A Layered View Model for XML Repositories & XML Data Warehouses

Rajugan, R. [1], Elizabeth Chang[2], Tharam S. Dillon[1] and Ling Feng[3]
[1]*eXel Lab, Faculty of Information Technology, University of Technology, Sydney, Australia*
E-mail: {rajugan, tharam}@it.uts.edu.au
[2]*School of Information Systems, Curtin University of Technology, Australia*
E-mail: Elizabeth.Chang@cbs.curtin.edu.au
[3]*Faculty of Computer Science, University of Twente, The Netherlands*
E-mail: ling@ewi.utwente.nl

Abstract

The Object-Oriented (OO) conceptual models have the power in describing and modeling real-world data semantics and their inter-relationships in a form that is precise and comprehensible to users. Conversely, XML is fast emerging as the dominant standard for storing, describing and interchanging data among various Enterprises Information Systems (EIS) and databases. With software engineering placing added emphasis on MDA solutions and modeling approaches, it is important to investigate data models for EIS that are abstract enough to be utilized under the MDA paradigm, yet is adoptable in the current data engineering practice. In this paper, we demonstrate a set of language independent conceptual operators for constructing abstract views for XML. The proposed view model has conceptual and logical extensions for supporting abstract views definitions at the conceptual level.

1. Introduction

In software engineering, many methodologies have been proposed to capture real-world problems into manageable segments, which can be communicated, modeled and developed into error-free maintainable software modules/systems. Similarly, in the case of data models, the main objective of conceptual models is to define real-world objects and their relationships in such a way that they represent meaningful units of information with respect to the semantics of the domain in question [1-3]. These models span from early data centered models (e.g. ER/DFD) [4, 5] to the modern Object-Oriented (OO) models [1, 3], where a software system is modeled at varying levels of abstractions.

Since the introduction of eXtensible Markup Language (XML) [6], it is fast emerging as the dominant standard for storing, describing and interchanging data among various Enterprises Information Systems (EIS) and heterogeneous databases. In combination with XML Schema [7], which provides rich facilities for constraining and defining XML content, XML provides the ideal platform and the flexibility for capturing and representing complex EIS data formats.

OO conceptual models have the power in describing and modeling real-world data semantics and their inter-relationships in a form that is precise and comprehensible to users [1, 3]. Yet, OO modeling languages (such as UML) provide insufficient modeling constructs for utilizing XML schema based data descriptions and constraints, while XML Schema lacks the ability to provide higher levels of abstraction (such as conceptual models) that are easily understood by humans. In addition, existing OO paradigm and modeling languages provide minimal or no semantics to capture abstract view formalisms [8-12] (at the conceptual and logical levels) and the existing XML technology standards have no support for concrete view formalisms [9, 13]. To tackle this issue, in our work, we have proposed to extend the concept of semi-structured XML view with conceptual and schemata notions [14], so that it can be applied in modeling and designing complex, large-scale EIS such as XML document warehouses [15, 16], web and e-commerce systems and Semantic Web/Ontology views [17].

In this paper, we present a systematic way of constructing conceptual views [8] using *conceptual operators*. *Note* that the intension of this paper is neither to propose a new view standard for XML nor extensions to XML query languages. Rather, we focus on providing XML view mechanism at the conceptual, schema, and document levels by means of OO conceptual modeling and view definition using conceptual operators.

The rest of this paper is organized as follows. In section 2, we review early work done in view related

Proceedings of the 2005 The Fifth International Conference on Computer and Information Technology (CIT'05)
0-7695-2432-X/05 $20.00 © 2005 IEEE

domains, followed by the introduction of our layered view model in section 3. In section 4, we provide a formal introduction to conceptual operators, followed by a practical walkthrough of the view model using an illustrative case study in section 5. Section 6 discusses the transformation between the conceptual operators and the language specific query expressions. We conclude the paper and outline future work in section 7.

2. Related Work

Here we first briefly look at some of the view models available today and some of the proposals for new view mechanisms supporting semi-structured data, namely XML, RDF and OWL and view constraint specification. A detailed, comprehensive discussion on view models can be found in [17].

