MODELING ONTOLOGY VIEWS:
An Abstract View Model for Semantic Web

Rajugan, R.[1], Elizabeth Chang[2], Tharam S. Dillon[1], Ling Feng[3] and Carlo Wouters[4]
[1]eXel Lab, Faculty of IT, University of Technology, Sydney, Australia; [2]School of Information Systems, Curtin University of Technology, Australia; [3]Faculty of Computer Science, University of Twente, The Netherlands; [4]Department of Computer Science & Computer Engineering, La Trobe University, Australia

Abstract: The emergence of Semantic Web (SW) and the related technologies promise to make the web a meaningful experience. However, high level modeling, design and querying techniques proves to be a challenging task for organizations that are hoping to utilize the SW paradigm for their industrial applications. To address one such issue, in this paper, we propose an abstract view model with conceptual extensions for the SW. First we outline the view model, its properties and some modeling issues with the help of an industrial case study example. Then, we provide some discussions on constructing such views (at the conceptual level) using a set of operators. Later we provide a brief discussion on how such this view model can utilized in the MOVE [1] system, to design and construct *materialized* Ontology views to support Ontology extraction.

Key words: Semantic Web (SW), view models, Ontology views, Object-Oriented conceptual models (OOCM), conceptual views, Ontology extraction

1. INTRODUCTION

The emergence of Semantic Web (SW) and the related technologies promise to make the web a meaningful experience. Conversely, success of SW and its applications depends heavily on utilization and interoperability of well formulated Ontology bases in an automated, heterogeneous environment. This creates a need to investigate utilization of (materialized) Ontology views [2] in SW applications, such as; (a) Ontology extraction, (b) Ontology versioning, (c) sub-ontology bases, and (d) SW-wrappers for traditional data sources, in industrial settings. However, unlike

traditional database systems, high level modeling, design and querying techniques still proves to be a challenging task for SW paradigm as, Ontology view definitions and querying have to be done at high-level abstraction [2, 3].

The databases systems (from relational to deductive systems) have matured enough to face growing challenges faced by the organizations (both commercial and governments) and their emerging (and aging) Enterprise Information Systems (EIS). They have well defined basic principles [4] on which they are built upon. Due to this, supporting data intensive technologies, such as transaction processing, business queries, data warehousing, data mining etc. have evolved to a level that can be considered as "matured". Many new and ongoing research directions in data intensive domains still follow the basic principles of databases [5], namely meta-data, schema and instance data. This, in our view is one of the major differences between the database and the SW principles, where meta-data schemas and instance data may overlap. Also, the data extraction process (e.g. queries), in direct contrast to user queries in database systems, is usually automated and involves meta-data extraction as part of the process.

On the other hand, Semantic Web directives are still at its infancy in areas such as data organization, meta-data models and query languages. As a result, in the present stage of SW developments, there are lots of contradictions than agreements in regards to basic concepts and definitions of the SW vocabularies (*see* section 2). Regardless of contradictions, many organizations, both academic and industry are working tirelessly in proposing new methodologies, models and are vigorously formulating standards to streamline the SW paradigm (some consider the present SW phase to be level 2 activities [1]).

On a positive note, there is an exponential growth in new research directions in SW applications. These applications range from SW-enabled traditional enterprise meta-data repositories to time-critical medical information and infectious decease classification databases. For such vast Ontology bases to be successful and to support autonomous computing in a distributed (and heterogeneous) environment, the preliminary design and engineering of such Ontology bases should follow a strict software engineering discipline [6]. Furthermore, supporting technologies for Ontology engineering such as data extraction, integration and organization have be matured to provide adequate modeling and design mechanism to build, implement and maintain successful Ontology bases. For such purpose, Object-Oriented (OO) paradigm seems to be ideal choice as it has been proven in many other complex applications and domains [7, 8].

During mid relational and early Object-Oriented (OO) revolution, during similar phase of the technological development and standardization (level 2), all agreed (both academia and industry) that the data models should be independent of the underlying language semantics and syntaxes and be able to provide needed abstraction and model portability [7, 9]. Today, this notion still holds true for SW paradigm.

To address such an issue, in this paper, we propose and *abstract* view model for modeling and designing views for SW paradigm (SW-view). Such abstract view model is defined using a high-level modeling OO language (such XSemantic net

[10, 11] or OMG's UML [12] or Ontology Web Language (OWL) [13], in contrast to Ontology specific data language) that is capable of modeling Ontologies.

The rest of this paper is organized as follows. In section 2, we briefly describe some of the terminologies used in the context of SW, followed by some of the early work done in view related domain in section 3. Section 4 describes our view model. Section 5 briefly outlines how our view model is utilized in the MOVE [1] system. In section 6 we provide some illustrative examples of our view model concepts that are based on a real-world industrial case study. Section 7 concludes the paper with some discussion on our future research directions.

2. DATABASES, ONTOLOGIES AND VIEWS

Databases and Ontologies serve to structure vast amount of information that is available at given point in time [14]. But in theory, there exists a clear distinction between databases and Ontologies, namely, the clear distinction between the schema and the instances. In databases (relational, OO, active, etc.), schemas are precisely defined in one level of abstraction (usually at the logical or schemata level) and instances are added, edited and/or validated in another layer. Usually views in database systems are defined as part of the external schema. Conversely, Ontology bases tend to have heterogeneous schemas at varying levels of abstraction (logical or instantiated schemas) and instances may co-exist among these schemas to convey information, concepts or relationship between two concepts to the users.

Another intriguing difference between database and Ontology base is that, database trend to follow a well-defined and established standard/(s), while Ontology standards, functionality and definitions trend to differ between implementations and models due to its infancy [2, 15]. For example, in OWL [16] one can create instances as part of the ontology but not in the DOGMA approach [6].

For the purpose of this paper, we need to make a distinction between the concept of abstract view definitions (addressed in this paper) for SW and the view definitions in SW languages such as Resource Description Framework (RDF) [17] and the Ontology Web Language (OWL, previously known as DAML+OIL) [16]. Though expressive, SW related technologies and languages suffer from visual modeling techniques, fixed models/schemas and evolving standards. In contrast, higher-level OO modeling language standards (with added semantics to capture Ontology domain specific constraints) are well-defined, practiced and transparent to any underlying model, language syntax and/or structure [18]. They also can provide well-defined models that can be transferred to the underlying implementation models with ease. Therefore for the purpose of this paper, an abstract view for SW is a view, where its definitions are captured at a higher level of abstraction (namely, conceptual), which turn can be transformed, mapped and/or materialized at any given level of abstraction (logical, instance etc.) in a SW specific language and/or model.

In addition, an abstract view model for SW should be able to deal with not just one but multiple data encoding language standards and schemas (such as XML,

RDF, OWL etc.), as enterprise content may have not one, but multiple data coding standards and ontology bases. Another issue that deserves investigation is the modeling techniques of views for SW. Though expressive, SW related technologies suffer from proven visual modeling techniques [18]. This is because Object-Oriented (OO) modeling languages (such as UML) provide insufficient modeling constructs for utilizing semi-structured (such as XML, RDF, OWL) schema based data descriptions and constraints, while XML/RDF Schema lacks the ability to provide higher levels of abstraction (such as conceptual models) that are easily understood by humans. To address this issue, many researchers have proposed OMG's UML (and OCL) based solutions [2, 15, 18-21], with added extensions to model semi-structured data.

