# Using Ontologies to Synchronize Change in Relational Database Systems

**Waqas Ahmed**
Department of Computer Science, COMSATS Institute of Information Technology, Pakistan
Department of Computer Forensics, Punjab Forensic Science Agency, Pakistan
wqs.ahmed@gmail.com

**Muhammad Ahtisham Aslam**
Department of Computer Science, COMSATS Institute of Information Technology, Pakistan
Faculty of Computing and Information Technology, King Abdul Aziz University, Saudi Arabia
ahtisham_a@hotmail.com

**Antonio A. Lopez-Lorca, Jun Shen and Ghassan Beydoun**
School of Information System and Technology, University of Wollongong, Australia
aall645@uowmail.edu.au, jshen@uow.edu.au, beydoun@uow.edu.au

**Debbie Richards**
Department of Computing, Macquarie University, Australia
deborah.richards@mq.edu.au

*Ontology is a building block of the semantic Web. Ontology building requires a detailed domain analysis, which in turn requires financial resources, intensive domain knowledge and time. Domain models in industry are frequently stored as relational database schemas in relational databases. An ontology base underlying such schemas can represent concepts and relationships that are present in the domain of discourse. However, with ever increasing demand for wider access and domain coverage, public databases are not static and their schemas evolve over time. Ontologies generated according to these databases have to change to reflect the new situation. Once a database schema is changed, these changes in the schema should also be incorporated in any ontology generated from the database. It is not possible to generate a fresh version of the ontology using the new database schema because the ontology itself may have undergone changes that need to be preserved. To tackle this problem, this paper presents a generic framework that will help to generate and synchronize ontologies with existing data sources. In particular we address the translation between ontologies and database schemas, but our proposal is also sufficiently generic to be used to generate and maintain ontologies based on XML and object oriented databases.*

*Keywords: Semantic Web, ontology, relational database schema*

*Categories and Subject Descriptors: I.2.6 [Artificial Intelligence]: Learning – Knowledge acquisition, H.2.4 [Database management]: Systems – Relational databases*

## 1. INTRODUCTION

The World Wide Web (WWW) was originally designed to display information in a human-readable way (Shadbolt, Hall and Berners-Lee, 2006). It was up to the people accessing Web pages to

*Manuscript received: 8 April 2011*
Communicating Editor: Rafael Valencia García

interpret them using their own localized semantics at the time they access them. The forthcoming generation of WWW, the semantic Web, will be oriented to be machine-understandable. Web documents will include their own semantics and computers will be able to intelligently process the information in them (Berners-Lee, Hendler and Lassila, 2001). Several authors have recently explored applications of the Semantic Web to various domains such as customer relationship management (García-Crespo, Colomo-Palacios, Gómez-Berbís and Ruiz-Mezcua, 2010), e-commerce (García-Sánchez, Valencia-García, Martínez-Béjar and Fernández-Breis, 2009), multimedia contents annotation (Paniagua-Martín, García-Crespo, Colomo-Palacios and Ruiz-Mezcua, 2011), e-learning (Chou, Wu, Li and Chen, 2009; Fernández-Breis, Castellanos-Nieves and Valencia-García, 2009), internet access control (Fitzgerald, Foley and Foghlú, 2009; García-Crespo, Gómez-Berbís, Colomo-Palacios and Alor-Hernández, 2011) or systems security (Lasheras, Valencia-García, Fernández-Breis and Toval, 2009; Vorobiev and Bekmamedova, 2010). The building blocks of this new Web are ontologies, understood as formal, explicit specifications of shared conceptualizations (Studer, Benjamins and Fensel, 1998). Ontologies can be used in knowledge-based systems and one of their most important features is their inference capabilities. The ontology building process requires a deep understanding of the domain to identify concepts, relationships between them and axioms defining the rules constraining the relationships. Ontology languages, such as RDF (W3C, 2004a), RDFS (W3C, 2004b) or OWL (W3C, 2009), facilitate this process but development still involves at least an ontology engineer, in charge of formalizing the knowledge provided by the domain expert, who typically will not have skills in ontology development. The development of ontologies is a costly and time consuming process. One way to reduce this cost is to extract ontologies from existing databases. Currently most industrial and research data are stored in relational databases. Usually a detailed domain analysis is performed at the time when these relational databases are built. An automated approach having capability of transforming information from a relational database into an ontology can speed up the process of ontology building. Once generated, the ontology user may further populate it with instances or add more domain details into it according to his or her needs.

Our starting point is a relational database's schema (data model), which contains information about concepts and relationships present in a domain of discourse. We use this information to automatically extract an ontology. The challenge that we address in this paper is maintaining consistency between such an ontology and an evolving/changing database where the schema changes. This is quite a common problem in database applications, particularly where databases are used to store data about an evolving domain, e.g. research data (Kupfer, Eckstein, Neumann and Mathiak, 2006), and where a change in the schema is often a reflection of new understanding in the domain. Under such circumstances, having a database that more accurately defines the domain and incorporates latest changes in the domain model is a lot simpler if an ontology is previously used to describe the old schema generated from the old database (Noy and Klein, 2004). Using an ontology in the updating the schema of the database will help maintaining consistency of the database. This can be easier undertaken by co-evolving the two representations, the database metadata and the ontology, through careful maintenance procedures and consideration of any manual changes in the ontology. This paper presents a framework to generate and then synchronize a generated ontology with source database schema to ensure that changes in domain conceptualizations are captured in any later database design as the domain itself is evolved.

The rest of this paper is structured as follows: Section 2 discusses related work highlighting shortcomings of existing techniques in database maintenance when domains themselves evolve. Section 3 shows how ontologies can be constructed from relational database metadata which later in Section 4 will be used to construct a mechanism to detect and implement changes in the automatically generated ontology. Section 5 describes a tool realization of the framework: DATAONTO, an automated ontology generation and change synchronization plug-in for Protégé. Section 6 closes the paper with conclusions and future work.

## 2. RELATED WORK

Using ontologies to maintain and manage the evolution of conceptual models in software systems is not new. There are numerous examples on validation of conceptual models using ontologies. For instance Shanks *et al* (2003) use a formal ontology to validate the choice of a specific suitable language for conceptual modeling of a given domain. Benevides and Guizzardi (2009) propose an Eclipse-based tool to build and automatically verify conceptual models developed in a language (OntoUML) that uses a foundation ontology to extend UML. In Lopez-Lorca, Beydoun, Sterling and Miller (2010) ontologies are used to properly propagate requirements analysis updates in the requirements models focussing on the requirements activities of validation and verification as characterised in Sadraei, Aurum, Beydoun and Paech (2007). In Beydoun, Low, Mouratidis and Henderson-Sellers (2009) and Tran, Low and Beydoun (2006), ontologies are used to support systems analysis and design.

Established software systems have their domain knowledge embedded in them. Some authors advocate the use of ontologies to recover this implicit knowledge for improving the understanding of the system or for maintenance purposes. Meng, Rilling, Zhang, Witte and Charland (2006) propose an ontology-based method to support software maintenance. They argue that software comprehension is essential for its maintenance, thus, they model the comprehension process by means of an ontology. Reasoning mechanisms provide the user with suggestions of tools suitable for certain comprehension tasks. Daga, Cesare, Lycett and Partridge (2005) tackle the problem of renovation of legacy systems from a technology-independent point of view. They focus on using ontologies to recover the business knowledge, which is an important asset for the company and is often forgotten when legacy systems are adapted. Ontologies provide improved semantics, better interoperability, less complexity and technology independence.