We can group the existing view models into three categories, namely; (a) classical (namely relational) views, (b) OO view models, (c) semi-structured (namely XML) view models and (d) views for SW. Early view models includes relational [4, 18] and OO [19, 20]. 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/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).

Since the emergence XML [21], the need for semi-structured data models that have to be independent of the fixed data models and data access, violates fundamental properties of classical data models. Many researchers attempted to solve semi-structured data issues by using graph based [22] and/or semi-structured data models [23, 24]. Again, the actual view definitions are only available at the lower level of the implementation and not at the conceptual and/or logical level [17]. One of the early discussion on XML views was by Serge Abiteboul [13] and later more formally by Sophie Cluet et al. [25]. 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 [13]. These concepts, which are implemented in the Xyleme project [26], provide one of the most comprehensive mechanisms to construct an XML view to-date. But, in relation to conceptual modeling, these view concepts provide no support. Other view models for XML include; (a) the MIX (Mediation of Information using XML) view system [27, 28] and (b) an intuitive view model for XML using Object-Relationship-Attribute model for Semi-Structured data (ORA-SS) [9]. This is one of the first view model that supports some of abstraction above the data language level.

In related work in Semantic Web (SW) [29] paradigm, some work has been done in views for SW [12, 30], where the authors proposed a view formalism for RDF document with support for RDF [31] schema (using a RDF schema supported query language called RDQL [32]). 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 [33]. In related area of research, the authors of the work propose a logical view formalism for ontology [10] 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 [34]for SW Meta languages such as OWL [35], namely views for OWL in the "User Oriented Hybrid Ontology Development Environments" [36] project.

3. The Layered View Model

Our Layered View Model is comprised of three different levels of abstraction, namely, *conceptual level, logical or schema level*, and *document or instance level*. Our XML view study is based on the postulates 1 and 2, about the real world.

Postulate 1: The term ***context*** refers to the domain that interests an organization as a whole. It is more than a measure, and implies a meaningful collection of objects, relationships among these objects, as well as some constraints associated with the objects and their relationships, which are relevant to its applications.

Postulate 2: The term ***view*** refers to a certain perspective of the context that makes sense to one or more stakeholders of the organization or an organization unit at a given point in time.

3.1. Conceptual Level

The top conceptual level describes the structure and semantics of views in a way which is more comprehensible to human users. It hides the details of view implementation and concentrates on describing objects, relationships among the objects, as well as the associated constraints upon the objects and relationships. This level can be modeled using some well-established modeling language such as UML, or our developed XML-specific XSemantic Net [37-39],

etc. Thus, the modeling primitives include object, attribute, relationship, and constraint. The output of this level is a well-defined *valid* conceptual model in UML, XSemantic Net, or even OMG's MOF (Meta-Object-Factory), which can be either visual (such as UML class diagrams) or textual (in the case of XMI models).

***Definition 1*: A conceptual view V^c** is a 4-ary tuple $V^c = (V^c_{name}, V^c_{obj}, V^c_{rel}, V^c_{constraint})$, where V^c_{name} is the name of the XML conceptual view V^c, V^c_{obj} is a set of objects in V^c, V^c_{rel} is a set of object relationships in V^c, and $V^c_{constraint}$ is a set of constraints associated with V^c_{obj} and V^c_{rel} in V^c.

***Definition 2*:** Let $C = (C_{name}, C_{obj}, C_{rel}, C_{constraint})$ denote a context which consists of a context name C_{name}, a set of objects C_{obj}, a set of object relationships C_{rel}, and a set of constraints associated with its objects and relationships $C_{constraint}$. Let $\hat{\lambda}$ be a set of conceptual operators. $V^c = (V^c_{name}, V^c_{obj}, V^c_{rel}, V^c_{constraint})$ is called a *valid conceptual view of the context* $\boldsymbol{C}$, if and only if the following conditions satisfy;

(1) For any object $\forall o \in V^c_{obj}$, there exist objects $\exists o_1, ..., o_n \in C_{obj}$, such that $o = \lambda_{1...}\lambda_m (o_1, ..., o_n)$ where $\lambda_{1...}\lambda_m \in \hat{\lambda}$. That is, o is a newly derived object from existing objects $o_1, ..., o_n$ in the context via a series of conceptual operators $\lambda_{1,...}\lambda_m$ like select, join, etc (*see* section 4).