3. RELATED WORK

We can group the existing view models into four categories, namely; (a) classical (or relational) views [4, 22], (b) Object-Oriented (OO) view models [5, 23], (c) semi-structured (namely XML) view models [24-26] and (d) view models for SW. An extensive set of literature can be found in both academic and industry forums in relation to various view related issues such as (i) models, (ii) design, (iii) performance, (iv) automation and (v) turning and refinement, mainly supporting the 2-Es; data Extraction and Elaboration (with and some research directions towards 3-Es, i.e. 2-Es and data Extension). A comprehensive discussion on existing view models can be also found in [26]. Here, we focus only on view models for semi-structured data and SW.

Since the emergence of XML [27], the need for semi-structured data models to be independent of the fixed data models and data access, violates fundamental properties of the classical data models. Many researchers have attempted to solve semi-structured data issues by using graph based [28] and/or semi-structured data models [29, 30]. But, as in the case of relational and OO, the actual view definitions are only available at the lower levels of the implementation and not at the conceptual and/or logical level [26, 31].

One of the early discussion on XML views was by Serge Abiteboul [24] and later more formally by Sophie Cluet et al. [32]. They proposed a declarative notion of XML views. Abiteboul et al. pointed out that, a view for XML, unlike classical views, should do more than just providing different presentation of underlying data [24]. This, he argues, arises mainly due to the nature (semi-structured) and the usage (primarily as common data model for heterogeneous data on the web) of XML. He also argues that, an XML view specification should rely on a data model (like ODMG [33] model) and a query language. In the paper [32], they discuss in detail on how abstract paths/DTDs are mapped to concrete paths/DTDs. These concepts, which are implemented in the Xyleme project [34, 35], provide one of the most comprehensive mechanisms to construct an XML view to-date. The Xyleme project uses an extension of ODMG Object Query Language (OQL) to implement such an XML view. But, in relation to conceptual modeling, these view concepts provide no

support. The view model is derived from the instantiated XML documents (instant level) and is associated with DTD in comparison to flexible XML Schema. Also, the Xyleme view concept is mainly focused on web based XML data.

Another XML view model; the MIX (Mediation of Information using XML) [36] view system, is a by-product of developing web scale mediator systems. The MIX system is based on mediator architecture supporting to provide the user with an integrated view of the underlying heterogeneous information/data sources. The MIX system employs XML as the data exchange and integration medium between mediator components and the XML DTD to provide structural descriptions of the data. Though MIX system provides support for XML views, it is not an XML view by nature. It is a by-product to support data mediation for web-based information systems. Though powerful, the drawback includes no standalone framework to support XML views and non-standard language/(s) used to query/manipulate data.

Another view model for XML, which is based on Object-Relationship-Attribute model for Semi-Structured data (ORA-SS) was proposed by authors in [25]. It is an intuitive data model for XML based on Entity-Relationship (ER) model and the static OO model. An object in ORA-SS is similar to that of an entity in ER (similar to that of an XML element), while a relationship is similar to that of a relationship between two entities in ER. Attributes of ORA-SS describe the objects and relationships. This is one of the first view model that supports some of abstraction above the data language level.

In the work [26, 31], we proposed a layered view model for XML with three levels of abstraction, namely; conceptual, logical and instance levels. In the view model, the view definitions are captured at the conceptual level using a set of conceptual operators [37]. The conceptual view definitions are transformed to logical/schema view definitions (using XML schema definition language) and to document/instance view query expressions (e.g. such as XQuery and or SQL 2003). An added advantage of such view model include; (a) capture conceptual semantics that are easily understood by both human and machines (in contrast to machine-friendly query expressions), (b) view definition are independent of any query language syntax, (c) provide view validation using XML (view) schema and (d) expressive view semantics that support extraction, elaboration or both.

In related work in Semantic Web (SW) [38] paradigm, some view models have been proposed in [3, 39], where the authors present a RDF views with support for RDF [17] schema (using a RDF schema supported query language called RQL). This is one of the early works focused purely on RDF/SW paradigm and has sufficient support for logical modeling of RDF views. The extension of this work (and other related projects) can be found at [40]. RDF is an object-attribute-value triple, where it implies object has an attribute with a value [41]. It only makes intentional semantics and not data modeling semantics. Therefore, unlike generic view models, views for such RDF (both logical and concrete) have no tangible scope outside its domain. In related area of research, the authors of the work proposed a logical view formalism for ontology [1, 15, 42] with limited support for conceptual extensions, where materialized ontology views are derived from conceptual/abstract view extensions.

Another area that is currently under development is the view formalism for SW Meta languages such as OWL. In some SW communities, OWL is considered to be a conceptual modeling language for modeling Ontologies, while some others consider it to be a crossover language with rich conceptual semantics and RDF like schema structures [1]. It is outside the scope of this paper to provide argument for or against OWL being a conceptual modeling language. Here, we only highlight one of view formalism that is under development for OWL, namely views for OWL in the "User Oriented Hybrid Ontology Development Environments" [43] project.

4. OUR ABSTRACT VIEW MODEL FOR SEMANTIC WEB

In this paper, we present an abstract view model with conceptual extensions for the SW (SW-view). Initially such view model was proposed for XML by us in [26, 31], with clear distinction between three levels of abstraction namely; (a) conceptual, (b) logical (or schematic) and (c) document (or instance). Here it is adopted for the SW paradigm.

In work with XML, we provided clear distinction between conceptual, logical and document levels views, as in the case of data engineering, there exists a need to clearly distinguish these levels of abstractions. But in the case of SW domain, though there exists a clear distinction between conceptual and logical models/schemas, the line between the logical (or schema) level and document (or instance) level trends to overlap due to the nature of the data sources (namely Ontology bases), where concepts, relationships and values may present mixed sorts, such as schemas and values [14]. Therefore, in the SW-view model, we provide a clear distinction between conceptual and logical views, but depending on the application, we allow an overlap between logical views and document views. This is one of the main differences between the XML view model and the SW-views.

To our knowledge, other than our work, there exist no research directions that explore the conceptual and logical view model for the Semantic Web (SW) paradigm. This notation of SW-view model has explicit constraints and an extended set of expressive conceptual operators to support Ontology extraction in the MOVE [1, 2, 15] system.

4.1 Conceptual Views

The conceptual views are views that are defined at the conceptual level with conceptual level semantics using higher-level modeling languages such as UML. To understand the SW-view and its application in constructing ontology views, it is imperative to understand its concept and its properties. First, an informal definition of the view concept is given followed by a formal definition that serves the purpose of highlighting the view model properties and the modeling issues associated with such a high-level construct.

Definition 1*:* A **conceptual view** is the one which is defined at the conceptual level with higher level of abstraction and semantics.

