Metadata Management in Databases

Uzma Mahmood
College of Computer Science
Karachi Institute of Economics and Technology, Pakistan
uzma.mahmood@aku.edu

Dr. Irfan Hyder
College of Computer Science
Karachi Institute of Economics and Technology, Pakistan
hyder@pafkiet.edu.pk

Abstract

The database has emerged as a major tool across all enterprises. The databases are becoming increasing complex, both in their internal structure and in their interactions with applications. When we are making a database schema change, it is important to identify all the dependencies in the program code that might be affected by the change. Dependency on the underlying database structure is typically hardcoded. During upgrades when the database structure changes, the application needs to be recoded and recompiled i.e. dependency on the hardcoded database structure makes the upgradation of the application tedious and error prone process. The Metadata framework proposed in the thesis provides enough information to make the application modify itself to changes in the database at runtime.

1. Introduction

Typically software development is driven by the design of a database. Many of the tools especially Oracle Development Environment and Microsoft Integrated Development Environment's (IDE's) have their typical development methodologies and IDE's driven from the database structure. When we are making a database schema change, it is important to identify all the dependencies in the program code that might be affected by the change. Without doing a complete impact assessment, we run the risk of causing problems when implementing the schema change.

The Metadata framework proposed in the thesis provides enough information to make the application modify itself to changes in the database at runtime. At runtime, Metadata helps us in finding out what is available and then with the help of that information (called metadata); we can build proper database queries.

Metadata, which can be broadly defined as "data about data", refers to the searchable definitions used to locate information. On the other hand, database metadata, which can be broadly defined as "data about database data", refers to the searchable definitions used to locate database metadata. Imagine, at runtime, trying to execute a SQL query in a relational database without knowing the name of tables, columns or views. Metadata helps us in finding out what is available in the database and then with the help of that information, we can build proper SQL queries at runtime. In a nutshell, database metadata enables dynamic database access [17].

When we want to make a change in database, instead of directly making changes, we will make the changes to the metadata. The framework will then automatically (i) design the queries (ii) make necessary changes to database. Figure 1 shows the Typical IDE framework in which the development environments directly fetches values from the database and bind those values to forms, reports and Data Access layer. Figure 2 shows the proposed metadata framework in which the Application frameworks retrieves values from the database with the help of Application metadata and bind those values to forms, reports and Data Access layer.

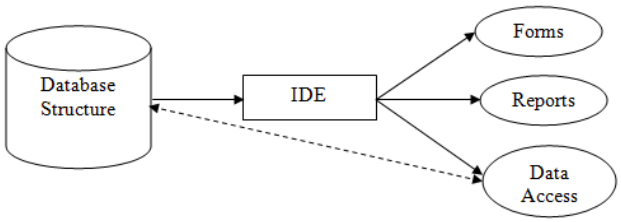

Figure 1: Typical IDE Scenario

The aim is to populate the combos on the basis of metadata at runtime. SQL parameterized queries are used for data integration. For instance, a simple join query between two relations has the generic structure:

Select <attr-list>

From <rel1> r1, <rel2>r2

Where r1.<attr1>=r2.<attr2>

To construct the above query at runtime and populate combos, an algorithm is used which fetches the values from database based on<AttrID>. For this purpose, I worked on ERP based software '**Business Pattern**' which is a complete information system. It takes organization to levels of higher efficiency and effectiveness.

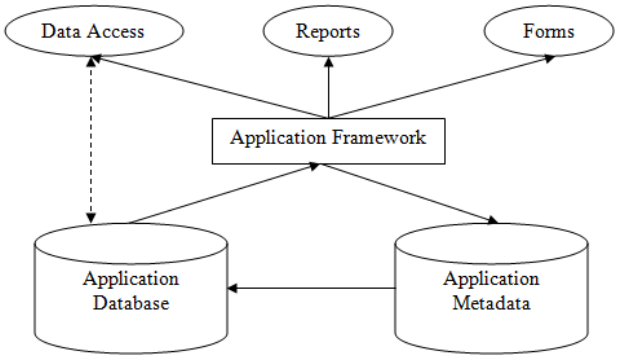

Figure 2: Proposed Metadata Framework

The rest of the paper is organized as follows: Section 3 deals with related work, Section 3 describes the issues and problems related to database dependency on its underlying architecture, Section 4 gives a brief introduction to Business Pattern Software. Section 5 focuses on the methodology used for the successful adoption of removing database dependency on its underlying architecture, Section 6 focuses on open Issues and future work and concludes the research paper.