(2) For any constraint $\forall c \in V^c_{constraint}$, there exists a constraint $\exists c' \in C_{constraint}$ or a new constraint c'' constraints associated with V^c_{obj} or V_{rel}.

(3) For any hierarchical relationship $\forall r^h \in V^c_{rel}$, there *does not exist* a relationship between one or more and V^c_{obj} and C_{obj}.

(4) For any association relationship/dependency relationships $\forall r^a \in V^c_{rel}$, there *may exist a relationship between one or more* V^c_{obj} *and* C_{obj}.

3.2. Logical/Schema Level

The middle scheme level describes the schema of views for the view implementation, using the XML Schema definition language. Views at the conceptual level are mapped into the views at the schema level via the schemata transformation mechanism developed in previous works such as [39-42]. The output of this level will be in either textual (such as XML Schema language) or some visual notations that comply from the schema language (such as graph).

***Definition 3*:** A (logical) *schema view* V^s is a triple $V^s = (V^s_{name}, V^s_{simpleType}, V^s_{complexType}, V^s_{constraint})$, where V^s_{name} is the name of the XML schema view V^s, $V^c_{simpleType}$, $V^s_{complexType}$ are simple and complex type definitions for XML elements/attributes, and $V^s_{constraint}$ is a set of constraints upon the defined XML elements/attributes. Here, $V^c_{simpleType}$, $V^s_{complexType}$, and $V^s_{constraint}$ are expressed in the XML Schema Language, and V^s_{name} is also the name of the resulting XML schema file, i.e., a valid W3C XML document name [21].

***Definition 4*:** Given an conceptual view $V^c = (V^c_{name}, V^c_{obj}, V^c_{rel}, V^c_{constraint})$, $V^s = (V^s_{name}, V^s_{simpleType}, V^s_{complexType}, V^s_{constraint})$ is a **valid schema view of** V^c, if and only if V^s is transformed from V^c by $\aleph^c_s$. That is, $\aleph^c_s : V^c \rightarrow V^s$.

In our previous works such as [14, 38, 43], we have shown how conceptual views (captured either in UML/OCL or XSemantic nets) are mapped to XML Schema. This includes mapping UML (view specific) stereotypes, constraints (both UML and XSemantic nets) and constructional constructs (such as bag, set, list etc.) to XML Schema.

3.3. Document/Instance Level

In the layered view model, the bottom layer; the instance level, implies a fragment of instantiated XML data, which conforms to the corresponding view schema defined at the upper level. Here, the conceptual operators are mapped to query expressions (e.g. XQuery), which are syntax specific.

***Definition 5*:** Given an schema view V^s and a set of XML source documents S, V^i is called a *valid XML instance view* **of** V^s **over** S if and only if V^i is a well-formed XML document, extracted from S by certain query operators in $\hat{\lambda}_{Query}$ and conforming to the XML schema V^s. The following section provide a detailed discussion on conceptual construct $\hat{\lambda}$ and the mapping between $\hat{\lambda}$ and $\hat{\lambda}_{Query}$.

4. Conceptual Operators

Context is presented in UML using modeling primitives like object, attribute, relationship and constraint in this study. To enable the construction of a valid conceptual view from a context, we introduce the notion of *conceptual operator* $\hat{\lambda}$. These conceptual level operators are comparable to relational operator in the relational model, but they operate on conceptual level objects and relationships.

Conceptual operators are grouped into set operators, namely union, difference, intersection, Cartesian product and unary operators namely projection, rename, restructure, selection and joins, and can

facilitate systematic construction of conceptual views from context. These conceptual operators can be easily transformed into query segments, user-defined functions and/or procedures for implementation. By doing so, they help the modeler to capture view construct at the abstract level without knowing or worrying about query/language syntax. The set of binary and unary operators provided here is a complete or basic set; i.e. other operators, such as division operator and compression operator [17] can be derived from these basic set of operators.

4.1. 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 set of objects, ($x, y \in C_{obj}$; called operand$_1$ and operand$_2$ respectively) that belong to the domains $\mathcal{D}(x) = dom(x)$ and $\mathcal{D}(y) = dom(y)$ respectively.