One such abstract view model will; (i) provide data abstraction to view data set similar to a class (in OO) does to real-world objects, (ii) enable the software designers (not the programmers) to visualise, construct and validate constructed data sets (views) that are normally left to implementers, (iii) utilise as a tool to communicate better with the domain users and to improve domain user feedback (as users usually used to visualise data as a constructed data sets (views) than a stored/modelled data) and (iv) be utilised by system designers to add additional data semantics at a higher level of abstractions to data intensive domains (such as SW or XML domains), where both data and data semantics are important.

4.2 Conceptual View Properties

To utilize the SW-view model in applications, it is imperative that, one must first understand some of its unique properties and characteristics. In this section, we first provide some of the SW-view formal semantics followed by the derivation of the conceptual view definition. It should be note here that, though there can be more elaborated definitions are possible depending on the application domain, here we provide a simplified generic SW-view definition that can be easily applied to ontology extraction. Following the conceptual view definition are the sections that address some of the unique characteristics of the SW-views, conceptual operators, some modeling issues and the descriptive constraint model.

Conceptual Objects (CO): CO refers to model elements (objects, their properties, constraints and relationships) and their semantic inter-relationships (such as composition, ordering, association, sequence, all etc) captured at the conceptual level, using a well-defined modeling language such as UML, or XSemantic nets [10, 11], OWL or E-ERD [4] etc. A CO can be either of type simple content ($s_{content}$) or complex content ($c_{content}$) depending on its internal structure [10, 41, 44]. For example, CO that uses primitive types (such as integer, character etc) as their internal structure corresponds to $s_{content}$ and CO that uses composite objects represent their internal structure corresponds to $c_{content}$.

Conceptual Schema (CS): We refer conceptual schema as the meta-model (or language) that allow us to define, model and constrain COs. For example, the conceptual schema for a valid UML model is the MOF. Also, the UML meta-model provides the namespace of such schemas.

Like XML/RDF Schema, where the instance will be an XML/RDF document, here, an instance of the conceptual schema will be a *well-defined*, *valid* conceptual model (in this case in UML) or other conceptual schemas (i.e. such as MOF), which can be either visual (such as UML class diagrams) or textual (in the case of UML/XMI models).

Logical/Schema Objects (LO): When CO are transformed or mapped into the logical/schema level (such rules and mapping formalism described in works such as [10, 21, 41, 45, 46]), the resulting objects are called LO. These objects are represented in textual (such as a schema language, OWL) or other formal notations that support schema objects (such as graph).

Postulate 1: A ***context*** (ς) is an item (or collection of items) or a concept that is of interest for the organization as a whole. It is more than a measure [47, 48] and is a

meaningful collection of model elements (classes, attributes, constraints and relationships) at the conceptual level, which can satisfy one or more organizational perspective/(s) in a given domain. Simply said, it is a collection of concepts, attributes and relationships that are of interest in construction of other ontology/(ies).

Postulate 2: A ***perspective*** (∂) is a viewpoint of an item (or a collection of items) that makes sense to one or more stakeholders of the organization or an organizational unit, at given point in time. That is, one viewpoint of a context at a given point in time.

Definition 2: A ***conceptual view*** ($\mathcal{V}_{co}$) [31] is a view, defined over a collection of *valid* model elements, at the conceptual level. That is, it is a perspective for a given context at a given point in time.

Let X be a collection of COs. Let $\Re$ be the rule set, constraints and syntaxes that makes X a ***valid*** collection of CO (according to a meta-modeling language such as MOF or UML or XSemantic nets). Therefore it can be shown that, a *valid conceptual collection set* $\mathcal{X}$ is a function of $\Re$, shown as;

$$\mathcal{X} = \Re(X) \tag{1}$$

We can show that, a *valid* **conceptual view** [14] ($\mathcal{V}_{co}$) of the valid CO set **collection** $\mathcal{X}$ is defined as the **perspective** ∂ constructed over a **context** ς by the conceptual **construct** λ. The resulting conceptual view belongs to the **domain** $\mathcal{D}(\mathcal{V}_{co})$ **,** (where $\mathcal{D}(\mathcal{V}_{co}) = \mathcal{D}_{co}(\varsigma)$) with **schema** $\mathcal{S}_{co}(\mathcal{V}_{co})$**,** (where $\mathcal{S}_{co}(\mathcal{V}_{co}) = \mathcal{S}_{co}(\partial)$). The conceptual view is said to be valid if it is a valid instance of the view schema $\boldsymbol{\mathcal{S}}_{co}(\partial)$. Therefore **conceptual view** $\mathcal{V}_{co}$;

$$\mathcal{V}_{co} = (\varsigma, \partial, \lambda, \mathcal{X}) \tag{2}$$

where; (a) the *view name* of $\mathcal{V}_{co}$ is provided by the perspective ∂, (b) the *domain* and the *namespace* for $\mathcal{V}_{co}$ is provided by the context ς in the *valid* CO collection set of $\mathcal{X}$, (c) the view construction is provide by the conceptual construct λ; i.e. *conceptual operators* that construct the view over a given context, (d) the *valid* collection set $\mathcal{X}$ set provides the data for the view $\mathcal{V}_{co}$ instantiation, (e) the *view schema* $\mathcal{S}_{co}(\mathcal{V}_{co})$ that constrains and validates the view instances of the view $\mathcal{V}_{co}$ and (f) the domain $\mathcal{D}(\mathcal{V}_{co})$ provides the domain for the view $\mathcal{V}_{co}$. Another equivalent form of this definitions can be found in our work in [26].

As we stated earlier, unlike XML-view model, the distinction between conceptual and logical levels are clearly state for SW-views, but not between logical and document views. A detailed discussion of this work can be found in [14].

4.3 Conceptual View Operators

The conceptual constructor is a collection of binary and unary operators, that operates on CO (at the conceptual level) to produce result that is again a valid CO collection. The set of binary and unary operators provided here is a complete or basic set; i.e. other operators, such as division operator [4] and compression (*see* section 6) can be derived from these basic set of operators.

4.3.1 Conceptual Binary Operators

The conceptual set operators are binary operators that take in two operands produces a result set. The following algebraic operators are defined for manipulation of CO collection sets. A CO collection set can be represented in UML, XSemantic nets or other high-level modeling languages.

Let x, y be two valid CO collection sets (operands) that belongs to domains $\mathcal{D}_{co}(x) = dom(x)$ and $\mathcal{D}_{co}(y) = dom(y)$ respectively.

1. *Union Operator*: A Union operator $\bigcup_{(x,y)}$ of operands x, y produces a CO collection set $\mathcal{R}$, such that $\mathcal{R}$ is again a valid CO collection that includes all COs that are either in x or in y or in both x and y with no duplicates. This can be shown as in (3) below, where $dom(\mathcal{R}) = \mathcal{D}_{co}(x) \cup \mathcal{D}_{co}(y)$.

$$\bigcup_{(x,y)} = \mathcal{R} = x \cup y = x \cup x' = y \cup y' \quad \textbf{(3)}$$

2. *Intersection Operator*: An *Intersection* operator $\bigcap_{(x,y)}$ of operands x, y produces a CO collection set $\mathcal{R}$, such that $\mathcal{R}$ is again a valid CO collection that includes all COs that are in both x and y.

$$\bigcap_{(x,y)} = \mathcal{R} = x \cap y \quad \textbf{(4)}$$

where $dom(\mathcal{R}) = \mathcal{D}_{co}(x) \cap \mathcal{D}_{co}(y)$. *Note*: Since both Union and Intersection operators are commutative and associative, they can be applied to n-ary operands.