Along the same lines, but focusing on databases, several authors have developed tools to extract ontologies from databases. Mukhopadhyay, Banik and Mukherjee (2007) present an approach to construct RDF ontologies from existing databases. Their proposal uses RDF as ontology language which is less expressive than OWL. This approach uses the database physical schema to construct corresponding classes. As the physical schema has no associated semantics, the transformation may result in an inaccurate representation of the domain. For example, if there is a bridge table in the relational database then it will also be transformed into an RDF class. Our approach is more expressive and keeps the semantics implicit in the database as it is based on OWL and extracts the conceptual model from the database schema.

Cullot, Ghawi and Yetongnon (2007) present DB2OWL, a local automatic database to ontology mapping tool that focuses on local ontology generation from relational databases. One shortcoming with the approach presented is that it uses predefined table cases and maps directly physical schema into an ontology. Another issue with the tool is that the object properties are defined but not added to the class as restrictions. This implies that only the sufficient criteria to be the instance of a class is provided but necessary and sufficient criteria are not given. Our proposal is more flexible as it does not use predefined table cases, but uses an intermediate conceptual model to analyse the database. It also defines object properties such as "has a".

Trinkunas and Vasilecas (2007) present an approach to construct ontologies from relational databases using reverse engineering. The proposed approach converts a conceptual model into an ontology but it requires an existing conceptual model as input. Most of the time the conceptual models of operational databases are not available therefore, ontology generation will not be possible in that case. Our approach is based as well on a conceptual model but it is designed to extract it from the database schema, so no pre-existing conceptual model is required.

Upadhyaya and Kumar (2005) take an extended ER model as input to construct an ontology out of it. As ontology is a relatively new construct as compared to relational databases, the relational databases which are in operation often do not have associated models or supporting documentation. Therefore, it is difficult to find the extended ER model and then transform it into an ontology. Similarly to the approach of Trinkunas and Vasilecas (2007), in this work the authors rely on the existence of a previous model.

Our approach is different from the analysed existing works in that it does not rely on existing models, but builds a conceptual model representing the database, using the database schema. As we are using OWL the semantics are preserved. And more importantly, unlike any of the related works, we provide an automatic mechanism to evolve the ontology from changes in the original databases while preserving manual changes made to the ontology.

## 3. ONTOLOGY GENERATION FROM RELATIONAL DATABASES

Ontology generation from a database requires knowledge of the relational database schema. This section first discusses what schema information is read from the database then how this schema information is mapped as ontology constructs. Before transforming the database schema into an ontology, a mapping scheme is required that can perform transformations of database schema into ontology elements. This mapping scheme is also discussed.

### 3.1 Database Schema to Conceptual Model Transformation

Database schemas contain information of tables, columns, columns' datatype, primary and foreign keys and other constructs such as stored procedures, views and triggers. All these details are dependent on the RDBMS used to implement the database (Elmasri and Navathe, 2010). The conceptual model of a database provides insight about different entities present in the domain of discourse, their attributes and relationships (Felden and Kilimann, 2006). When a conceptual model is transformed into a physical implementation schema, the semantics which are associated with the conceptual model are lost (Trinkunas and Vasilecas, 2007). These semantics are very important from the ontology's point of view. An example of these lost semantics is transformation of a bridge table that is a result of a many-to-many relationship between two entities. At the time of converting the conceptual model into a relational database schema, a many-to-many relationship results in three tables. The third table that represents this many-to-many relation consists of primary keys of two participating entities as its attributes. These primary keys of individual participating entities constitute a composite primary key in the bridge table (Connolly and Begg, 2004). This table did not exist in the original domain model that was captured using conceptual modeling. If the ontology is generated directly from this schema then it will certainly lose domain information like this therefore, it is required to transform physical schema into corresponding conceptual representation.

The conceptual graph is a data model that represents higher level details of how data is stored in a database (Chen, 1976). In a prototype implementation of the tool DATAONTO, this conceptual data model is represented in the form of a directed label graph. Formally we can define this graph $G$ as $G = \{N, E\}$, stating that graph $G$ is a finite set of labeled nodes $N$ and labeled edges $E$. Each node represents an entity in the database and edge represents the relationship between entities

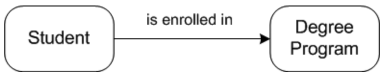


**Figure 1: An example of a conceptual model**

(Trinkunas and Vasilecas, 2007). The direction of the edge represents the direction of the original constraint present in the database. The label of the edge is the relationship name. Each edge is written as a triplet $E = [N_1, \alpha, N_2]$ where $N_1$ and $N_2$ are the nodes that are connected by the edge and $\alpha$ is the label of the edge. Figure 1 shows an excerpt of a conceptual graph, in which two nodes, *Student* and *Degree Program*, are related to each other by means of the directed edge representing the relationship *is enrolled in*.

Every node represents an entity in the database. A node has its *unique name* and *attributes* with their *datatype*. These attributes are actually the columns of a database table with their datatype values. An edge contains the *node names* that it is connecting and a *label* that represents the relationship between the two nodes.

Before a model can be constructed, the information about the database schema (database metadata) is required. The following information about each table is read before converting it into a model:

1. Name of table.
2. List of columns in the table along with their datatype.
3. List of primary keys.
4. List of foreign keys along with the names of tables in which these are primary keys.

Once this information is available, a conceptual graph model can be constructed from a database table, *T*, using the following four Mapping Rules (MR) that have been logically described in Table 1:

**MR1.** Table *T* is said to be a *bridge table* if it connects tables $T_1$ and $T_2$ in such a way that it has two distinct subsets of columns $A_1$ and $A_2$, each set belonging to the table $T_1$ and $T_2$ respectively. Attributes $A_1$ and $A_2$ are foreign and primary keys in *T*. This will be mapped in the conceptual graph as two edges, one directed from table $T_1$ to $T_2$ and another from $T_2$ to $T_1$. The label of these new edges is set to *has a* (Figure 2).

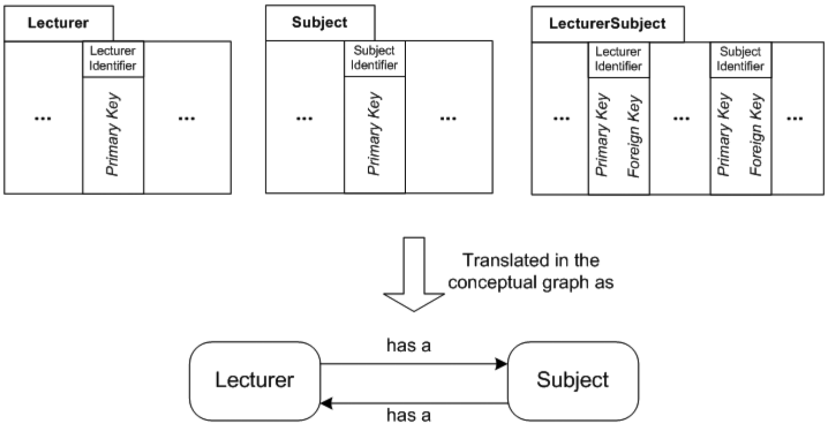


**Figure 2: An example of the Mapping Rule 1 (MR1)**

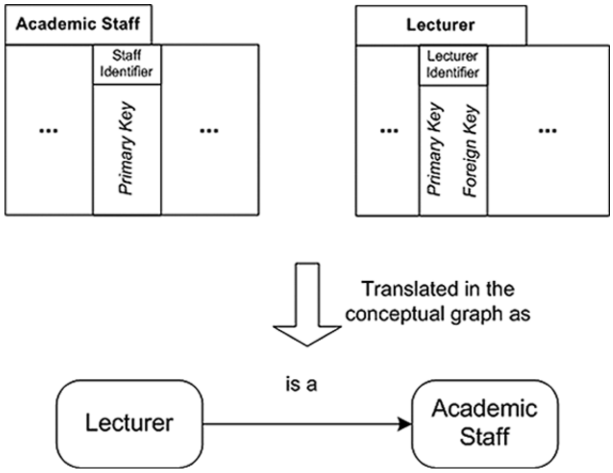


**Figure 3: An example of the Mapping Rule 2 (MR2)**

**MR2.** Table *T* is said to be a subclass of table $T_1$ if table *T* has a referential integrity constraint with table $T_1$ such that the primary key of table $T_1$ is a foreign key in table *T* and it is also a primary key in table *T*. This will be mapped in the conceptual graph as an edge from table $T_1$ to *T* labeled as *is a* (Figure 3).