(1) *Union Operator*: A Union operator $\bigcup_{(x,y)}$ of operands x, y will produce a result $\mathcal{R}$, ($\mathcal{R} \in V^c$), such that it includes all C_{obj} that are either in x or in y or in both x and y with no duplicates. This can be shown as;

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

where $dom(\mathcal{R}) = \mathcal{D}(x) \cup \mathcal{D}(y)$

(2) *Intersection Operator*: An *Intersection* operator $\bigcap_{(x,y)}$ of operands x, y will produce a result $\mathcal{R}$, ($\mathcal{R} \in V^c$), such that it includes all C_{obj} that are in both x and y.

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

where $dom(\mathcal{R}) = dom(x \cap y)$.

Note: Since both Union and Intersection operators are *commutative* and *associative*, they can be applied to n-ary operands, i.e. for $\bigcup_{(x_i,x_j)}$ where $1 \le i \le n; 1 \le j \le i$

$$\bigcup_{(x_i,x_j)} = \mathcal{R}$$
$$= x_1 \cup x_2 \cup .....x_i \cup x_j .... \quad \text{and}$$
$$= x_{n-1} \cup x_{n-2} \cup .....x_j \cup x_i ....$$

$$\bigcup_{(x_i,x_j)} = \mathcal{R}$$
$$= (x_1 \cup x_2) \cup x_3 .....x_i \cup x_j ....$$
$$= x_1 \cup (x_2 \cup x_3) .....x_i \cup x_j ....$$

Similarity it can be shown for $\bigcap_{(x_i,x_j)}$ as well, for $1 \le i \le n; 1 \le j \le i$.

(3) *Difference Operator*: A *Difference* operator $\overline{D_{(x,y)}}$ of operands x, y will produce a result $\mathcal{R}$, ($\mathcal{R} \in V^c$), such that it includes all C_{obj} that are in x but not in y.

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

where $dom(\mathcal{R}) = \mathcal{D}(x)$ or $dom(\mathcal{R}) \subseteq \mathcal{D}(x)$

Also note that the difference operator is NOT *commutative.*

(4) *Cartesian product Operator*: A *Cartesian product* operator $\times_{(x,y)}$ of operands x, y will produce a result $\mathcal{R}$, ($\mathcal{R} \in V^c$), such that it includes all C_{obj} of x and y, combined in combinatorial fashion.

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

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

(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 objects in a given context C.

A join-condition $j_{condition}$ be of the form; (1) simple-condition: where the join-condition $j_{condition}$ is specified using C_{obj} simple content $s_{content}$ types, (2) complex-condition: where the join-condition $j_{condition}$ is specified using C_{obj} complex content $c_{content}$ types and (3) pattern-condition: where the join-condition $j_{condition}$ is specified using a combination of one or more C_{obj} simple and complex content types in a hierarchy with additional constraints.

(i) Natural Join: A natural join operator $\triangleright\triangleleft_{(x,y)}$ of operands x, y is a join operator with no join-condition specified will produce a result $\mathcal{R}$, ($\mathcal{R} \in V^c$), such that, $\mathcal{R}$ is equivalent to a Cartesian product operator. This can be shown as;

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

(ii) Conditional Join: A join operator $\triangleright\triangleleft_{(x,y)}$ of operands x, y with explicit join-condition $j_{condition}$ specified will produce a result $\mathcal{R}$, ($\mathcal{R} \in V^c$), such that $\mathcal{R}$ will have *only* the combination of C_{obj} that satisfies the join-condition $j_{condition}$. 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;

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

(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.