3. *Difference Operator*: A *Difference* operator $\overline{D_{(x,y)}}$ of operands x, y produces a CO collection set $\mathcal{R}$, such that $\mathcal{R}$ is again a valid CO collection that includes all COs that are in x but not in y.

$$\overline{D_{(x,y)}} = \mathcal{R} = x - y \quad \textbf{(5)}$$

where $dom(\mathcal{R}) = \mathcal{D}_{co}(x)$. Also note; the difference operator is NOT commutative.

4. *Cartesian product Operator*: A *Cartesian product* operator $\times_{(x,y)}$ of operands x, y produces a CO collection set $\mathcal{R}$, such that $\mathcal{R}$ is again a valid CO collection that includes all COs of x and y, combined in combinatorial fashion.

$$\times_{(x,y)} = \mathcal{R} = x \times y \quad \textbf{(6)}$$

where $dom(\mathcal{R}) = \mathcal{D}_{co}(x) \times \mathcal{D}_{co}(y)$

5. *Join Operator*: A *Join* operator can be shown in its general form as;

$$\triangleright\triangleleft_{(x,y)} = \mathcal{R} = x \triangleright\triangleleft_{[j_{condition}]} y$$

where, optional join-condition provides meaningful merger of COs. A join-condition $j_{condition}$ be of the form; (1) simple-condition: where the join-condition $j_{condition}$ is specified using CO simple content $S_{content}$ types, (2) complex-condition: where the join-condition $j_{condition}$ is specified using CO complex content $C_{content}$ types and (3) pattern-condition: where the join-condition $j_{condition}$ is specified using a combination of one or more CO simple and complex content types in a hierarchy with additional constraints, such as ordering etc.

(i) Natural Join

A natural join operator $\triangleright\triangleleft_{(x,y)}$ of operands x, y is a join operator with no join-condition specified, produces a CO collection set $\mathcal{R}$, such that $\mathcal{R}$ it is equivalent to a Cartesian product operator. This can be shown as;

$$\triangleright\triangleleft_{(x,y)} = R = x \triangleright\triangleleft y = \times_{(x,y)} \quad \textbf{(7)}$$

(ii) Conditional Join

A join operator $\triangleright\triangleleft_{(x,y)}$ of operands x, y with explicit join-condition $j_{condition}$ specified produces a CO collection set $\mathcal{R}$, such that $\mathcal{R}$ will have only the combination of CO collection set that satisfies the join condition. The join-condition $j_{condition}$ can only be of type; (1) simple-condition and (2) complex-condition. This join is comparable to the relational operator θ join. This can be shown as;

$$\triangleright\triangleleft_{(x,y)} = \mathcal{R} = x \triangleright\triangleleft_{(j_{condition1}[AND.....])} y \quad \textbf{(8)}$$

(iii) Pattern Join

A join by pattern $\triangleright\triangleleft_{(x,y)}$ is a join by condition operator where the join-condition $j_{condition}$ is of type pattern-condition.

4.3.2 Conceptual Unary Operators

We propose four unary conceptual operators to construct conceptual views without loss of CO semantic that are represented in the model. The four conceptual operators are projection, selection, rename, and restruct(ure).

1. *PROJECT Operator*: Given a valid CO collection set x, and a set of CO (either $s_{content}$ or $c_{content}$ or combination of both $s_{content}$ and $c_{content}$), the project operator $\prod_{(x)}$ will produce a CO collection set $\Re$ where it has only the specified CO set with; (a) persevered node hierarchy, (b) preserved node order and (c) preserved semantic relationships (if any). If need to , the projected CO set (in the case of hierarchical CO/(s) can be specified using the W3C XPath [49] standard.

$$\prod_{(x)} = \Re = \prod_{(CO_1, CO_2, \ldots\ldots)}(x) \quad \textbf{(9)}$$

where the domain of $\Re$ is $dom(\Re) = \bigcup_{k=1}^{m} dom(CO_k)$

2. *SELECT Operator*: Given a valid CO collection set x, the select operator $\sigma_{(x)}$ will produce a CO collection set $\Re$, where it contains one or more matching CO (or collection) that satisfy the select-condition $s_{condition}$. In addition, the select-conditions can be combined using the AND, OR, NOT logical operators.

$$\sigma_{(x)} = \Re = \sigma_{s_{condition}}(x) \quad \textbf{(10)}$$

Again, here, the select-condition $S_{condition}$ be of the form; (1) simple-condition: where the select-condition $S_{condition}$ is specified using CO simple content $S_{content}$ types and the select operator is called *value-based*, (2) complex-condition: where the select-condition $S_{condition}$ is specified using CO complex content $C_{content}$ types and the select operator is called *structure-based* and (3) pattern-condition: where the select-condition $S_{condition}$ is specified using a combination of one or more CO simple and complex content types in a hierarchy with additional constraints, such as ordering etc, where the select operator is called *structure-based*.

3. *RENAME Operator*: Given a valid CO collection set x, and a CO src (with old and new labels $(l^{old}, l^{new}) \in L_{able}$), the rename operator $\rho_{(x)}$ will return x where the label of src is changed. A RENAME operation *cannot*; (a) alter src specific data types and (b) alter src specific contents, values or constraints.

$$\rho_{(x)} = \Re = \rho_{src(l^{old}, l^{new})}(x) \quad \textbf{(11)}$$

4. *RESTRUCT(ure)Operator*: Given a CO collection set x, and a CO, src (with a pair of positions, old and new (pos_1, pos_2)), where the positions can be either absolute or relative (in a CO hierarchy), the restructure operator $\delta_{(x)}$ will return $\Re$, where the position of src (src can be either $s_{content}$ or $c_{content}$) is changed from pos_1 to pos_2.

$$\delta_{(x)} = \Re = \delta_{src(pos_1, pos_2)}(x) \quad \textbf{(12)}$$

But a restructure operation does not allow; (a) deletion of CO/(s) in the hierarchy, (b) alter CO structural relationships, constraints, names or cardinality and (c) alter CO data type or values.

Note: The operators presented above are referred to as extended or non-restive *basic set*, as many secondary (e.g. DIVISION operator) and restrictive operators (*see* section 5) can be derived by combining one or more of these binary and unary operators.

4.4 Modeling Conceptual Views for SW

In this paper, to model conceptual views, we propose OMG's UML (for modelling Ontologies). The only purpose we use this notation is to demonstrate our concepts and applications, and not to emphasis or promote this as the only modeling notation for conceptual views.

