementation RDBMS is used for storing all data that are not of XML type.

IV. Implementation of Security in ebXML Repository

In ebXML RIM specification, UML diagrams are used as a way to concisely describe concepts. However, they are not intended to convey any specific implementation or methodology requirements. Here is our proposal of mapping security part of ebXML RIM specification to data model.

This section concerns only about the access security to each object in ebXML Registry. It implicates that authorization and authentication are already successfully finished and identity of the ebXML Registry user is known. From now on we will use the term principal for authorized and authenticated user to conform specification nomenclature. Principal is a generic term used by the security community to include both people and software systems.

Fig. 3 shows UML Class diagram of the objects in the Registry from the security perspective.

Every RegistryObject is associated with exactly one AccessControlPolicy. AccessControlPolicy is a collection of Permissions, which defines the policy rules that govern access to operations or methods performed on that RegistryObject. Requesting Principal has access to a method in a RegistryObject if it has any of the Privileges defined in the Permission. A Privilege object contains zero or more PrivilegeAttributes. A PrivilegeAttribute can be a Group, a Role, or an Identity and can be granted to a Principal object. The Principal object can have a set of PrivilegeAttributes, but at least one must be identity, and other can be role memberships or group memberships.

This model of security enables the flexibility to have object access control policies that are based on any combination of Roles, Identities or Groups.

There are several solutions for mapping this kind of UML diagrams, especially PrivilegeAttribute part. One of the possible solutions leads to three tables for handling many-to-many relationships between Privilege and PrivilegeAttribute, mapped into RDBMS as PRIVILEGE, ROLE, GROUP and IDENTITY tables. Tree more tables also have to be made in order to handle many-to-many relationships between Principle and Role, Group and Identity. These six tables used only for handling many-to-many relationships significantly increase total number of tables and lead us toward finding more appropriate solution.

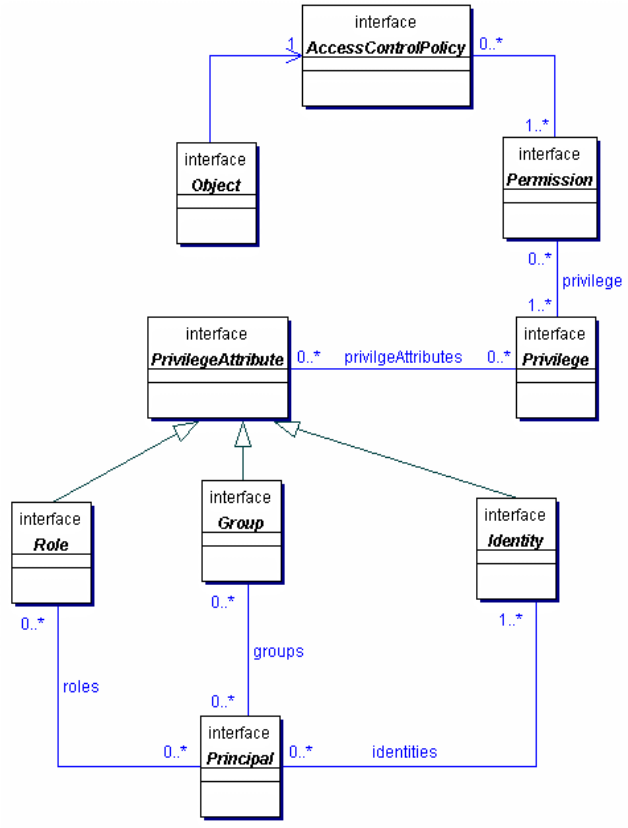

Figure 3. UML class diagram: Security view

We offer less complex but still efficient solution. Entity-Relationship diagram is shown in Fig. 4.

In our database relational schema Role, Group and Identity classes are not mapped into separate tables. We have created PRIVILEGE_ATTRIBUTE_TYPE table to store the instances of groups, roles and identities. In this

ConTEL 2003, ISBN: 953-184-052-0

way the final number of tables in database relational schema is reduced from eight to tree.