**MR3.** If table *T* has a foreign key that is a primary key in the table $T_1$ each instance of table $T_1$ is said to be related to multiple instances of *T*. This will be mapped in the conceptual graph as an edge from *T* to $T_1$ labeled as *has a* (Figure 4).

**MR4.** Each table *T* in the database is considered a node to be added to the conceptual graph. Columns of every *T* with their associated datatype are added as attributes. Columns acting

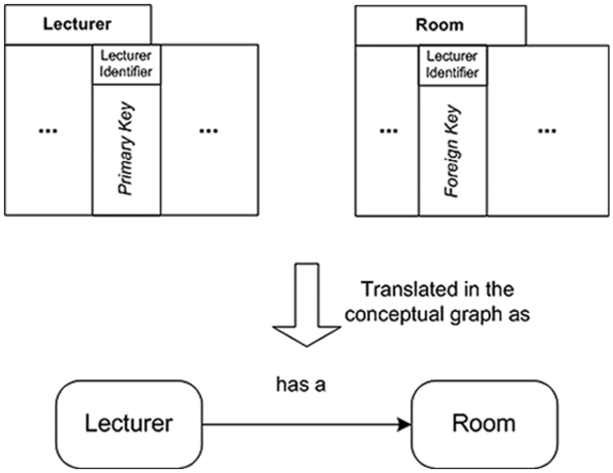


**Figure 4: An example of the Mapping Rule 3 (MR3)**

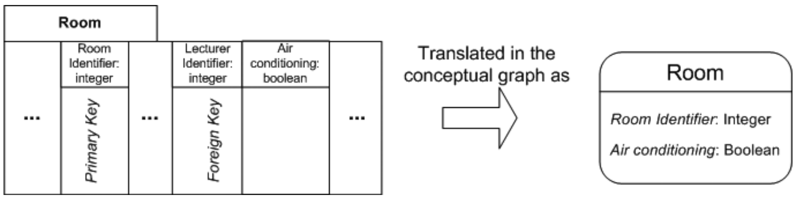

Room
Room Identifier: integer
Lecturer Identifier: integer
Air conditioning: boolean
Primary Key
Foreign Key
...
Translated in the conceptual graph as
Room
Room Identifier: Integer
Air conditioning: Boolean

**Figure 5: An example of the Mapping Rule 4 (MR4)**

| | |
|---|---|
| **Rule Id** | MR1 |
| **Determines** | Bridge table |
| **Rule** | $\exists A_1, A_2 \in T$ / $A_1 \in T_1, A_2 \in T_2, (A_1, A_2)$ *isPrimaryKey(T) and* $(A_1, A_2)$ *isForeignKey(T)* |
| **Mapping** | Two edges *has a* from $T_1$ to $T_2$ and from $T_2$ to $T_1$ respectively |
| **Rule Id** | MR2 |
| **Determines** | Class/subclass |
| **Rule** | $\exists A_1 \in T_1, A_2 \in T_2$ / $A_1$ *isPrimaryKey*$(T_1)$, $A_2$ *isPrimaryKey*$(T_2)$, $A_2$ *isForeignKey*$(T_2)$ *and* $A_1 = A_2$ |
| **Mapping** | Edge *is a* from $T_1$ to $T_2$ |
| **Rule Id** | MR3 |
| **Determines** | Relationship |
| **Rule** | $\exists A_1 \in T_1, A_2 \in T_2$ / $A_1$ *isForeignKey*$(T_1)$, $A_2$ *isPrimaryKey*$(T_2)$ *and* $A_1 = A_2$ |
| **Mapping** | Edge has *a* from $T_1$ to $T_2$ |
| **Rule Id** | MR4 |
| **Determines** | Node |
| **Rule** | $\forall T_i$ |
| **Mapping** | A new node for each $T_i$. |

**Table 1: Summary of mapping rules between database schema and conceptual graph, being $T_i$ tables of the database and $A_i$ columns of the tables $T_i$**

as foreign keys in the database will not be added. If a primary key is also a foreign key, it will be added as a node attribute (Figure 5).

## 3.2 Conceptual Model to Ontology Transformation

Once the database schema is read and transformed into a conceptual graph model, this graph model can be used to construct the ontology. An ontology is formed by classes and properties. These properties can be object properties and datatype properties. Object properties are used to represent a relationship between two classes. Datatype properties describe a relationship between an individual and data values (W3C, 2009). If we want to construct an ontology from a relational database conceptual model then we require a mapping scheme. This mapping scheme will tell which relational database conceptual model element is transformed into what ontology constructs.

An entity (also referred to as an entity type) is a basic object of conceptual database model that represents a thing which has independent physical or logical existence (Elmasri and Navathe, 2010). In Connolly and Begg (2004) an entity type is defined as the group of objects that exists in the real world and has the same properties. An ontology class defines a group of individuals that share some common characteristics (W3C, 2009). It is in fact a set that contains individuals that fulfill certain criteria (Horridge, Jupp, Moulton, Rector, Stevens and Wroe, 2007).

It is clear from the definitions of entity type and ontology class that both contain individuals sharing some common properties or characteristics. Since a node in a conceptual model represents an entity in a domain of discourse therefore, it may become a class in an ontology. We define the first Ontology Mapping Rule (OMR) as:

**OMR1.** Each node in the conceptual graph is mapped as a class in the ontology. To define necessary and sufficient criteria for being a member of the class, class will be an intersection of all restrictions on its datatype and object properties.

A set of properties that completely define an entity type are called its attributes (Elmasri and Navathe, 2010). Attributes have values that comprise the main part of the data stored in database (Connolly and Begg, 2004). OWL datatype properties represent the relationship of an individual with a data value. Keeping this similarity in mind, we can replace node attributes with OWL datatype properties in the ontology. Therefore, we define the second ontology mapping rule as:

**OMR2.** All node attributes are mapped as datatype properties in the ontology. The range of the datatype property is set to XML Schema Datatype (XSD) (W3C (2004c), (2004d), (2004e)), the equivalent to an attribute's datatype in a relational database. As attributes with the same name may belong to more than one node, the node name has to be appended as prefix to the attribute name.