4.2. 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 context x, ($x \in C$), the project operator $\prod_{(x)}$ will produce a result $\mathcal{R}$, ($\mathcal{R} \in V^c$), where it has only the specified C_{obj} (and related constraints and/or relationship/(s)) with; (a) persevered node hierarchy, (b) preserved node order and (c) preserved semantic relationships (if any). If need to , C_{obj} (in the case of hierarchical C_{obj}) can be specified using the W3C XPath [44] standard.

$$\prod_{(x)} = \mathcal{R} = \prod_{(C_{obj1,}\, C_{obj2,}\, C_{obj2,}\, ...)}(x)$$

where the domain of $\mathcal{R}$ is;

$$dom(\mathcal{R}) = \bigcup_{k=1}^{m} dom(C_{obj_k})$$

(2) SELECT Operator: Given a valid context x, ($x \in C$), the select operator $\sigma_{(x)}$ will produce a result $\mathcal{R}$, ($\mathcal{R} \in V^c$), where it contains one or more matching C_{obj} (or collections) that satisfy the select-condition $s_{condition}$. In addition, the select-conditions can be combined using the AND, OR, NOT logical operators;

$$\sigma_{(x)} = \mathcal{R} = \sigma_{s_{condition}}(x)$$

Again, here, the select-condition $s_{condition}$ be of the form; (1) simple-condition: where the select-condition $s_{condition}$ is specified using C_{obj} 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 C_{obj} 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 C_{obj} 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 context x, ($x \in C$), and a C_{obj} src (with old and new labels $(l^{old}, l^{new}) \in C_{obj}(label)$), the rename operator $\rho_{(x)}$ will return $\mathcal{R}$ 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.

$$\rho_{(x)} = \mathcal{R} = \rho_{src(l^{old}, l^{new})}(x)$$

(4) RESTRUCT(ure) Operator: Given a valid context x, ($x \in C$), and a C_{obj} src (with a pair of positions, old and new (pos_1, pos_2)), where the positions can be either absolute or relative (in a C_{obj} hierarchy), the restructure operator $\delta_{(x)}$ will return $\mathcal{R}$, where the position of src (src can be either $s_{content}$ or $c_{content}$) is changed from pos_1 to pos_2; $\delta_{(x)} = \mathcal{R} = \delta_{src(pos_1, pos_2)}(x)$.

But a restructure operation does not allow; (a) deletion of C_{obj}/(s) in the hierarchy, (b) alter C_{obj} structural relationships (C_{rel}), constraints ($C_{constraint}$), names or cardinality and (c) alter C_{obj} data type or value.

Note: The operators presented above are referred to as extended or non-restive basic set, as many secondary operators (e.g. DIVISION and restrictive operators [17] for Ontology extraction) can be derived by combining one or more of these binary and unary operators.

5. A Practical Walkthrough

To help illustrate our concepts, in this section, we conduct a real-world case study in a fictitious global

logistic & warehousing company called LWC & e-Solutions Inc., e-Sol in short (Fig. 1). The e-Sol Inc. aims to provide logistics, warehouse, and cold storage space for its global customers and collaborative partners. Some real-world applications of such company, its operations and IT infrastructure can be found in [45-47]. Here, we only use a subsystem of the e-Sol, the Warehouse Management System (WMS).

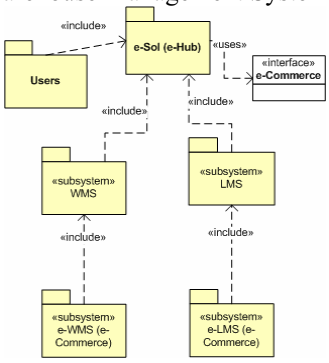

Figure 1: e-Sol. Overview (context diagram)

In WMS (Fig. 2), 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. 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.

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

Example 2: Conceptual views, for example, "`processed-order`" and "`overdue-order`" are two contrasting views in the context of "`order`" of the e-Sol system.

Example 3: In Fig. 2, "`Warehouse-Manager`" is a valid XML conceptual view, named in the context of "`Staff`". Its constructed using the conceptual SELECT (*see* Section 4.2) operator, which can be shown as;

$$\sigma_{\texttt{Warehouse-Staff.Role="manager"}}(\texttt{Core-Users}).$$

Example 4: If a new domain requirement exists to add new conceptual view "`Management-Memo`" send to all "`Warehouse-Manager`", we can do that using