This solution is also open to further extensions of ebXML Registry security. When a new subclass of PrivilegeAttribute is included, there is no need for adding a new table; only an additional entry is inserted in the PRIVILEGE_ATTRIBUTE_TYPE table.

Meaning and purpose of rest of the tables are the same as described in specification [6].

V. Registry Service Implementation Concept

The ebXML Registry specification defines its two parts, the ebXML Registry Service and ebXML Registry Client. The ebXML Registry Client is used for access to registry. The ebXML Registry Service provides the interfaces and methods for managing the ebXML Repository. Fig. 5 presents high view on ebXML Registry architecture [7].

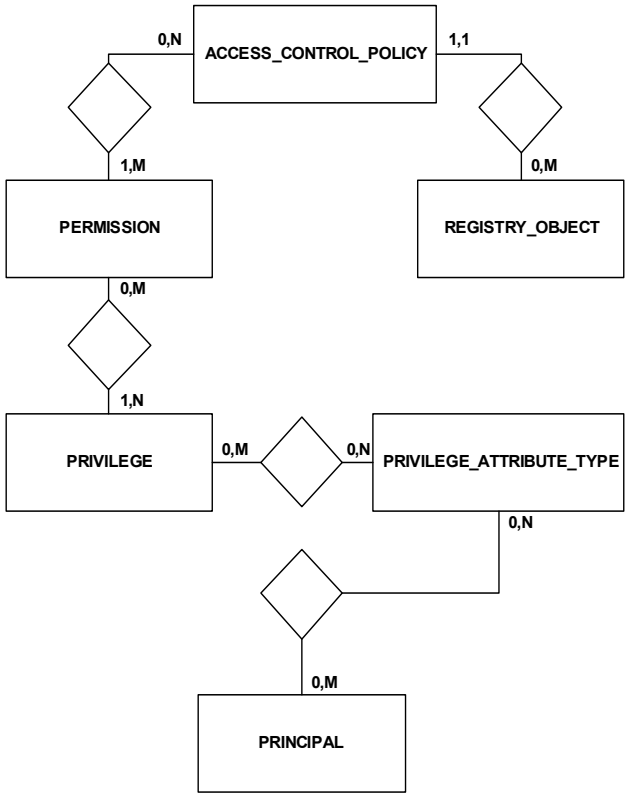

Figure 4. E-R diagram: Security view

We will now detail on the implementation concept of the ebXML Registry Service.

The ebXML Registry Service consists of two interfaces:

- The Life Cycle Management (LM) interface provides a set of methods for managing the contents of the registry.
- The Query Management (QM) interface controls the discovery and retrieval of information from the registry.

The ebXML compliant implementation of the Registry must implement these interfaces along with all the methods they specify. The implemented ebXML Registry Service interfaces must be bound to a communication protocol. The ebXML Registry Service specification describes two concrete bindings:

- A SOAP binding using the HTTP protocol
- An ebXML Messaging Service (ebMS) binding

An implementation of ebXML Registry Service may implement one or both of these bindings (as it is shown in Fig. 6).

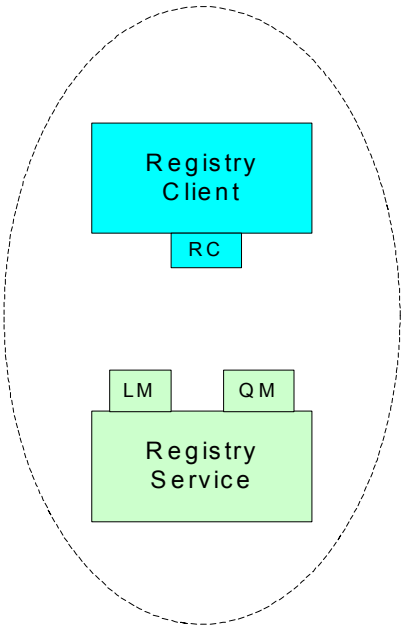

Figure 5. Registry Service Architecture

In implementation of the ebXML Registry Service we use SOAP binding because the SOAP is widely accepted and it has became a standard communication protocol for communication between web services. It assures possibility of communication for an ebXML Registry with other types of registries (for example UDDI) [11].