Whenever an attribute of an entity type refers to another attribute of some other entity, then there exists a relationship between these two entities. If an entity type can be mapped as an ontology class then there is also a need to map relationships between entity types. These relationships can be mapped by using object properties of the ontology. In OWL the object properties are used to represent some relationships between classes. So relationships which exist in the conceptual database model will be transformed into OWL object properties. An edge with label has a represents a one-to-many relationship between the nodes it is connecting. This relationship can be mapped by adding an object property in the OWL ontology. The third ontology mapping rule will be:

**OMR3.** If an edge with label *has a* connects node $N_1$ with node $N_2$ then an object property *OP* named as $hasN_2$ will be created in the ontology. Domain and range of property *OP* will be set to $N_1$ and $N_2$ respectively. The property will also be made functional to make sure that it connects only one instance at a time.

Entities can be specialized in the form of a specialization hierarchy. This hierarchy is very important if these entities are to be transformed into an ontology. Each sub entity can be a subclass of its corresponding specialized entity class. An edge with label is a is mapped as class/subclass relation in the ontology. We define the fourth and last ontology mapping rule as:

| Rule ID | Conceptual Graph | Ontology |
|---|---|---|
| OMR1 | Node | Class |
| OMR2 | Node attribute | Datatype property |
| OMR3 | Edge *has a* from $N_1$ to $N_2$ | Functional object property, $hasN_2$, with $N_1$ as domain and $N_2$ as range |
| OMR4 | Edge *is a* from $N_1$ to $N_2$ | $N_1$ subclass of $N_2$ |

**Table 2: Summary of ontology mapping rules**

**OMR4.** If an edge $E$ with label *is a* connects two nodes $N_1$ and $N_2$ then in the ontology, $N_1$ will be mapped as subclass of $N_2$.

Table 2 logically describes the conceptual graph to ontology mapping rules.

## 4. DETECTION OF CHANGES IN A DATABASE SCHEMA AND IMPLEMENTATION OF THESE CHANGES INTO AN ONTOLOGY

Public and research databases are not static and their schema keeps on changing (Kupfer *et al*, 2006). New tables are added, existing tables are deleted and new columns are added or deleted from existing tables. If an ontology is generated from a database schema considering the fact that this schema contains domain information then this change in schema will directly imply a change in the domain model. Once the domain model is changed, the ontology representing a domain of discourse should also be changed.

Once the ontology is generated from a database schema using the technique described in Section 3, this database schema may go through changes. To keep the generated local ontology consistent with the database schema, it is necessary to detect and then implement schema changes into the existing ontology. A database schema may go through the following changes:

1. Addition of new table(s).
2. Deletion of table(s).
3. Addition of column(s) in table(s).
4. Deletion of column(s) in table(s).
5. Change of datatype of column(s).
6. Addition of relationship between tables.
7. Deletion of relationship between tables.

The process of change detection begins by constructing a conceptual model graph from the changed database schema. The construction of a conceptual graph is the same as discussed in Section 3.1. Once we have the new conceptual model, it is compared with the previous version of the conceptual model that was used to generate the ontology. The difference between two models represents the changes in schema. These changes are maintained in a change set. This change set is then used to map these changes into the ontology. In what follows we define the Change Detection Rules (CDR), which are the operators that we use to detect differences between conceptual models (summarized in Table 3). $G_N$ and $G_O$ symbolize the newly generated and old graph model, respectively.

**CDR1.** If a node $N$ exists in $G_O$ but is not present in $G_N$ then it has been deleted from the new version of the database.

**CDR2.** If a node $N$ exists in $G_N$ but is not present in $G_O$ then it has been added to the new version of the database.

**CDR3.** If a node attribute $A$ is part of node $N$ in $G_N$ but the same node $N$ does not have such attribute $A$ in $G_O$ then $A$ has been added to table $N$ in the new version of the database.

**CDR4.** If a node attribute $A$ is part of node $N$ in $G_O$ but the same node $N$ does not have such attribute $A$ in $G_N$ then $A$ has been removed from table $N$ in the new version of the database.

**CDR5.** If a node $N$ having an attribute $A$ exists in both $G_N$ and $G_O$ but the datatype of $A$ in $G_N$ is $D_1$ and the datatype of $A$ in $G_O$ is $A_2$ and $A_1$ is different from $A_2$, then the datatype of $A$ has changed in the new version of the database.

**CDR6.** If an edge $E$, labeled $L$ connects two nodes $N_1$ and $N_2$ in $G_N$ but the same edge does not exist in $G_O$ then $E$ represents a newly added relationship between nodes $N_1$ and $N_2$ in the new version of the database.

**CDR7.** If an edge $E$, labeled $L$ connects two nodes $N_1$ and $N_2$ in $G_O$ but the same edge does not exist in $G_N$ then $E$ represents a deleted relationship between nodes $N_1$ and $N_2$ in the new version of the database.

## 4.1 Change Set

The above described CDRs can be used to detect changes in the database schema. Once changes are detected they are stored in the form of a change set. This change set is then used to implement these changes into the ontology. A change set consists of following items:

1. A conceptual graph model.
2. Names of nodes to be deleted.
3. Names of edges to be deleted.
4. Names of attributes to be deleted along with their node names.
5. Names of attributes whose datatypes are to be changed along with their class name and new datatype value.

| Rule ID | Detects | Rule |
|---|---|---|
| CDR1 | Deleted table | $\exists N_1 \in G_O, \sim\exists N_2 \in G_N / N_1 = N_2$ |
| CDR2 | Added table | $\sim\exists N_1 \in G_O, \exists N_2 \in G_N / N_1 = N_2$ |
| CDR3 | Added column | $A \notin N / N \in G_O$ *and* $A \in N / N \in G_N$ |
| CDR4 | Deleted column | $A \in N / N \in G_O$ *and* $A \in N / N \in G_N$ |
| CDR5 | Change of datatype | $A_1 \in N / N \in G_O$ *and* $A_2 \in N / N \in G_N$ *and* $A_1 = A_2$ *and* $DataType(A_1) \neq DataType(A_2)$ |
| CDR6 | Addition of relationship | $\sim Connects(E, N_1, N2) / E, N_1, N_2 \in G_O$ *and* $Connects(E, N_1, N_2) / E, N_1, N_2 \in G_N$ |
| CDR7 | Deletion of relationship | $Connects(E, N_1, N_2) / E, N_1, N_2 \in G_O$ *and* $\sim Connects(E, N_1, N_2) / E, N_1, N_2 \in G_N$ |

**Table 3: Change detection rules. $G_O$ and $G_N$ represent the old and new conceptual graphs respectively. *N* represents nodes of the graph (similarly tables in the database). *A* represents attributes of the nodes (similarly columns of tables in the database).**

One thing to be noted here is that the above mentioned list does not contain the names of newly added nodes and edges but it contains a graph model. This graph model is actually a subset of changes that represents newly added nodes and edges. Figure 6 shows the process of constructing a change set.

Before a change is made into ontology it is necessary to know which conceptual model element was mapped to what ontology construct. Once the changes to be made are identified, they can now be implemented into the ontology. For this purpose we may use the rules specified in Section 3.1. It is clear that a node in the graph model is mapped as a class, an edge is mapped as a relationship and an attribute is mapped as datatype properties. In what follows we present the Change Mapping Rules (CMR) to implement changes into an ontology.

**CMR1.** The conceptual graph present in the change set will be mapped in a similar way to the graph transformation specified in Section 3.2. This will add new classes and relationships between these new classes to the ontology.

**CMR2.** For each node present in the list of nodes to be deleted in the change set, the corresponding class in the ontology will be deleted. All datatype properties of this class will also be deleted.

**CMR3.** For each edge $E$ labeled as *has a* connecting nodes $N_1$ and $N_2$, in the list of new edges in the change set, an object property with domain $N_1$ and range $N_2$ will be added to $N_1$. If edge label is *is a* then class corresponding node $N_1$ will be made subclass of class represented by $N_2$.