UML [12] has established itself as the *defacto* modelling language of choice in OO conceptual modelling paradigm. UML provides a well defined rich collection of tools to visually model a given domain into needed level of abstraction. It can be said that, UML helps to provide a well-defined blue print for a software system that is easily understood both by users and developers alike. UML also provides extensibility to the modelling language in the form of *stereotypes* which we utilise in defining our *conceptual views*. In the case of Ontology engineering, UML provide classes (similar to concepts in ontology), attributes and relationships that are used in defining Ontology models [2] in this paper.

Another reason we adopt UML is that, its models are portable, i.e. many schemata transformation rules and mapping techniques exists for transforming UML models to [20, 21, 41]; (a) XML Schema, (2) Ontology Web Language (OWL), (c) RDF and (d) XMI. Therefore, for the purpose of this paper, UML is visual modelling language of choice for OO conceptual modelling and supports abstraction from classical data models to ontology bases.

4.5 Conceptual View Constraints

In data modeling, specifications often involve constraints. In the case of views, it is usually specified by the data languages (and mostly excluding constraints associated with data semantics) in which they are defined in. For example, in relational model, views are defined using SQL and a limited set of constraints can be defined using SQL[4, 22], namely, (i) presentation specific (such as display headings, column width, pattern order etc), (ii) range and string patterns for aggregate fields, (iii) input formats for updatable views, and (iv) other DBMS specific (such view materialization, table block, size, caching options etc).

In Object-Relational and OO models, views had similar constraints but they are more extensive and explicit due to the data model (yet data language dependent). The OO views are constructed and specified using DBMS specific (such as OQL[33]) and/or external languages (such as C++, Java or O_2C[23]). It is a similar situation in views for semi-structured data paradigm, where rich set of view

constrains are defined using languages such as OQL based LOREL [50, 51]. Today, in the case of Ontology engineering (and in Ontology views), this is still holds true, where constraints are specified using programming modules than at the schemata and/or logical level. In doing so, the constraints are implicit and mostly accessible only at runtime of the system and not at the modeling and/or design time.

But the work by authors of [25] provides some form of higher-level view constraints (under ORA-SS model) for XML views, while the work in [3] provides some form of logical level view constraints to be defined in views for in SW/RDF paradigm. As our conceptual view mechanism is defined at a higher-level of abstraction, we can provide an explicit view constraint specification model, as most high-level OO languages (such as UML, XSemantic nets, E-ER) provide some form of constraint specification.

Here, for our view model, we look into using UML/OCL [52] as our view constraint specification language. Also, our work should not be confused with work such as [53], where authors use OCL to "model" (not to specify) relational views (in contrast to ontology views), which utilizes OCL from a data modeling point of view. In UML, the Object Constraint Language (OCL), which is now a part of the UML 2.0 standard, can support unambiguous constraints specifications for UML models including specification of ontology model elements . In our conceptual view model, we incorporate OCL (in addition to built-in UML constraint features) as our view constraint specification language to explicitly state view constraints. It should be noted that, we do not use OCL to define views, rather state additional constraints using OCL. OCL supports defining *derived* classes [52, 54], which is close to a view concept [53]. Some examples of UML/OCL constraints for conceptual views are given in section 6, below.

5. CONCEPTUAL VIEWS AND THE MOVE [1] SYSTEM

In the sections 4 above, we have shown how conceptual views can be constructed in a given industrial settings. Here, we briefly discuss how such views can be applied in Ontology extraction in the Materialized Ontology View Extractor (MOVE) system [1]. The MOVE system was initially proposed by Wouters et al. [1, 2, 15], for the construction *optimized* materialised Ontology views, with emphasis on automation and quality of the views generated. The MOVE view process includes model and design of conceptual views with the utilization of restricted conceptual operators in deriving materialized Ontology views. Some of the restricted view operators include [2, 14]; (a) synonymous rename (2) selection and (3) compression.

Definition 3: [14] (Informal) A Strict Semantic Web View (or Ontology View) is a materialized SW-view that is derived from an Ontology (called the base ontology). The derivation can consist of any (combination) of the following operations; synonymous rename, selection and compression.

6. AN ILLUSTRATIVE INDUSTRIAL CASE STUDY

The e-Sol Inc. aims to provide logistics, warehouse, and cold storage space for its global customers and collaborative partners. The e-Sol solution includes a standalone and distributed Warehouse Management System (WMS/e-WMS), and a Logistics Management System (LMS/e-LMS) on an integrated e-Business framework called e-Hub [55] for all inter-connected services for customers, business customers, collaborative partner companies, and LWC staff (for e-commerce B2B and B2C). Some real-world applications of such company, its operations and IT infrastructure can be found in [55-57]. Here, use this system as the base to model and integrate (using Ontology views) Ontology bases and various sub-ontology vocabularies used at various customer and collaborative partner locations.

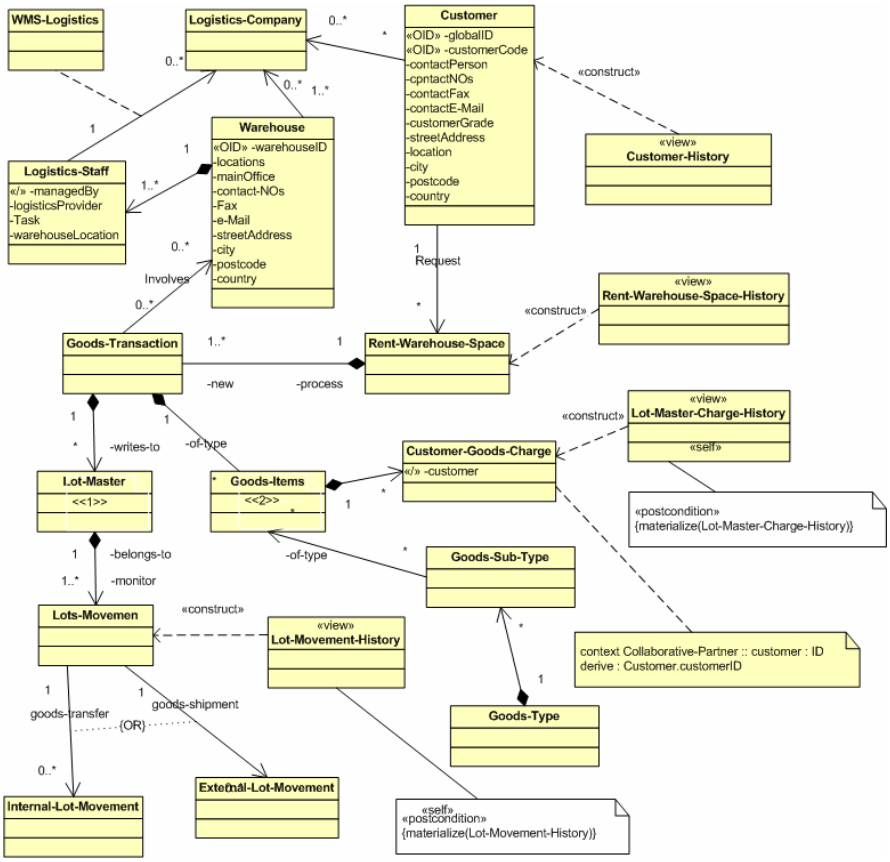

Figure 1. e-Sol example, Core Data Store Model (UML/OCL)