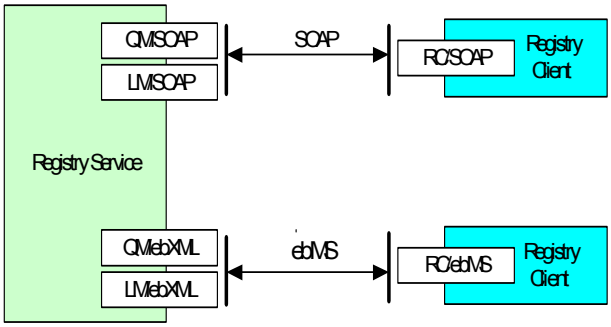

Figure 6. ebXML Registry Service protocol binding

The Registry Service, that implements the life cycle management functionality of the Registry, exposes this interface to the Registry Clients. It can be viewed as sub-

service of the Registry Service and the Registry Client invokes its methods. It provides the functionality required by the Registry Clients to manage the life cycle of repository items (e.g. XML documents required for ebXML business processes). The Registry Clients use this interface to submit objects, to classify, to associate, to deprecate and to remove objects within the repository. Full semantic meaning of these terms can be found in [7]. All methods of the Life Cycle Management Service are listed in the Table I.

The main purpose of the Life Cycle Management Service is to manage the life cycle of repository items. The UML statechart diagram in Fig. 7 shows typical life cycle of a repository item [8].

TABLE I

LIFE CYCLE MANAGEMENT INTERFACE METHODS

Method Summary of Life Cycle Management Interface	
RegistryResponse	approveObjects(ApproveObjectsRequest req) Approves one or more previously submitted objects.
RegistryResponse	deprecateObjects(DeprecateObjectsRequest req) Removes one or more previously submitted objects from the registry.
RegistryResponse	removeObjects(RemoveObjectsRequest req) Removes one or more previously submitted objects from the Registry.
RegistryResponse	submitObjects(SubmitObjectsRequest req) Submits one or more objects and possibly related metadata such as Associations and Classifications.
RegistryResponse	updateObjects(UpdateObjectsRequest req) Updates one or more previously submitted objects.
RegistryResponse	addSlots(AddSlotsRequest req) Add slots to one or more registry entries.
RegistryResponse	removeSlots(RemoveSlotsRequest req) Remove specified slots from one ore more registry entries.

The Registry Service exposes the Query Management interface to the Registry Clients. It can be viewed as the second sub-service of the Registry Service. The Registry Clients may use this interface to perform browse and drill down queries or ad hoc queries on registry content. The quality of the response greatly depends on metadata stored in repository such as the associations and the classifications. This interface has only one method and it is shown in Table II.

Filter Query must be supported by every ebXML Registry Service compliant implementation. The implementation of the other type of query capabilities, the SQL Query, is optionally.

TABLE II

QUERY MANAGEMENT INTERFACE METHODS

Method summary of Query Management Interface	
RegistryResponse	submitAdhocQuery(AdhocQueryrequest req) Submit an ad hoc query request

The architecture of our implementation of the ebXML Registry/Repository is shown in Fig. 8. The implementation of the ebXML Registry Services layer relies on open-source project hosted at [12].