**CMR4.** For each edge $E$ labeled as *has a* connecting nodes $N_1$ and $N_2$, in the list of edges to be deleted in the change list, the object property corresponding to this edge in the ontology will be deleted. If the edge label is *is a* then the class/subclass relation between $N_2$ and $N_1$ will be removed.

**CMR5.** For each element in the list of attributes to be added in the change set, a new datatype property $P$ will be added to the class corresponding to the node the attribute belongs to. The range of datatype property will be set as the XSD equivalent of SQL datatype.

**CMR6.** For each element in the list of attribute datatypes to be changed in the change set, the range of the datatype property $P$ that corresponds to that attribute in the ontology will be changed to new XSD equivalent of SQL datatype.

**CMR7.** For each element in the list of attributes to be deleted in the change set, the datatype property $P$ that represents the attribute in ontology will be deleted.

Once these changes have been implemented into the ontology, the new graph model $G_N$ is kept for future change detection and implementation purposes.

## 5. DATAONTO: A CONVERSION AND SYNCHRONIZATION PLUG-IN FOR PROTÉGÉ

DATAONTO (2009) is a plug-in for the widely used ontology editor, Protégé (2011), developed using the presented framework. Protégé-based tools have been widely used before, e.g. Girardi and Leite (2008) present a Protégé-based tool for the development of families of Multi Agent Systems (MAS). Hajnal, Pedone and Varga (2007) present a Protégé plug-in to develop MAS models. Knublauch (2004) advocates the use of Protégé to develop applications for the semantic Web highlighting quick prototyping as a key appeal. Similar to those works, our plug-in tool produces

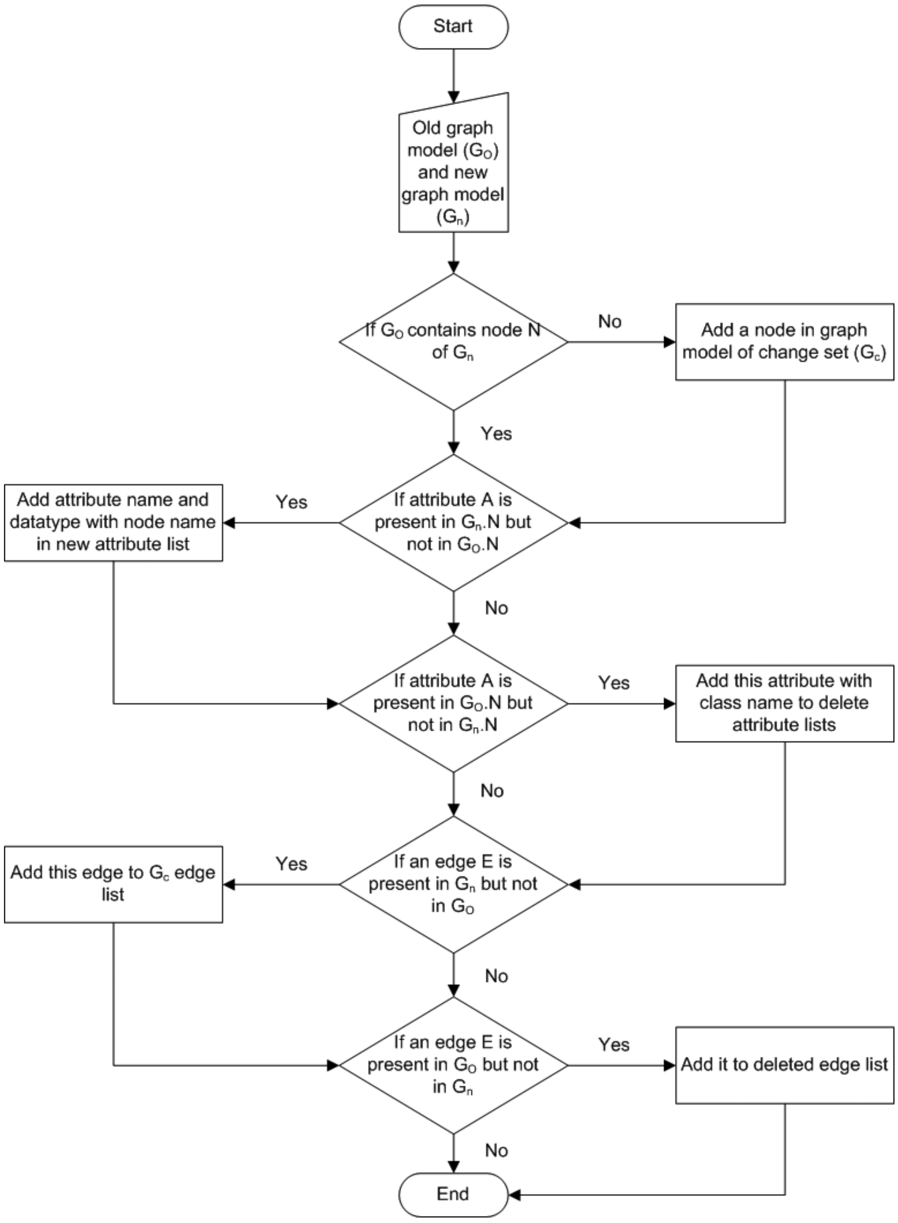

Start
Old graph model ($G_O$) and new graph model ($G_n$)
If $G_O$ contains node N of $G_n$
No
Add a node in graph model of change set ($G_c$)
Yes
If attribute A is present in $G_n$.N but not in $G_O$.N
Yes
Add attribute name and datatype with node name in new attribute list
No
If attribute A is present in $G_O$.N but not in $G_n$.N
Yes
Add this attribute with class name to delete attribute lists
No
If an edge E is present in $G_n$ but not in $G_O$
Yes
Add this edge to $G_c$ edge list
No
If an edge E is present in $G_O$ but not in $G_n$
Yes
Add it to deleted edge list
No
End

**Figure 6: Process of change set construction**

an OWL-DL ontology. It transforms a database schema into an OWL-DL ontology using the semantic mappings described in Section 3. Once the ontology is generated, DATAONTO also looks for any database schema changes in the source database and implements those in the existing ontology. JAVA JDBC API (ORACLE, 2010) is used for reading the database schema and Jena 2.1

(Jena, 2010) is utilized for writing or reading ontology files. Some important features of DATAONTO are the following:

– DATAONTO requires only database location and access permissions as input. It automatically processes its schema and transforms it into a conceptual model.

– When a conceptual domain model is transformed into a database physical schema, the associated semantics are lost (Upadhyaya and Kumar, 2005). DATAONTO, to acquire these implicit semantics, automatically transforms physical schema into a conceptual graph model. This graph model contains the relationships and attributes of entities. Association and generalization of entities is expressed in a graph model using edges having labels *has a* and *is a*.

– DATAONTO uses OWL-DL as ontology definition language. This provides more expressiveness and inference capabilities as compared to OWL-Lite (W3C, 2009). Class hierarchies are computed and specified in the output ontology. Moreover, classes are defined as intersections of restrictions on object and data properties.

– To make an ontology consistent with a database, existing approaches have only one possible solution, i.e. regenerate the ontology from the changed database schema. This regeneration of ontology will result in a loss of all manual annotations in the ontology. DATAONTO detects changes in database schema that are not part of the ontology. These changes are computed by comparing the domain model, from which the ontology was generated, with the changed domain model. Detected changes are maintained in the change set

– DATAONTO automatically implements detected changes into the ontology.