2. Related Work

A lot of work has been proposed in removing database dependencies on its underlying structure. Our main focus is based on removing database dependencies using Metadata. The research literature focuses on eliminating Database dependencies and Metadata management.

2.1 Normalization

One of the purposes of normalization is to precisely enhance logical database independencies, such that changes in the database structure will necessities the minimum possible application maintenance effort. But it is difficult to work with a completely normalized database. If we need to make a change in one table, the entire database application breaks.

2.2 Dependency Injection

Dependency Injection is about the principle of separating configuration from use. Instead of embedding hard coded values into program code or even having that piece of program code gather the values it needs from an external location, it completely remove the burden of finding out about these dependencies from the code. Instead, it ensures that when the time comes to run the code, all the information it needs to have is available.

Dependency Injection makes it easier to reuse the code in various environments provided that the injector does its job in a timely fashion [20].

2.3 Ontology Management

The use of ontology could both speed up the design process and make the results more accurate. In [4], SQL parameterized queries are used for data integration. For instance, a simple join query between two relations has the generic structure

Select <attr-list>
From <rel1> r1, <rel2>r2
Where r1.<attr1>=r2.<attr2>

Expressions appearing inside angle brackets are formal parameters. Consider the following:

<rel1> = Item <attr1> = ItemID
<rel2> = ItemAtWH <attr2> = ItemID
<attr-list> = (ItemID,ItemName)

The join perform is then

Select ItemID,ItemName
From Item r1, ItemAtWH r2
Where r1.ItemID=r2.ItemID

This approach is very useful for our research but only requires query optimization.

2.4 EbXML

As discussed by OASIS, "*ebXML is a set of specifications, and these specifications construct the implementation framework of e-business. The vision of ebXML is implementing a global e-business market place where enterprises of any size and in any geographical location can meet and conduct business with each other through the exchange of XML based message*" [27].

A registry/repository is a common means to create, deploy, discover and retrieve web services over the internet. The registry provides information (metadata) about the registry objects, while the repository is where the authentic registry object resides.

The ebXML registry provides a set of services that enable sharing of information between interested parties for the purpose of business process integration. Vendor can register their service in the ebXML registry and clients can retrieve their required service from it [28]. When a vendor registers an artifact into a repository (through registry object), it should also provide some information to describe the artifact. This kind of information is called semantic or metadata of the repository [28].

3. Issues and Problems

Issues and problems can be categorized as follows:

3.1 Removing Database Dependencies

When we are making a database schema change, it is important to identify all the dependencies in the program code that might be affected by the change. Without doing a complete impact assessment, we run the risk of causing problems when implementing the schema. Therefore, having some easy, and automated method helps identify the objects that will be impacted by our schema change is critical and a time saver. Following are some of the approaches to deal with the problems of schema change:

3.1.1 Normalization

- A completely normalized database is far too complex to work with;
- Change in one application can break the entire database application;
- Database application change so frequently that normalization can't break it.

3.1.2 Dependency Injection

If we apply the concept of Dependency Injection to SQL, it would mean no more hard coded configuration details in SQL code, no more dependencies of code on potentially changing conditions provided that the injector does its job in a timely fashion [20].

3.1.3 Using SYSCOMMENT Table

The actual code for views, rules, defaults, triggers and stored procedures are stored in syscomments table. . By scanning syscomments table, we narrow our focus of impact analysis down to those objects only that might be affected by the change [19]. Following are the issues related to using syscomments table:

- Only scans objects that are stored in SQL Server (views, rules, defaults, triggers);
- If the application uses T-SQL SELECT, UPDATE, INSERT and DELETE statements in code blocks that are not stored in SQL Server, then we need to use other methods for scanning the code.