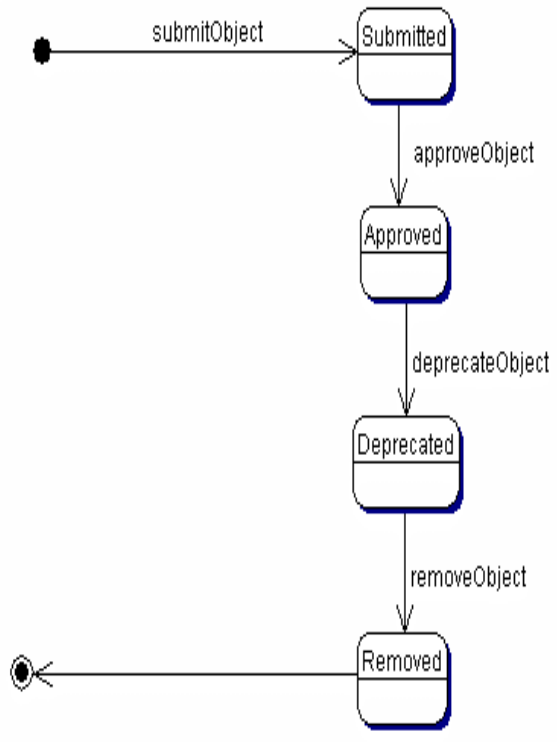

Figure 7. UML diagram of the life cycle of a repository item

There are few reasons for this. This open-source project is sponsored and leaded by the founders of the ebXML specification. That means that we have very good idea in which direction this specification will progress and also we have been avoiding, in many parts of the implementation, reinventing the wheel.

In our implementation we have made many improvements over the existing open-source implementation.

- We have made that ebXML Registry Services server can be distributed which was not the case with open-source implementation. This is very important knowing that processing of the business documents is highly resource demanding. Now we can have system of Registry Services sharing same registry items and metadata about them.
- Among several database technologies which have possibility of storing the XML documents in the native XML database we have chosen Oracle 9i database. The underlying database schema model is described in chapter III. All other types of repository items (UML diagrams, software, pictures etc.) are stored in the database as binary objects.

Implementation of the Access Control Policy control system, which implements RIM security model, will be finished in future. The Access Control Policy control system is separated of the rest of the implementation in which are implemented only authorization and authentication of the ebXML Registry users.

Knowing that primary interest of the project sponsor is fast adoption of the ebXML as a standard for e-business in Croatia, we have decided to do our implementation in two steps. The first step which is described in this paper relies on open-source project [12].

Our future work will be aimed on deploying the whole implementation of the ebXML Registry Services server on the J2EE platform specification. The second step will bring mostly performance improvements over the implementation in the first step and will not affect the way users use the ebXML Registry Services. The second step demands more time and careful modeling of the Life Cycle Management and Query Management engine. This is the reason why we have decided to divide implementation in two steps. Doing this the end-users will have more time to validate and to accept the ebXML as a standard for e-business.