DATAONTO Java implementation is platform independent. JDBC is an open source solution provided by Sun is used for database connectivity and schema reading purposes. This enables database independent connectivity through provision of specific APIs that give access to multiple data sources. For the purpose of manipulating OWL-DL ontologies, Jena 2.1 API is used. Jena is an open source, well maintained Java framework that can be used to build semantic Web applications. It supports RDF, RDFS, OWL and SPARQL along with a rule based inference engine.

The architecture of DATAONTO consists of two modules (Ontology Generator and Change Mapper). The Ontology Generator module generates a conceptual graph model from a given database schema along with its associated OWL-DL ontology. The technique and mapping rules to transform the database schema into a conceptual graph model and then this conceptual graph model into an OWL-DL ontology have been described in Section 3. Figure 7 shows the architecture of the Ontology Generator.

The Change Mapper module of DATAONTO detects changes in a database schema and then maps back these changes into the ontology that was generated from that very database. Figure 8 details the architecture of the Change Mapper module.

The Java implementation of DATAONTO consists of three packages:

1. *Schema Reader.* This package contains classes which establish a database connection, read the database schema, and identify and separate table keys, columns and their datatype.

2. *Graph Generator.* This package consists of a graph class and a graph generator. The graph generator class takes database table information manipulated by the Schema Reader package and transforms it into a conceptual graph model.

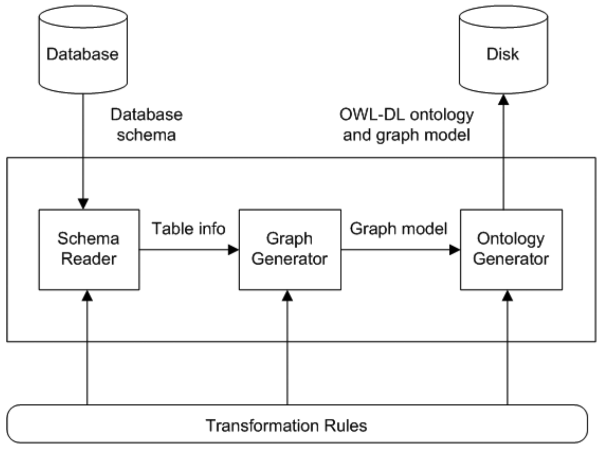


**Figure 7: Architecture of the Ontology Generator module**

3. *Ontology Generator.* This package implements two major functionalities of DATAONTO: The generation of a new ontology from graph model; and the implementation of changes into an existing ontology. The change set is also processed by this package in such a way that changes which are part of the change set are implemented into the existing ontology using the change mapper function of the Ontology Generator package.

DATAONTO can be used to generate a new ontology from a database or synchronize changes between a database schema and already generated ontology from the same database. For the purpose of detecting changes between the database schema and ontology, a conceptual graph model that represents the updated state of the ontology is required as input. This existing conceptual graph model and the model generated from the database schema are compared and then changes are detected and mapped into the ontology.

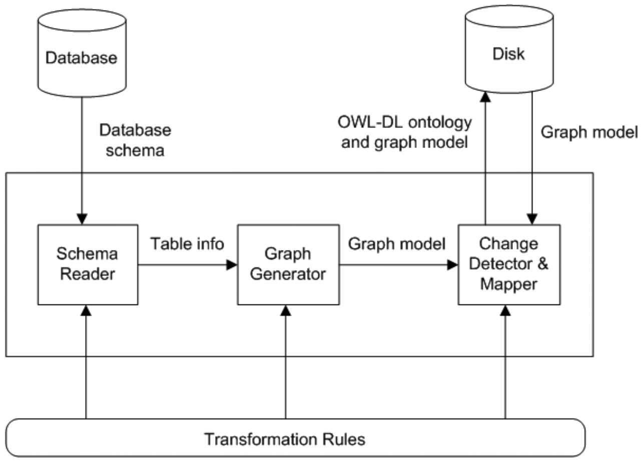
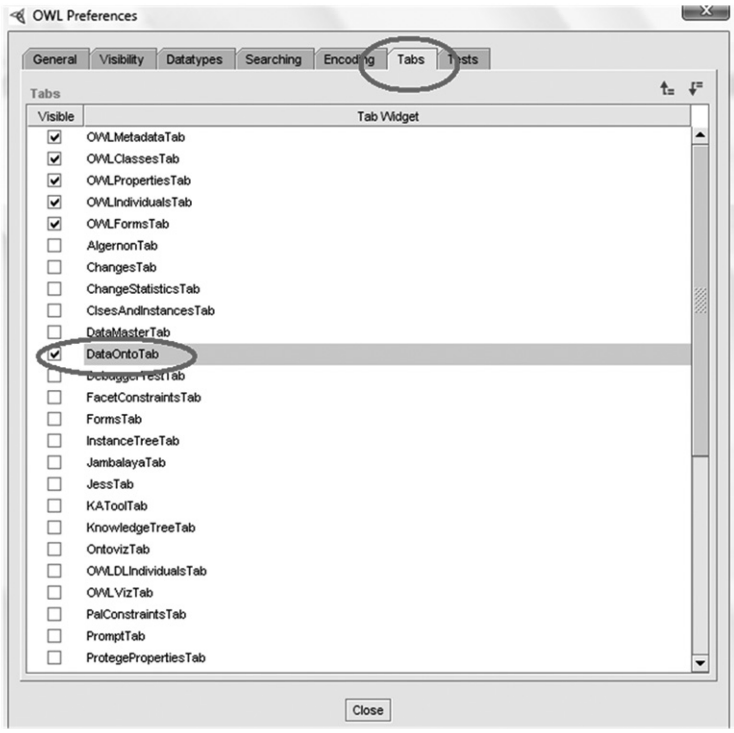


**Figure 8: Architecture of the Change Mapper module**


OWL Preferences

General | Visibility | Datatypes | Searching | Encoding | Tabs | Tests

Tabs

| Visible | Tab Widget |
|---|---|
| ✓ | OWLMetadataTab |
| ✓ | OWLClassesTab |
| ✓ | OWLPropertiesTab |
| ✓ | OWLIndividualsTab |
| ✓ | OWLFormsTab |
| ☐ | AlgernonTab |
| ☐ | ChangesTab |
| ☐ | ChangeStatisticsTab |
| ☐ | ClsesAndInstancesTab |
| ☐ | DataMasterTab |
| ✓ | DataOntoTab |
| ☐ | DebuggerTestTab |
| ☐ | FacetConstraintsTab |
| ☐ | FormsTab |
| ☐ | InstanceTreeTab |
| ☐ | JambalayaTab |
| ☐ | JessTab |
| ☐ | KAToolTab |
| ☐ | KnowledgeTreeTab |
| ☐ | OntovizTab |
| ☐ | OWLDLIndividualsTab |
| ☐ | OWLVizTab |
| ☐ | PalConstraintsTab |
| ☐ | PromptTab |
| ☐ | ProtegePropertiesTab |

Close

**Figure 9: Selecting DataOntoTab to become a Protégé plug-in.**

Setting DATAONTO up in Protégé as a plug-in is a very simple task. DATAONTO is first included in the plug-in directory inside the Protégé installation directory. To display DATAONTO as an option tab in Protégé, in the preferences window of OWL, select the DATAONTO plug-in (as shown in Figure 9). DATAONTO will then become visible in the Protégé tab list and will be ready to use.