3.1.4 Using sp_depends Stored Procedure

SQL Server maintains a database system table named sysdepends. SQL Server uses this table to store object dependencies. This table only contains information about objects that are dependent upon other objects. To access the sysdepends table information SQL Server provides the sp_depends system stored procedure [19]. Following are the issues related to using sp_depends stored procedure:

- Only scans objects that are stored in SQL Server (views, rules, defaults, triggers);
- If the application uses T-SQL SELECT, UPDATE, INSERT and DELETE statements in code blocks that are not stored in SQL Server, then we need to use other methods for scanning the code.

4. Business Pattern Software

Business Patterns is a complete information system for organization. It takes organization to levels of higher efficiency and effectiveness. Implementing Business Patterns in organization means more control of business by making good use of state-of-the-art-technology. Business Patterns transforms organization to work in a new way to meet the challenges of twenty first century.

Business Pattern salient features include:

- User friendly Windows based environment
- Strong command level security
- Scalable according to requirements
- Customizable/Flexible
- Central storage of database

Business Pattern advantages include:

- Better controls
- Accurate and Reliable information
- Effective decision making
- Time saving
- Adds profitability

Business Pattern Software includes following modules:

- Accounts
- Financial Budgeting and Forecasting
- Purchases
- Sales
- Inventory
- Production
- Management

5. Approaches and Methodologies

The importance of managing information in modern life and especially in midsized and large enterprises should not be underestimated. Usually various heterogeneous large databases are filled with valuable information about products, customers, processes, histories of all these, and much more. The need to process, manage and especially find that information is already addressed in the previous researches. There is a consequent need for understanding, maintaining, querying, integrating and evolving databases. In successfully performing these tasks, metadata plays an important role. The Metadata framework proposed in the thesis provides enough information to make the application modify itself to changes in the database at runtime. At runtime, Metadata helps us in finding out what is available in the database and then with the help of that information (called metadata); we can build proper SQL queries.

For this purpose, I worked on ERP based software '**Business Pattern**' which is a complete information system. It takes organization to levels of higher efficiency and effectiveness.

Figure 3 shows the Login screen of Business Pattern software.

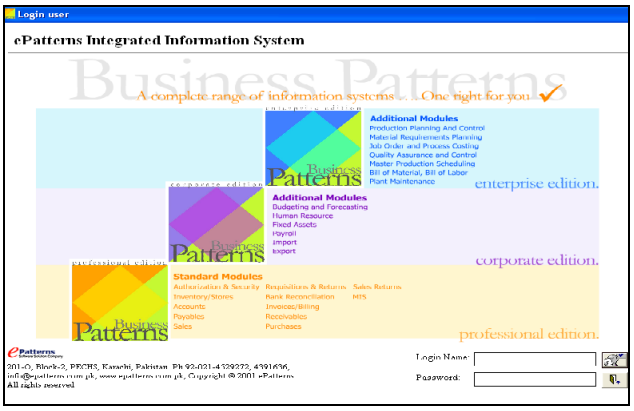

Figure 3: Login User

We have limited our scope of removing underlying database dependencies till combos i.e. dropdown list. Our aim is to populate the combos on the basis of metadata at runtime. The algorithm used to remove the application's underlying dependency on code is shown in figure 4. This algorithm consists of following six steps:

Step 1

The first step is to get the <Attr ID> of the combo. We get this value from the code at runtime when the value from the combo is selected.