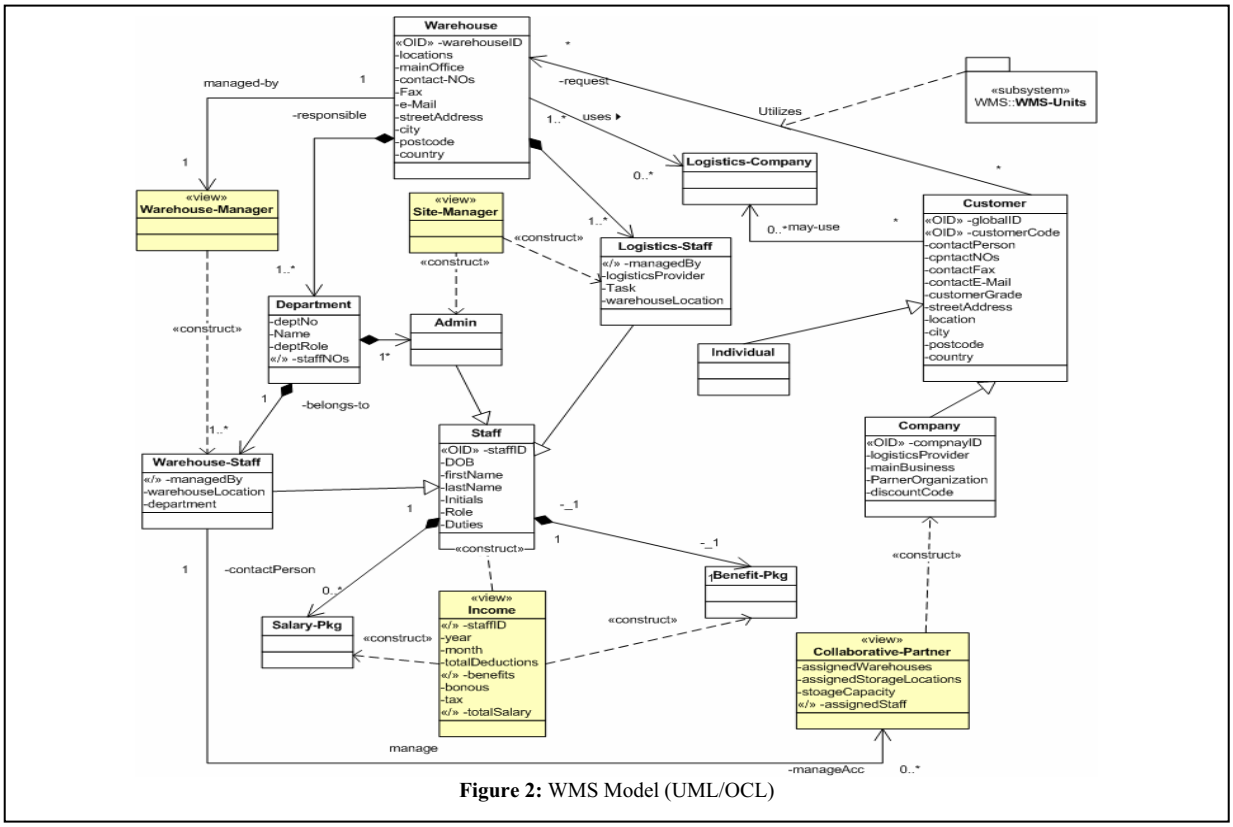

Figure 2: WMS Model (UML/OCL)

Cartesian Product conceptual operator, where x = `Warehouse-Manager` and y = `Management-Memo`;

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

Example 5: In the case of conceptual view "`Income`" (shown in Fig. 2), the conceptual construct is a conceptual JOIN operator with join conditions, where x = `Staff`, y = `Salary-Pkg` and z = `Benefit-Pkg`:

$$\Re = (x \rightarrow_{(x.staffID=y.staffID)} y) \ and$$

$$(x \rightarrow_{(x.staffID=z.staffID)} z)$$

6. Transformation of Conceptual Operators to Query Expressions

An XML instance view is an instantiated imaginary XML document which conforms to the XML schema view defined at the schema level. An instance view can be used for many purposes such as database views, materialised semantic Web-views, etc., and can be generated from simple projection of selected document tags to provide dynamic window into complex heterogenous documents.

XML instance views can be constructed via a native XML query language or some specific algorithms such as Ontology Extraction Methodology (OEM) [10, 17], which can be achieved by the transformation of conceptual operators $\hat{\lambda}$, defined at the conceptual level into a language-specific query fragment $\hat{\lambda}_{Query}$, say FLWOR expressions in XQuery [48], SQL 2003 [49] or OEM etc. In a related work [17], the authors have shown how conceptual operators can be transformed into Ontology extraction algorithms.