In WMS (Fig. 1-3), customers book/reserve warehouse and cold storage space for their goods. They send in a request to warehouse staff via fax, email, or phone, and depending on warehouse capacity and customers' grade (individual, company or

collaborative partner), they get a booking confirmation and a price quote. In addition, customers can also request additional services such as logistics, packing, packaging etc. When the goods physically arrive at the warehouse, they are stamped, sorted, assigned lots numbers and entered into the warehouse database (in Lots-Master). From that day onwards, customers get regular invoices for payments. In addition, customers can ask the warehouse to handle partial sales of their goods to other warehouse customers (updates Lots-Movement and Goods-Transfer), sales to overseas (handled by LMS) or take out the goods in full or in partial (Lots-Movement). Also customer can check, monitor their lots, buy/sell lots and pay orders via an e-Commerce system called e-WMS. In LMS, customers use/request logistics services (warehouse or third-party logistics providers) provided by the warehouse chains. This service can be regional or global including multi-national shipping companies. Like e-WMS, e-LMS provide customers and warehouses an e-Commerce based system to do business. In e-Hub, all warehouse services are integrated to provide one-stop warehouse services (warehouse, logistics, auction, goods tracking, payment etc) to customers, third-party collaborators and potential customers.

In e-Sol, due to the business process, data semantics have to be in different formats (Ontology bases and vocabularies) to support multiple systems, customers, warehouses and logistics providers. Also, data have to be duplicated at various points in time, in multiple databases, to support collaborative business needs. In addition, since new customers/providers to join the system (or leave), the data formats has to be dynamic and should be efficiently duplicated without loss of semantics. This presents an opportunity to investigate how to integrate and utilize various customers' and collaborative partners' Ontology bases for mutual benefit and SW applications. The following example highlights some example of conceptual views developed for e-Sol. ***Note:*** It should be note that, the examples and the figures given for the e-Sol are demonstration purpose only and do not provide the complete Ontology base model of the system.

Example 1: Context (in Fig. 1-2), "`staff`", "`order`", and "`customer`" can be some of the context examples in the e-Sol system.

Example 2: Conceptual views (Fig. 1), "`Customer-History`", "`Lot-Master-Charge-History`" and "`Rent-Warehouse-Space-History`" are perspectives / views in the context of "`Warehouse-History`" of the e-Sol system.

Example 3: Conceptual view (Fig. 2), "`Collaborative-Partner`" is a perspectives / view in the context of "`Customer`" in e-Sol.

Example 4: Conceptual views, for example, "`processed-order`" and "`overdue-order`" are two contrasting views in the context of "`order`" of the e-Sol system.

Example 5: In Fig. 2, "`Warehouse-Manager`" is a valid XML conceptual view, named in the context of "`Staff`". It is constructed using the conceptual SELECT operator, which can be shown as;

$$\sigma_{warehouse\text{-}Staff.Role=\text{“}manager\text{”}}(Core\text{-}Users).$$

Example 6: Similarly (Fig. 2), "`Site-Manager`" is a perspective/ view in the given context of "`Warehouse-Manager`".

Example 7: Another valid conceptual view "`Lot-Master-Charge-History`" in the given context of "`Warehouse-History`". Here, at the conceptual level, it is stated as a *materialized* conceptual view, implying that it is a persistence view during the life time of the system. This characteristic is also stated in the OCL statement (Fig.1).

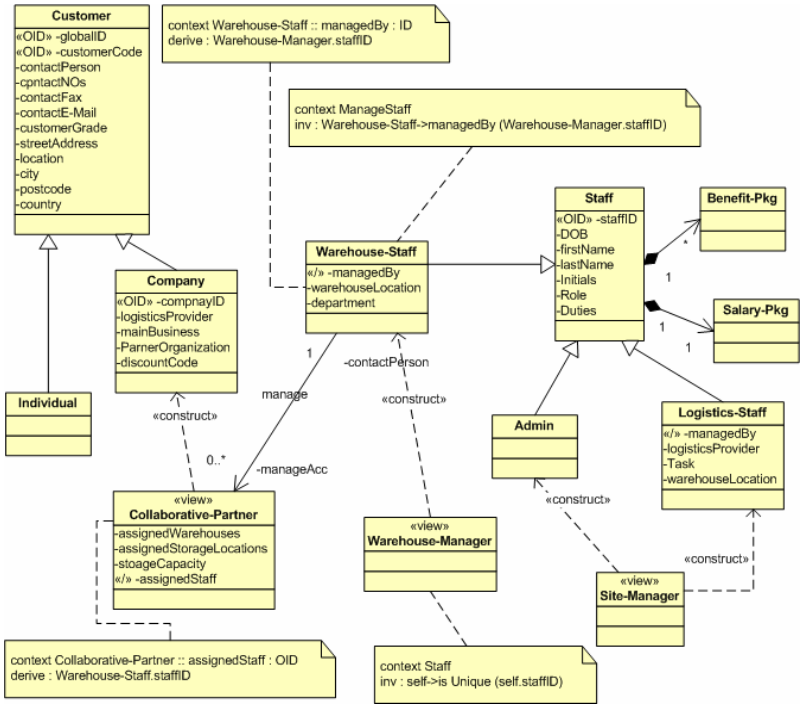

Figure 2. e-Sol, Business User Model (UML/OCL)

Example 8: In the case of conceptual view "`Warehouse-Manager`" (Fig. 2), we indicate the unique `staffID` by the following OCL expression;

```
context Staff
inv : self->isUnique(self.staffID)
```

Example 9: In the case of conceptual view "`Income`" (Fig. 3), the following OCL statements hold true;

```
context Income :: Staff : ID
derive : Staff.staffID

context Income :: benefits : Real
derive : Benefit-Pkg.totalBenefits

context Income :: baseSalary : Real
derive : Salary-Pkg.baseSalary
```

```
context Income :: totalSalary : Real
derive : totalSalary =
    (self.baseSalary - self.tax)
     + benefits
     - self.totalDeductions
```

Example 10: In the case of conceptual views "`Warehouse-Manager`" and "`Warehouse-Staff`", in the context of "`Staff`" (Fig. 2), we indicate the adhesion relationship between them using the following OCL statements given below.

```
context Warehouse-Staff :: managedBy : ID
derive: Warehouse-Manager.staffID

context Warehouse-Manager
inv: self.responsibleFor := Set(Warehouse-Staff.staffID)

context ManageStaff
inv : Warehouse-Staff->managedBy (Warehouse-Manager.staffID)
```

Example 11: In the case of conceptual views "`Lot-Movement`" (Fig. 1), the exclusive disjunction between `Internal-Lot-Movement` (stored goods change owners) and `External-Lot-Movement` (goods shipped outside the warehouse) can be show via the OCL statement "`OR`" between the relationships as shown in Fig. 1.