Step 2

Our next step is to get the Text Field, Name Field of the combo and the Table Name from which we have to populate our combo i.e. we have to identify the <IDField>, <NameField> and <TableName> against the <Attr ID>. <Attr> table is the main table where

our metadata is stored. Considering table 1, our SQL statement will be:

Select <IDField>, <NameField>, <TableName>

From <Attr>

Where <AttrID>=<@AttrID>

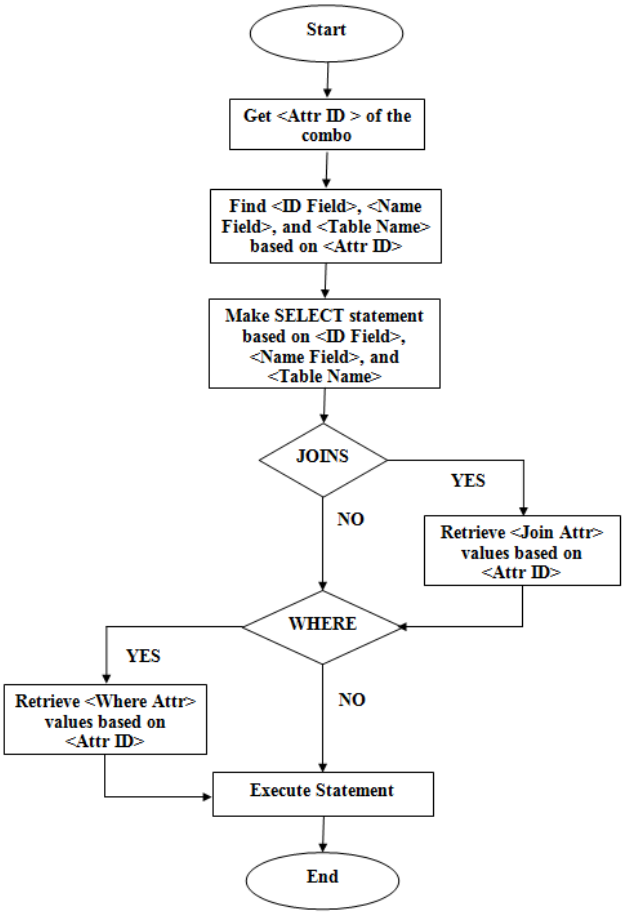

Figure 4: Algorithm

In this way we will get the Table Name from which we have to fetch the ID Field and Name Field to bind it with combo.

Table 1: <Attr> Table Structure

Attr-ID	ID-Field	Name-Field	Table-Name
1	TestID	TestName	TestTable

Step 3

Once we have identified the <IDField>, <Name-Field> and <TableName>, our next step is to make a SELECT statement and bind it to the combo. Considering table 2, our SQL statement will be:

Select <TestID>, <TestName>

From < TestTable>

Table 2: <TestTable> Table Structure

TestID	TestName
1	Testing Data

If our query has join statements then we have to follow step 4, else we can switch to step 5.

Step 4

<TestTable2> table as shown is table 3 is used to demonstrate a join query example. <TestID> is used as a foreign key.

Table 3: <TestTable2> Table Structure

TestID2	TestName2	TestID
1	Testing Data	1
2	Sample Data	1

Table 4 is a <JoinAttr> table which contains <Attr-ID>, <JoinTable> and <JoinStatement>.

Table 4: <JoinAttr> Table Structure

Attr-ID	JoinTable	JoinStatement	Join-Type
1	TestTa-ble2	TestTa-ble.TestID= Test-Table2.TestID	InnerJoin

In order to find the join statement we have to retrieve the <JoinTable>, <JoinStatement> and <JoinType> values against the <AttrID>. Consider the following SQL Query:

Select <JoinTable>, <JoinStatement>, <JoinType>

From < JoinAttr>

Where <AttrID>=<@AttrID>

Once we have the <JoinTable>, <JoinStatement> and <JoinType> our original SQL query would be:

Select <TestID>, <TestName>

From < TestTable>

<JoinType> <JoinTable> on <JoinStatement>

If our query has a where clause then we have to follow step 5, else we can switch to step 6.

Step 5

<WhereAttr> table as shown is table 5 is used to demonstrate a where clause example.