Here, we choose XQuery as document view constructor as it is gaining momentum as the language of choice for XML databases and repositories. It is a functional language [50] and its functionalities are comparable to early/mid SQL standards (in the relational model). In addition it is well-suited for our purpose better than other XML query languages (such as XPath or XSLT) as; (a) XQuery is easy to read and write in comparison XPath and XSLT, (b) in direct contrast to XQuery's powerful For-Let-Where-Order-Return (FLWOR) expression, XPath/XSLT are purely presentation oriented, (c) XQuery provides User-Defined Functions (UDF) and variable binding and (d) XQuery support XML Schema.

Example 6: As described in Examples 3 and 8, the conceptual view operators of the view "`Warehouse-Manager`" can be mapped to the document view construct (XQuery expression) as shown below in the code segment.

```
for   $role in document ("staff.xml")//Role
where        $role = "Manager"
return
   <Warehouse-Manager> {$role}
   </Warehouse-Manager>
```

Example 7: As described in Example 4, we can show the Cartesian product conceptual operator can be mapped to the following XQuery expression, at the document level as;

```
For $a in document
        ("Warehouse-Manager.xml")//Role,
   $b in document
        ("Management-Memo.xml")//Message
return
   <send-memo> <W-Mgr>{$a}</W-Mgr>
   <Mgr-Memo>{$b}</Mgr-Memo></send-memo>
```

7. Conclusion and Future Work

In this paper, we presented a view formalism with conceptual and schema extensions with illustrated case study examples. We also presented a detailed discussion on mapping conceptual operators to query language specific expressions (namely XQuery) at the document level.

For future work, some further issues deserve investigation. First, the investigation of a formal transformation methodology between conceptual operators and the query language expressions. Second, the automation of the mapping process between conceptual operators to various query language expressions (mainly XQuery). Third, is the investigation into dynamic perspectives of the conceptual view formalism that can be applied to Semantic Web and web service domains.

8. References

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

[2] P. Coad and E. Yourdon, *Object-oriented analysis*, 2nd ed. London: Prentice Hall, 1991.

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

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

[5] P. Rob and C. Coronel, *Database systems : design, implementation, and management*, 6th ed. Boston: Course Technology, 2004.

[6] W3C-XML, "Extensible Markup Language (XML) 1.0, (http://www.w3.org/XML/)," 3 ed: The World Wide Web Consortium (W3C), 2004.

[7] W3C-XSD, "XML Schema," vol. 2004, 2 ed: W3C, 2004.

[8] R.Rajugan, E. Chang, T. S. Dillon, and F. Ling, "XML Views: Part 1," 14th Int. Conf. on Database and Expert Systems Applications (DEXA '03), Prague, Czech Republic, 2003.

[9] Y. B. Chen, T. W. Ling, and M. L. Lee, "Designing Valid XML Views," Proc. of the 21st Int. Conf. on Conceptual Modeling (ER '02), Tampere, Finland, 2002.