Example 12: If a new domain requirement exists to add new conceptual view "`Management-Memo`" send to all "`Warehouse-Manager`", we can do that using Cartesian Product conceptual operator, where x = `Warehouse-Manager` and y = `Management-Memo`;

$$\times_{(x,y)} = \mathcal{R} = x \times y$$

Example 13: In the case of conceptual view "`Income`" (Fig. 3), the conceptual construct is a conceptual JOIN operator with join conditions, where x = `Staff`, y = `Salary-Pkg` and z = `Benefit-Pkg`:

$$\mathcal{R} = (x \rightarrow_{(x.staffID=y.staffID)} y) \; AND \; (x \rightarrow_{(x.staffID=z.staffID)} z)$$

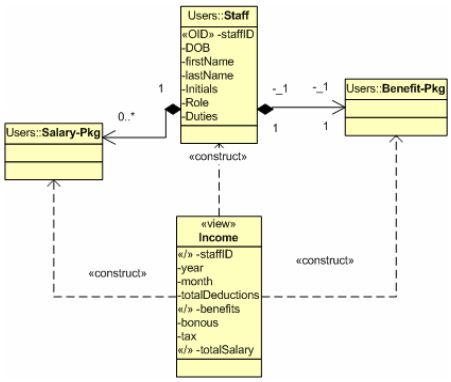

Figure 3. A conceptual view example (Income)

Example 14: A compression of elements indicates that those elements are replaced by a single element in the Ontology view [14]. The element itself can be a new element, but it will not provide additional semantic information (compared to the base ontology). The compression operator constituted of one or more of unary operations combined in sequence.

7. CONCLUSION AND FUTURE WORK

Views have proven to be very useful in databases and here, we presented a descriptive discussion of an abstract view model for SW (SW-view). We first provided formal properties of the SW-view model including a set of binary and unary conceptual operators. Secondly, we provided a brief discussion on issues related to SW-view model, including some modelling issues and the view constraint model. Then we briefly presented how SW-views can be utilized in the MOVE system, followed by some illustrative SW-view based on an industrial case study.

For future work, some further issues deserve investigation. First, the investigation of a formal mapping and transformation approach of the view constraints, and to automate the constraint model transformation between the SW-view model to SW languages, such as RDF and OWL schema constraints. Second, the automation of the mapping process between conceptual operators to various SW (high-level) query language expressions (e.g. RDQL) with emphasis on performance. Third, is the investigation into the dynamic properties of the SW-view model.

8. REFERENCES

[1] C. Wouters, T. S. Dillon, J. W. Rahayu, E. Chang, and R. Meersman, "Ontologies on the MOVE," 9th International Conference on Database Systems for Advanced Applications (DASFAA '04), Jeju Island, Korea, 2004.

[2] C. Wouters, T. S. Dillon, J. W. Rahayu, E. Chang, and R. Meersman, "A Practical Approach to the Derivation of a Materialized Ontology View," in *Web Information Systems, Edited by D. Taniar and J.W. Rahayu*, D. Taniar and W. Rahayu, Eds. USA: Idea Group Publishing, 2004.

[3] R. Volz, D. Oberle, and R. Studer, "Views for light-weight Web ontologies," Proceedings of the ACM Symposium on Applied Computing (SAC '03), USA, 2003.

[4] R. Elmasri and S. Navathe, *Fundamentals of database systems*, 4th ed. New York: Pearson/Addison Wesley, 2004.

[5] W. Kim and W. Kelly, "Chapter 6: On View Support in Object-Oriented Database Systems," in *Modern Database Systems*: Addison-Wesley Publishing Company, 1995, pp. 108-129.

[6] P. Spyns, R. Meersman, and J. Mustafa, "Data Modeling Versus Ontology Engineering," SIGMOD, 2002.

[7] T. S. Dillon and P. L. Tan, *Object-Oriented Conceptual Modeling*: Prentice Hall, Australia, 1993.

[8] I. Graham, A. C. Wills, and A. J. O'Callaghan, *Object-oriented methods : principles & practice*, 3rd ed. Harlow: Addison-Wesley, 2001.

[9] OMG-MDA, "The Architecture of Choice for a Changing World®, MDA Guide Version 1.0.1 (http://www.omg.org/mda/)," OMG, 2003.

[10] L. Feng, E. Chang, and T. S. Dillon, "A Semantic Network-based Design Methodology for XML Documents," *ACM Transactions on Information Systems (TOIS)*, vol. 20, No 4, pp. 390 - 421, 2002.

[11] R.Rajugan, E. Chang, L. Feng, and T. S. Dillon, "Semantic Modelling of e-Solutions Using a View Formalism with Conceptual & Logical Extensions," 3rd International IEEE Conference on Industrial Informatics (INDIN '05), Perth, Australia, 2005.

[12] OMG-UML™, "UML 2.0 Final Adopted Specification (http://www.uml.org/#UML2.0)," 2003.

[13] W3C-OWL, "OWL: Web Ontology Language 1.0 reference (http://www.w3.org/2004/OWL/)," W3C, 2004.