Table 5: <WhereAttr> Table Structure

AttrID	WhereClause
1	**TestID**
1	**TestID2**

In order to find the where clause values, we have to retrieve the <WhereCluase> values against the <AttrID>. Consider the following SQL Query:

Select <AttrID>, <WhereCluase>

From <WhereAttr>

Where <AttrID>=<@AttrID>

Once we have the <WhereClause> our original SQL query would be:

Select <TestID>, <TestName>

From < TestTable>

<JoinType> <JoinTable> on <JoinStatement>

Where <TestID>=1 and <TestID2>=2

Step 6

Once the SQL query is completely formed, our last step is to execute that query and bind the results to the combo.

combo.DataTextField =”<TestID>”

combo.DataValueField =”<TestName>”

6. Conclusion and Future Work

In the past, metadata was often neglected. But, once computers were in common use for storing data, the need for techniques to retrieve the data became important. Since then the concept of metadata in computer science has evolved, starting from the simple file systems (file names and types) in the early 60s, then to database management systems (to describe database fields) in the early 70s.

Dependency on the underlying database structure of an application is typically hardcoded in the application code. During upgrades when the database structure changes, the application needs to be recoded and recompiled i.e. dependency on the hardcoded database structure makes the upgradation of the application tedious and error prone process. The Metadata framework proposed in the thesis provides enough information to make the application modify its interactions to changes in the database at runtime. At runtime, Metadata helps us in finding out what is available in the database and then with the help of that information (called metadata); we can build proper SQL queries i.e. when we want to make a change in database, instead of directly making the changes, we will make the changes in the metadata.

My future work includes implementing this algorithm on more complex database queries. Moreover, future work also includes implementing this algorithm using protégé 3.4 and protégé 4.0. However, protégé 4 does not implement a database backend. D2R mapping can also be used which exports data from database into RDF format and then querying the RDF to retrieve values.

Acknowledgments

I would like to thank my supervisor Dr. Irfan Hyder for his support and useful comments on this work.

7. References

[1] Vijayan, S., Veda,S.,The Role of Domain Ontologies in Database Design: An Ontology Management and Conceptual Modeling Environment. *In Proceedings of ACM Transactions on Database Systems (TODS), 2006*

[2] Heiner, S., Frank, H., Ontology Based Metadata Generation from Semi Structured Information**.** *In Proceedings of the 1st international conference on Knowledge capture, 2001*

[3] Sunita, S., Archana, S., Generating OWL Ontologies from a Relational Databases for the Semantic Web. *In Proceedings of the International Conference on Advances in Computing, Communication and Control, 2009*

[4] Shun, C., James, L., Catharine, W., Metadata Management and Relational Databases. *In Proceedings of the 43rd annual Southeast regional conference, 2005*

[5] Lipyeow, L., Haixun, W., Min, W., Unifying Data and Domain Knowledge Using Virtual Views. *In Proceedings of the 33rd international conference on Very large data bases, 2007*

[6] Joseph, P., Bruce, B., Ontology Guided Knowledge Discovery in Databases. *In Proceedings of the 1st international conference on Knowledge capture ,2001*

[7] www.sas.com/resources/asset/sas-metadata-server-factsheet.pdf

[8] openmeta.googlecode.com/files/OpenMeta.pdf

[9] www.sun.com/storage/white-papers/Metadata_Management.pdf

[10] Rada, C., Fereidoon, S., Query Optimization using Restructured Views. *In Proceedings of the 15th ACM international conference on Information and knowledge management, 2006*

[11] Divesh, S., Yannis, V., Intensional associations between data and metadata. *In Proceedings of the 2007 ACM SIGMOD international conference on Management of data, 2007*

[12] Gregory, K., Sofia, A., Vassilis, C., Dimtris, P., Michel, S., RQL: a declarative query language for RDF. *In Proceedings of the 11th international conference on World Wide Web, 2002*

[13] Xiaonan, L., Brewster, K., James, W., Lee, G., A metadata generation system for scanned scientific volumes. *In Proceedings of the 8th ACM/IEEE-CS joint conference on Digital libraries, 2008*

[14] John, R., Paul, J., Osprey: peer-to-peer enabled content distribution. *In Proceedings of the 5th ACM/IEEE-CS Joint Conference on Digital Libraries, 2005. JCDL '05.*

[15] Jane, H., Carl, L., Combining RDF and XML schemas to enhance interoperability between metadata application profiles. *In Proceedings of the 10th international conference on World Wide Web, 2001*