## 6. CONCLUSIONS, DISCUSSION AND FUTURE WORK

One of the problems that delay the development of the awaited semantic Web is the lack of ontologies. Building ontologies is an onerous task and requires a great deal of domain knowledge. Much knowledge is implicitly stored in existing applications, such as databases. In this paper, we focussed on how to extract this knowledge from relational databases. This is significant because of the widespread use of relational database systems. However, our proposal is sufficiently generic to be also applied to other types of databases. In this context, we presented a generic framework for ontology generation and change synchronization from relational databases. Our proposed framework is supported by implementing a tool DATAONTO that uses all the above mentioned various components to automatically build and synchronize changes via the use of an ontology into the relational database. DATAONTO automatically reads a relational database's schema, converts it into a conceptual graph model to acquire lost semantics and then maps it into an ontology using the mapping rules. Once this ontology is thus made available, the user can check the source database schema for changes and these changes are detected by comparing the two conceptual models. If changes are found they are implemented in the database and the existing ontology (for reference in any future change cycles).

We plan to validate our approach in a controlled experiment consisting of two stages. In the first stage we will provide a group ($G_1$) of information systems students with a database depicting characteristics and components of smartphones. We will ask them to use our proposed mechanism to automatically generate an ontology ($O_1$) from the database schema and, later on, manually add new knowledge to it. In the second stage we will give the same students ($G_1$) a new version of the database, in which some tables and relations have changed because of new developments in the state of the art of smartphones. They will be asked to use the evolution mechanism presented in this paper to automatically generate an updated version of the ontology ($O_2$) matching the changes in the database but without losing the manual changes made to the old version of the ontology. In parallel, a second group of students ($G_2$, disjoint with $G_1$) will be given specifications to develop, using traditional ontology engineering techniques ontologies $O_1$ and $O_2$. To evaluate the results of the experiment, we will compare the ontologies developed by $G_1$ and $G_2$, paying special attention to the feedback of the students in terms of developing time, ease of development and usability of the mechanism. We have chosen the smartphone domain because it is a highly dynamic field. Every day new phones are released making obsolete devices that are only months-old. Semantic web applications in this domain would have to tackle the problem of how to keep updated their smartphone specification ontologies in an ever changing market. Our experiment replicates this scenario realistically.

We are currently working on other directions as well to extend our approach. We plan to extend the mechanisms presented in this paper to identify mapping rules for XML and object oriented database schema so that ontologies can also be generated from them. We are developing a stable release of DATAONTO as a plug-in for widely used ontology editor Protégé. We aim to have DATAONTO as a general tool that can generate up-to-date ontology skeletons that are accurate and domain specific from any data source.

## REFERENCES

BENEVIDES, A.B. and GUIZZARDI, G. (2009): A model-based tool for conceptual modeling and domain ontology engineering in OntoUML. *Proceedings of International Conference on Enterprise Information Systems*.

BERNERS-LEE, T., HENDLER, J. and LASSILA, O. (2001): The semantic web. *Scientific American*.

BEYDOUN, G., LOW, G., MOURATIDIS, H. and HENDERSON-SELLERS, B. (2009): A security-aware metamodel for multi-agent systems. *Information and Software Technology* (51)5: 832–845.

CHEN, P.P-S. (1976): The entity-relationship model—Toward a unified view of data. *ACM Transactions on Database Systems* (1)1: 9–36.

CHOU, P-H., WU, M-J., LI, P-H. and CHEN, K-K. (2009): Accessing E-Learners' knowledge for personalization in E-Learning environment. *Journal of Research and Practice in Information Technology* (41)4: 295–318.

CONNOLLY, T.M. and BEGG, C.E. (2004): *Database Systems: A Practical Approach to Design, Implementation and Management*, Addison Wesley.

CULLOT, N., GHAWI, R. and YETONGNON, K. (2007): DB2OWL: A tool for automatic database-to-ontology mapping. *Proceedings of 15th Italian Symposium on Advanced Database Systems (SEBD 2007)*: 491–494.

DAGA, A., CESARE, S.D., LYCETT, M. and PARTRIDGE, C. (2005): An ontological approach for recovering legacy business content. *Proceedings of 38th Annual Hawaii International Conference on System Sciences*: 224–232.

DATAONTO (2009): DATAONTO: A database to OWL conversion and synchronization plug-in for Protégé. https://source forge.net/projects/dataonto/. Accessed 28 March 2011.

ELMASRI, R. and NAVATHE, S. (2010): *Fundamentals of Database Systems*, Addison Wesley.

FELDEN, C. and KILIMANN, D. (2006): Deployment of ontologies in business intelligence systems. *Proceedings of 8th International Conference on Enterprise Information Systems: Databases and Information Systems Integration (ICEIS 2006)*.

FERNÁNDEZ-BREIS, J.T., CASTELLANOS-NIEVES, D. and VALENCIA-GARCÍA, R. (2009): Measuring individual learning performance in group work from a knowledge integration perspective. *Information Sciences* (179)4: 339–354.

FITZGERALD, W.M., FOLEY, S.N. and FOGHLÚ, M.Ó. (2009): Network access control configuration management using semantic web techniques. *Journal of Research and Practice in Information Technology* (41)2: 99–117.

GARCÍA-CRESPO, Á., COLOMO-PALACIOS, R., GÓMEZ-BERBÍS, J.M. and RUIZ-MEZCUA, B. (2010): SEMO: A framework for customer social networks analysis based on semantics. *Journal of Information Technology*, (25)2: 178–188.

GARCÍA-CRESPO, Á., GÓMEZ-BERBÍS, J.M., COLOMO-PALACIOS, R. and ALOR-HENÁNDEZ, G. (2011): SecurOntology: A semantic web access control framework. *Computer Standards & Interfaces* (33)1: 42–49.

GARCÍA-SÁNCHEZ, F., VALENCIA-GARCÍA, R., MARTÍNEZ-BÉJAR, R. and FERNÁNDEZ-BREIS, J.T. (2009): An ontology, intelligent agent-based framework for the provision of semantic web services. *Expert Systems with Applications* (36)2 Part 2: 3167–3187.

GIRARDI, R. and LEITE, A. (2008): A knowledge-based tool for multi-agent domain engineering. *Knowledge-Based Systems* (21)7: 604–611.

HAJNAL, A., PEDONE, G. and VARGA, L. (2007): Ontology-driven agent code generation for home care in Protégé. *Proceedings of 10th International Protégé Conference*.

HORRIDGE, M., JUPP, S., MOULTON, G., RECTOR, A., STEVENS, R. and WROE, C. (2007): A practical guide to building OWL ontologies using Protégé 4 and CO-ODE yools. http://owl.cs.manchester.ac.uk/tutorials/protegeowltutorial/resources/ProtegeOWLTutorialP4_v1_1.pdf. Accessed 21 March 2011.

JENA (2010): Jena – A semantic web framework for Java. http://jena.sourceforge.net/. Accessed 21 March 2011.

KNUBLAUCH, H. (2004): Ontology-driven software development in the context of the semantic web: An example scenario with Protege/OWL. *Proceedings of 1st International Workshop on the Model-Driven Semantic Web*.

KUPFER, A., ECKSTEIN, S., NEUMANN, K. and MATHIAK, B. (2006): A coevolution approach for database schemas and related ontologies. *Proceedings of 19th IEEE Symposium on Computer-Based Medical Systems (CBMS'06)*.

LASHERAS, J., VALENCIA-GARCÍA, R., FERNÁNDEZ-BREIS, J.T. and TOVAL, A. (2009): Modeling reusable security requirements based on an ontology framework. *Journal of Research and Practice in Information Technology* (41)2: 119–133.