[14] C. Wouters, Rajugan R., T. S. Dillon, and J. W. Rahayu, "Ontology Extraction Using Views for Semantic Web," in *Web Semantics and Ontology*, D. Taniar and W. Rahayu, Eds. USA: Idea Group Publishing, 2005.
[15] C. Wouters, T. S. Dillon, J. W. Rahayu, and E. Chang, "A Practical Walkthrough of the Ontology Derivation Rules," Database and Expert Systems Applications : 13th International Conference (DEXA '02), Aix-en-Provence, France, 2002.
[16] W3C-OWL, "OWL web ontology language 1.0 reference," W3C, 2002.
[17] W3C-RDF, "Resource Description Framework (RDF), (http://www.w3.org/RDF/)," 3 ed: The World Wide Web Consortium (W3C), 2004.
[18] I. Cruz, S. Decker, J. Euzenat, and D. McGuinness, "The emerging semantic web : selected papers from the first Semantic Web Working Symposium," IOS Press,Tokyo : Ohmsha, 2002, pp. 297.
[19] P. Wongthamtham, E. Chang, T. S. Dillon, J. Davis, and N. Jayaratna, "Ontology Based Solution for Software Development," International Conference on Software Engineering and Applications (ICSSEA '03), Paris, France, 2003.
[20] D. Gaševic, D. Djuric, V. Devedzic, and V. Damjanovic, "Approaching OWL and MDA Through Technological Spaces," 3rd Workshop in Software Model Engineering (WiSME 2004), Lisbon, Portugal, 2004.
[21] D. Gaševic, D. Djuric, V. Devedzic, and V. Damjanovic, "Converting UML to OWL Ontologies," Proceedings of the 13 th International World Wide Web Conference, NY, USA, 2004.
[22] C. J. Date, *An introduction to database systems*, 8th ed. New York: Pearson/Addison Wesley, 2003.
[23] S. Abiteboul and A. Bonner, "Objects and Views," ACM SIGMOD Record, Proceedings of the International Conference on Management of Data (ACM SIGMOD '91), 1991.
[24] S. Abiteboul, "On Views and XML," Proceedings of the eighteenth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems (PODS '99), Philadelphia, Pennsylvania, USA, 1999.
[25] Y. B. Chen, T. W. Ling, and M. L. Lee, "Designing Valid XML Views," Proceedings of the 21st International Conference on Conceptual Modeling (ER '02), Tampere, Finland, 2002.
[26] R.Rajugan, E. Chang, T. S. Dillon, and F. Ling, "A Three-Layered XML View Model: A Practical Approach," 24th International Conference on Conceptual Modeling (ER '05), Klagenfurt, Austria, 2005.
[27] W3C-XML, "Extensible Markup Language (XML) 1.0, (http://www.w3.org/XML/)," 3 ed: The World Wide Web Consortium (W3C), 2004.
[28] Y. Zhuge and H. Garcia-Molina, "Graph structured Views and Incremental Maintenance," Proceeding of the 14th IEEE Conference on Data Engineering (ICDE '98), USA, 1998.
[29] S. Abiteboul, R. Goldman, J. McHugh, V. Vassalos, and Y. Zhuge, "Views for Semistructured Data," Workshop on Management of Semistructured Data, USA, 1997.
[30] H. Liefke and S. Davidson, "View Maintenance for Hierarchical Semistructured," Proceedings of the Second International Conference on Data Warehousing and Knowledge Discovery (DaWak '00), London, UK, 2000.
[31] R.Rajugan, E. Chang, T. S. Dillon, and F. Ling, "XML Views: Part 1," 14th International Conference on Database and Expert Systems Applications (DEXA '03), Prague, Czech Republic, 2003.
[32] S. Cluet, P. Veltri, and D. Vodislav, "Views in a Large Scale XML Repository," Proceedings of the 27th VLDB Conference (VLDB '01), Roma, Italy, 2001.
[33] R. G. G. Cattell, D. K. Barry, M. Berler, J. Eastman, D. Jordan, C. Russell, O. Schadow, T. Stanienda, and F. Velez, "The Object Data Standard: ODMG 3.0," Morgan Kaufmann, 2000, pp. 300.
[34] Xyleme, "Xyleme Project (http://www.xyleme.com/)," 2001.
[35] Lucie-Xyleme, "Lucie Xyleme: A dynamic warehouse for XML Data of the Web," *IEEE Data Engineering Bulletin*, vol. 24, No 2, pp. 40-47, 2001.
[36] B. Ludaescher, Y. Papakonstantinou, P. Velikhov, and V. Vianu, "View Definition and DTD Inference for XML," Post-ICDT Workshop on Query Processing for Semistructured Data and Non-Standard Data Formats, 1999.

[37] R.Rajugan, E. Chang, T. S. Dillon, and F. Ling, "A Layered View Model for XML Repositories & XML Data Warehouses," The 5th International Conference on Computer and Information Technology (CIT '05), Shanghai, China, 2005.
[38] W3C-SW, "Semantic Web, (http://www.w3.org/2001/sw/)," W3C, 2005.
[39] R. Volz, D. Oberle, and R. Studer, "Implementing Views for Light-Weight Web Ontologies," Seventh International Database Engineering and Applications Symposium (IDEAS'03), Hong Kong, SAR, 2003.
[40] KAON, "KAON Project (http://kaon.semanticweb.org/Members/rvo/Folder.2002-08-22.1409/Module.2002-08-22.1426/view)," 2004.
[41] L. Feng, E. Chang, and T. S. Dillon, "Schemata Transformation of Object-Oriented Conceptual Models to XML," *International Journal of Computer Systems Science & Engineering*, vol. 18, No. 1, pp. 45-60, 2003.
[42] C. Wouters, T. S. Dillon, J. W. Rahayu, E. Chang, and R. Meersman, "A Practical Approach to the Derivation of a Materialized Ontology View," in *Web Information Systems, Edited by D. Taniar and J.W. Rahayu, Idea Group Publishing, USA, 2004*, D. Taniar and W. Rahayu, Eds. USA: Idea Group Publishing, 2004.
[43] HyOntUse, "User Oriented Hybrid Ontology Development Environments, (http://www.cs.man.ac.uk/mig/projects/current/hyontuse/)," 2003.
[44] L. Feng, T. S. Dillon, H. Weigand, and E. Chang, "An XML-Enabled Association Rule Framework," 14th International Conference on Database and Expert Systems Applications (DEXA '03) 2003, Prague, Czech Republic, 2003.
[45] R. Conrad, D. Scheffner, and J. C. Freytag, "XML conceptual modeling using UML," 19th International Conference on Conceptual Modeling (ER '00), USA, 2000.
[46] D. Gaševic, D. Djuric, V. Devedžic, and V. Damjanovic, "UML for Read-To-Use OWL Ontologies," Proceedings of the IEEE International Conference Intelligent Systems, Vrana, Bulgaria, 2004.
[47] M. Golfarelli, D. Maio, and S. Rizzi, "The Dimensional Fact Model: A Conceptual Model for Data Warehouses," *International Journal of Cooperative Information Systems*, vol. 7, pp. 215-247, 1998.
[48] J. Trujillo, M. Palomar, J. Gomez, and I.-Y. Song, "Designing Data Warehouses with OO Conceptual Models," in *IEEE Computer Society, "Computer"*, 2001, pp. 66-75.
[49] W3C-XPath, "XML Path Language (XPath) Version 1.0," in *XML Path Language*, vol. November 1999: The World Wide Web Consortium (W3C), 1999.
[50] S. Abiteboul, J. Quass, J. McHugh, J. Widom, and J. Wiener, "The Lorel Query Language for Semistructured Data," *International Journal on Digital Libraries*, vol. 1, pp. 68-88, 1997.
[51] R. Goldman, J. McHugh, and J. Widom, "From Semistructured Data to XML: Migrating the Lore Data Model and Query Language," Proceedings of the 2nd International Workshop on the Web and Databases (WebDB '99), Philadelphia, Pennsylvania, 1999.
[52] OMG-OCL, "UML 2.0 OCL Final Adopted specification (http://www.omg.org/cgi-bin/doc?ptc/2003-10-14)," OMG, 2003.
[53] H. Balsters, "Modelling Database Views with Derived Classes in the UML/OCL-framework," The Unified Modeling Language: Modeling Languages and Applications (UML '03), USA, 2003.
[54] J. B. Warmer and A. G. Kleppe, *The object constraint language : getting your models ready for MDA*, 2nd ed. Boston, MA: Addison-Wesley, 2003.
[55] E. Chang, T. S. Dillon, W. Gardner, A. Talevski, Rajugan R., and T. Kapnoullas, "A Virtual Logistics Network and an e-Hub as a Competitive Approach for Small to Medium Size Companies," 2nd International Human.Society@Internet Conference, Seoul, Korea, 2003.
[56] E. Chang, W. Gardner, A. Talevski, E. Gautama, Rajugan R., T. Kapnoullas, and S. Satter, "Virtual Collaborative Logistics and B2B e-Commerce," e-Business Conference, Duxon Wellington, NZ, 2001.
[57] ITEC, "iPower Logistics (http://www.logistics.cbs.curtin.edu.au/)," 2002.