[16] David, N., Chu-Hsiang, C., David, B., Dana, B., Micheal, T., A lightweight metadata quality tool. *In Proceedings of the 8th ACM/IEEE-CS joint conference on Digital libraries. 2008*

[17] Divesh, S., Yannis, V., Internal Associations between Data and Metadata. *In Proceedings of the 2007 ACM SIGMOD international conference on Management of data, 2007*

[18] http://www.dbdebunk.com/

[19] http://www.databasejournal.com/

[20] http://technology.amis.nl/blog/

[21] Breitman, K.K., do Prado Leite, J.C.S., Ontology as a Requirement Engineering Product. *11th IEEE International Conference on Requirements Engineering,* 2003.

[22] Caralt, J.C., Kim, J.W., Ontology Driven Requirements Query. *40th Annual Hawaii International Conference on* System Sciences, 2007. HICSS 2007.

[23] http://www.govtalk.gov.uk/documents/Laymans_guide_to_metadata%20v1.1.pdf

[24] www.sun.com/storage/whitepapers/**Metadata_Management**.pdf

[25] Haase, K., Context for Semantic Metadat. *In* Proceedings of the 12th annual ACM international conference on Multimedia, 2004

[26] http://www.code.org/resources/tutorials/ProtegeOWLTutorial-p4.0.pdf

[27] Liang, P., He, K., Li, B., Liu, J**.,** Interoperability test of ebXML e-business solutions**.** *4th International Conference on Computer and Information Technology, 2004*

[28] Song, D., Liu, W., He, Y., He, K**.,** Ontology application in software component registry to achieve semantic interoperability**.** International Conference on *Information Technology: Coding and Computing, 2005. ITCC 2005*

[29] Wang, C., He, K., He,Y., *MFI4Onto*: *Towards Ontology Registration* on the Semantic W**eb**. *6th IEEE International Conference on Computer and Information Technology, 2006.*

[30] Lee, K, L., Lee, T, K., Lee, K, C**.,** A method to integrate business registries by using OWL-S ontologies. *International Conference on Next Generation Web Services Practices, 2005*

[31] Boughaci, D., Drias, H., An agent-based approach using the ebXML specifications for e-business. *InternationaConference on Information Technology: Coding and Computing, 2005.*

[32] Liu, W., He, L., He, K., A semantic interoperability extension model to the ebXML registry. *International Conference on Information Technology: Coding and Computing, 2005.*

[33] Song, D., Liu, W., He, Y., He, K., Ontology application in software component registry to achieve semantic interoperability. International Conference on *Information Technology: Coding and Computing, 2005. ITCC 2005.*

[34] Driouche, R., Boufaida, Z., Kordon, F., Towards Integrating Collaborative Business Process Based on a Process Ontology and EbXML Collaboration Scenario. *17th International Conference on Database and Expert Systems Applications, 2006. DEXA '06***.**

[35] Liang, P., He, K., Li, B., Liu, J., Interoperability test of ebXML e-business solutions. *4th International Conference on Computer and Information Technology, 2004.*

[36] Ahn, K., Kim, H., Chung, J., An efficient structure for an object-oriented database. *2nd International Conference on Embedded Software and Systems, 2005.*

[37] Tjoa, A, M., Andjomshoaa, A., Shayeganfar, F., Wagner, R., Semantic Web challenges and new requirements. *16th International Workshop on Database and Expert Systems Applications, 2005***.**

[38] Wen, Q., EbXML-based Application Integration in Service oriented Business. *IEEE Asia-Pacific Conference on Services Computing, 2006. APSCC '06.*

[39] Hofreiter, B.; Huemer, C.; Klas, W. , EbXML: Status, Research, Issues, and Obstacles. *12th International Workshop on Research Issues in Data Engineering: Engineering E-Commerce/E-Business Systems, 2002. RIDE-2EC 2002.*

[40] Huang, Ning., Diao, S., Structure-Based Ontology Evaluation. *IEEE International Conference on e-Business Engineering,* 2006. ICEBE; 06

[41] Mukhopadhyay, D.; Banik, A.; Mukherjee, S. A Technique for Automatic Construction from Existing Database

to Facilitate Semantic Web. *10th International Conference on Information Technology, (ICIT 2007).*

[42] Lee, S.W., Gandhi, R.A., Ontology-based Active Requirements Engineering Framework. *12th Asia-Pacific on Software Engineering Conference,* 2005.