LOPEZ-LORCA, A., BEYDOUN, G., STERLING, L. and MILLER, T. (2010): An ontology-mediated validation process of software models. *Proceedings of International Conference on Information Systems Development*.

MENG, W. J., RILLING, J., ZHANG, Y., WITTE, R. and CHARLAND, P. (2006): An ontological software comprehension process model. *Proceedings of 3rd International Workshop on Metamodels, Schemas, Grammars, and Ontologies for Reverse Engineering*.

MUKHOPADHYAY, D., BANIK, A. and MUKHERJEE, S. (2007): A technique for automatic construction of ontology from existing database to facilitate semantic web. *Proceedings of 10th International Conference on Information Technology, (ICIT 2007)*: 246–251.

NOY, N.F. and KLEIN, M. (2004): Ontology evolution: Not the same as schema evolution. *Knowledge and Information Systems* (6)4: 428–440.

ORACLE (2010): The Java database connectivity (JDBC). http://www.oracle.com/technetwork/java/javase/tech/index-jsp-136101.html. Accessed 21 March 2011.

PANIAGUA-MARTÍN, F., GARCÍA-CRESPO, Á., COLOMO-PALACIOS, R. and RUIZ-MEZCUA, B. (2011): SSAAMAR: Semantic annotation architecture for accessible multimedia resources. IEEE Multimedia (18)2: 16–25.

PROTÉGÉ (2011): The Stanford Center for Biomedical Informatics Research at the Stanford University School of Medicine Protégé. http://protege.stanford.edu/. Accessed 21 March 2011.

SADRAEI, E., AURUM, A., BEYDOUN, G. and PAECH, B. (2007): A field study of the requirements engineering practice in Australian software industry. *Requirements Engineering* (12)3: 145–162.

SHADBOLT, N., HALL, W. and BERNERS-LEE, T. (2006): The semantic web revisited. *IEEE Intelligent Systems* (21)3: 96–101.

SHANKS, G., TANSLEY, E. and WEBER, R. (2003): Using ontology to validate conceptual models. *Communications of the ACM* (46)10: 85–89.

STUDER, R., BENJAMINS, V.R. and FENSEL, D. (1998): Knowledge engineering: Principles and methods. *IEEE Transactions on Data and Knowledge Engineering* (25)122: 161–197.

TRAN, N., LOW, G. and BEYDOUN, G. (2006): A methodological framework for ontology centric agent oriented software engineering. *International Journal of Computer Systems Science and Engineering* (21)2: 117–132.

TRINKUNAS, J. and VASILECAS, O. (2007): Building ontologies from relational databases using reverse engineering methods. *Proceedings of International conference on Computer systems and technologies (CompSysTech '07)*.

UPADHYAYA, S.R. and KUMAR, P.S. (2005): ERONTO: a tool for extracting ontologies from extended E/R diagrams. *Proceedings of ACM symposium on Applied computing (SAC 05)*: 666–670.

VOROBIEV, A. and BEKMAMEDOVA, N. (2010): An ontology-driven approach applied to information security. *Journal of Research and Practice in Information Technology* (42)1: 61–76.

W3C (2004a): RDF Primer. http://www.w3.org/TR/rdf-syntax/. Accessed 22 March 2011.

W3C (2004b): RDF Vocabulary Description Language 1.0: RDF Schema. http://www.w3.org/TR/2004/REC-rdf-schema-20040210/. Accessed 22 March 2011.

W3C (2004c): XML Schema Part 0: Primer Second Edition. http://www.w3.org/TR/xmlschema-0/. Accessed 21 March 2011.

W3C (2004d): XML Schema Part 1: Structures Second Edition. http://www.w3.org/TR/xmlschema-1/. Accessed 21 March 2011.

W3C (2004e): XML Schema Part 2: Datatypes Second Edition. http://www.w3.org/TR/xmlschema-2/. Accessed 21 March 2011.

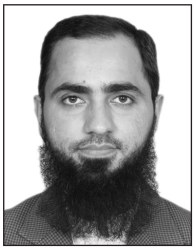
W3C (2009): OWL 2 Web Ontology Language Document Overview. http://www.w3.org/TR/owl2-overview/. Accessed 21 March 2011.

## BIOGRAPHICAL NOTES

*Waqas Ahmed is working as a computer forensic scientist at Punjab Forensic Science Agency, Lahore, Pakistan. He is an MS in Computer Science with specialization in software engineering from COMSATS Institute of Information Technology, Lahore. His research interests include computer forensics, mobile phone forensics, network security, social semantic networks and semantic web. He also enjoys RC- model flying and cricket in his free time.*

Waqas Ahmed

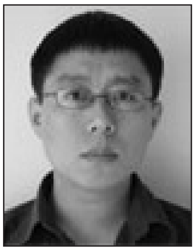
*Muhammad Ahtisham Aslam is working as assistant professor at Information Systems Department, King Abdul Aziz University, Jeddah, Saudi Arabia. He has been working as senior staff researcher at Artificial Intelligence Centre, Malaysian Institute of Microelectronics Systems (MIMOS), Kuala Lumpur, Malaysia. He also has been working as assistant professor at COMSATS Institute of Information Technology, Pakistan. He did his PhD from University of Leipzig, Germany. He has several International and local publications in the area of semantic web and web services. His research interests are semantic web, semantic web services, web 2.0 and social semantic network.*


Muhammad Ahtisham Aslam

*Antonio A. Lopez-Lorca holds a degree in computer science from University of Murcia in Spain. Currently he is a PhD candidate and lecturer at the School of Information Systems and Technology at the University of Wollongong in Australia. In his PhD, funded by the Australian Research Council, he studies the validation of multi agent systems models using ontologies. He is co-author of several papers in international journals and conferences. His research interests include multi agent systems, artificial intelligence, knowledge management, ontology modeling and reasoning and software engineering.*

Antonio A. Lopez-Lorca

*Jun Shen (SnrM'06) was awarded PhD in 2001 at Southeast University, China. He held positions at Swinburne University of Technology in Melbourne and University of South Australia in Adelaide before 2006. He is senior lecturer at University of Wollongong in Wollongong, NSW Australia. He has published more than 60 papers in journals and conferences in computer science and information system area. His expertise is on web services and semantic web. He has been editor, PC chair, guest editor, PC member for numerous journals and conferences published by IEEE, ACM, Elsevier and Springer. He was a chair of Education Chapter of IEEE NSW section since 2007.*


Jun Shen

*Dr Ghassan Beydoun received a degree in computer science and a PhD degree in knowledge systems from the University of New South Wales. He is currently a senior lecturer at the School of Information Systems and Technology at the University of Wollongong and an adjunct senior research fellow at the School of Information Systems, Management and Technology at the University of New South Wales. He has authored more than 90 papers for international journals and conferences. He is currently working on a project sponsored by an Australian Research Council Discovery Grant to investigate the best uses of ontologies in developing methodologies for distributed intelligent systems. His other research interests include multi agent systems applications, ontologies and their applications, and knowledge acquisition.*

Ghassan Beydoun

*Debbie Richards is a professor in the Computing Department at Macquarie University in Sydney. She has been interested in expertise and knowledge management from a theoretical and practical point of view since the early 80s. This was initially inspired by her work in industry with experts from various commercial and retail domains and explored further in her Masters and PhD theses, following completion of a Bachelor of Business. While much of Debbie's research is within the field of artificial intelligence she is keen to develop systems that people are able to use and which make a difference to practice in industry.*

Debbie Richards