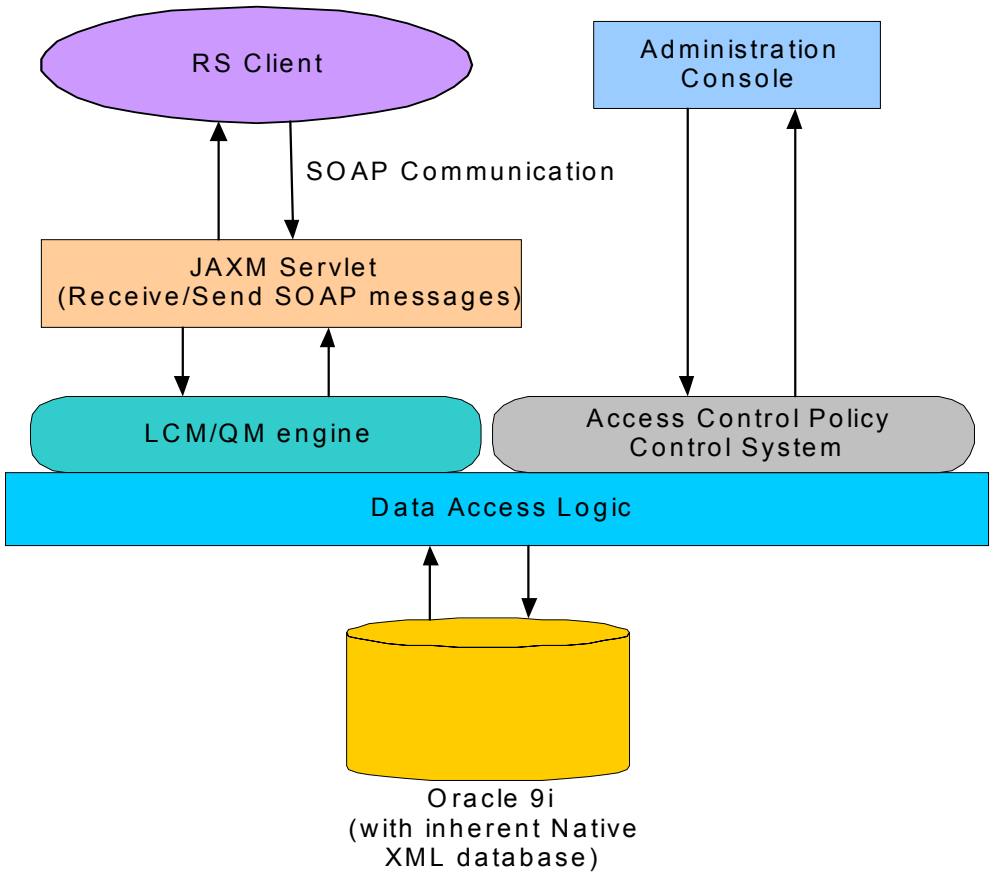

Figure 8. The architecture of the implementation of the ebXML Registry/Repository

VI. CONCLUSION

This paper presents part of an ongoing research project, aimed at setting up e-business support services in a country with SME-prevailing environment, such as Croatia. Implementation of ebXML Registry/Repository is an important part of the project, whereas this infrastructure acts as key component for the project. We made a short introduction to ebXML Registry/Repository, which acts as a heart of ebXML. During the process of implementation we had to deal with the specifications, given in the form of UML diagrams. We have mapped ebXML RIM specification in relational schema, which was considered to be the best choice for our implementation. We have used RDBMS technology to store all data which are not of XML type. XML documents are stored in Native XML database to enable technologies like XPath query, XUpdate Using these technologies can increase query capabilities and retrieval speed when dealing with XML documents.

We put special emphasis on the part of information model that is related to the security features of the ebXML Registry, because our proposal of solution remarkably simplifies development of application level and provides future extendibility of that part of ebXML Registry

We have also described the implementation of the ebXML Registry Services server. Through its interfaces users submit their business documents and make searches for business documents of the other users of the ebXML Registry/Repository.

We consider architecture of the ebXML Registry/Repository we are creating to be our contribution to better performance of Registry Service – Registry Service server distribution, Native XML database storage etc.

REFERENCES

[1] www.cecid.hku.hk

[2] I. Matasic, Z. Skocir: EbXML as developing Country e-business strategy proposal, Proceedings of 10th International Conference on Software, Telecommunications and Computer Networks SoftCOM 2002, Split, Dubrovnik, Croatia, Venice, Ancona, Italy, October 8-11.2002., p.p. 166-194.

[3] www.ietf.org.

[4] www.w3.org.

[5] UN/CEFACT-OASIS: EbXML Technical Architecture Specification v1.0.4, http://www.ebxml.org/specs/ebTA.pdf, February 2001.

[6] UN/CEFACT-OASIS: EbXML Registry Information Model, v2.0, http://www.ebxml.org/specs/ebrim2.pdf, December 2001.

[7] UN/CEFACT-OASIS: OASIS/ebXML Registry Services Specification, v2.0, http://www.ebxml.org/specs/ebRS2.pdf, December 2001.

[8] M. Vrhoci, D. Delmis, J. Maksimovic: Collaboration Protocol Profile (CPP) and Collaboration-Protocol Agreement (CPA) in B2B transactions, Proceedings of the MIPRO2002 (Electronic Commerce), Opatija , 2002, p.p. 45-50.

[9] www.xmldb.org.

[10] www.xmlglobal.com.

[11] I. Magdalenic, I. Pejakovic, D. Pranic: Business documents presentation and interchange using SOAP, Proceedings of the MIPRO2002 (Electronic Commerce), Opatija, 2002, p.p. 13-17.

[12] http://ebxmlrr.sourceforge.net