[43] Kaiya, H., Saeki, M., Ontology Based Requirements Analysis: Lightweight Semantic Processing Approach. *Fifth International Conference on Quality Software,* 2005. (QSIC 2005)

[44] Kaiya, H., Saeki, M., Using Domain Ontology as Domain Knowledge for Requirements Elicitation. *14th IEEE International Conference on Requirements Engineering***,** 2006.

[45] Zong-yong, L., Zhi-xue, W., Ying-ying, Y., Yue, W., Ying, LIU., Towards a Multiple Ontology Framework for Requirements Elicitation and Reuse. *31st Annual International Computer Software and Applications Conference,* 2007. COMPSAC 2007 - Vol. 1.

[46] Xuefeng, Z., Zhi, J., Detecting of Requirements Inconsistency: An Ontology Based Approach. *5th International Conference on Computer and Information Technology,* 2005. CIT 2005.

[47] Xuefeng, Z., Zhi, J., Inconsistency Measurement of Software Requirements Specifications: An Ontology Based Approach. *10th IEEE International Conference on Engineering of Complex Computer Systems, 2005.* ICECCS 2005.

[48] Xiang, J., Liu, L., Qiao, W., Yang, J., A Service Requirement Elicitation Mechanism Based On Ontology. *31st Annual International Computer Software and Applications Conference*, 2007. COMPSAC 2007

[49] Lin, L., Qiang, L., Chi-hung, C., Zhi, J., Yu, E., Towards A Service Requirements Ontology on Knowledge and Intention. *6th International Conference on Quality Software***,** 2006. QSIC 2006.

[50] Yuquin, L., Wenyun, Z., An Ontology Based Approach for Domain Requirements Elicitation and Analysis. *1st International Multi-Symposium Computer and Computational Sciences*, 2006. IMSCCS '06.

[51] Gall, M., Berenbach, B., Towards a Framework for Real Time Requirements Elicitation. *1st International Workshop on Multimedia Requirements Engineering,* 2006. MERE '06.

[52] Kato, J., Saeki, M., Ohnishi, A., Nagata, M., Kaiya, H., Komiyaa, S., Yamamoto, S., Horai, H., Watahiki, K., Package Oriented Requirements Elicitation. *10th Asia Pacific Software Engineering Conference*, 2003.

[53] Jinxin, L., Mark, S., Fox., Taner. B., A Requirement Ontology for Engineering Design

[54] Naveed, I., Saffena, R., Requirements Change Management Process Models: An Evaluation. Proceedings of the 25th conference on IASTED International Multi-Conference: Software Engineering, February 2007

[55] Deepak, S., Yavagal, S., Won, L.., Gail-Joon A., Robin, A. G., Common Criteria Requirements Modeling and its Uses for Quality of Information Assurance (QoIA). *Proceedings of the 43rd annual Southeast regional conference,* 2005

[56] Seok-Won, L., Robin, G., Divya, M., Deepak, Y., Gail-Joon, A., Building Problem Domain Ontology from Security Requirements in Regulatory Documents. *Proceedings of the 2006 international workshop on Software engineering for secure systems*

[57] Huang, Ning., Diao, S., Structure-Based Ontology Evaluation. *IEEE International Conference on e-Business Engineering*, 2006. ICEBE; 06.

[58] Osada. A., Ozawa. D., Kaiya. H., Kaijiri. K., The Role of Domain Knowledge Representation in Requirements Elicitation, *Proceedings of the 25th conference on IASTED International Multi-Conference: Software Engineering,* 2007

[59] Xin,W., Christine,W.C., Howard,J.H., Design of knowledge-based systems with the ontology-domain-system approach. *Proceedings of the 14th international conference on Software engineering and knowledge engineering,* 2002