[10] 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.
[11] E. J. Chang, "Object Oriented User Interface Design and Usability Evaluation," in *Department of Computer Science & Computer Engineering*: La Trobe University, Melbourne, Australia, 1996.
[12] R. Volz, D. Oberle, and R. Studer, "Views for light-weight Web ontologies," Proc. of the ACM Symposium on Applied Computing (SAC '03), USA, 2003.
[13] S. Abiteboul, "On Views and XML," Proc. of the eighteenth ACM PODS '99, USA, 1999.
[14] R.Rajugan, E. Chang, T. S. Dillon, and F. Ling, "A Three-Layered XML View Model: A Practical Approach," 24th Int. Conf. on Conceptual Modeling (ER '05), Austria, 2005.
[15] V. Nassis, R.Rajugan, T. S. Dillon, and W. Rahayu, "Conceptual and Systematic Design Approach for XML Document Warehouses," *Int. Journal of Data Warehousing and Mining*, vol. 1, No 3, 2005.
[16] V. Nassis, R.Rajugan, T. S. Dillon, and W. Rahayu, "XML Document Warehouse Design," 6th Int. Conf. on Data Warehousing and Knowledge Discovery (DaWaK '04), Zaragoza, Spain, 2004.
[17] 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.
[18] C. J. Date, *An introduction to database systems*, 8th ed. New York: Pearson/Addison Wesley, 2003.
[19] 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.
[20] S. Abiteboul and A. Bonner, "Objects and Views," ACM SIGMOD Record, Proc. of the Int. Conf. on Management of Data (ACM SIGMOD '91), 1991.
[21] W3C-XML, "Extensible Markup Language (XML) 1.0, (http://www.w3.org/XML/)," 3 ed: The World Wide Web Consortium (W3C), 2004.
[22] Y. Zhuge and H. Garcia-Molina, "Graph structured Views and Incremental Maintenance," Proceeding of the 14th IEEE Conf. on Data Engineering (ICDE '98), USA, 1998.
[23] S. Abiteboul, et al., "Views for Semistructured Data," Workshop on Management of Semistructured Data, USA, 1997.
[24] H. Liefke and S. Davidson, "View Maintenance for Hierarchical Semistructured," Proc. of the Second Int. Conf. on Data Warehousing and Knowledge Discovery (DaWak '00), London, UK, 2000.
[25] S. Cluet, P. Veltri, and D. Vodislav, "Views in a Large Scale XML Repository," Proc. of the 27th VLDB Conf. (VLDB '01), Roma, Italy, 2001.
[26] Lucie-Xyleme, "Xyleme: A Dynamic Warehouse for XML Data of the Web," Int. Database Engineering & Applications Symposium (IDEAS '01), Grenoble, France, 2001.
[27] B. Ludaescher, et al., "View Definition and DTD Inference for XML," Post-ICDT Workshop on Query Processing for Semistructured Data and Non-Standard Data Formats, 1999.
[28] B. Ludascher, Y. Papakonstantinou, and P. Velikhov, "Navigation-Driven Evaluation of Virtual Mediated Views," Extending DataBase Technology (EDBT '00), 2000.
[29] W3C-SW, "Semantic Web, (http://www.w3.org/2001/sw/)," W3C, 2005.
[30] R. Volz, D. Oberle, and R. Studer, "Implementing Views for Light-Weight Web Ontologies," Seventh Int. Database Engineering and Applications Symposium (IDEAS'03), Hong Kong, SAR, 2003.
[31] W3C-RDF, "Resource Description Framework (RDF), (http://www.w3.org/RDF/)," 3 ed: The World Wide Web Consortium (W3C), 2004.
[32] W3C-RDQL, "RDQL - A Query Language for RDF, (http://www.w3.org/Submission/2004/SUBM-RDQL-20040109/)," W3C, 2004.
[33] KAON, "KAON Project (http://kaon.semanticweb.org/Members/rvo/Folder.2002-08-22.1409/Module.2002-08-22.1426/view)," 2004.
[34] D. Gaševic, et al., "Approaching OWL and MDA Through Technological Spaces," 3rd Workshop in Software Model Engineering (WiSME 2004), Lisbon, Portugal, 2004.
[35] W3C-OWL, "OWL: Web Ontology Language 1.0 reference (http://www.w3.org/2004/OWL/)," W3C, 2004.
[36] HyOntUse, "User Oriented Hybrid Ontology Development Environments, (http://www.cs.man.ac.uk/mig/projects/current/hyontuse/)," 2003.
[37] R.Rajugan, E. Chang, L. Feng, and T. S. Dillon, "Semantic Modelling of e-Solutions Using a View Formalism with Conceptual & Logical Extensions," 3rd Int. IEEE Conf. on Industrial Informatics (INDIN '05), Perth, Australia, 2005.
[38] R.Rajugan, E. Chang, L. Feng, and T. S. Dillon, "XML Views, Part II: Modeling Conceptual Views Using XSemantic Nets," Workshop & Special Session on Industrial Informatics, The 30th Annual Conf. of the IEEE Industrial Electronics Society, IECON '04, S.Korea, 2004.
[39] 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.
[40] L. Feng, E. Chang, and T. S. Dillon, "Schemata Transformation of Object-Oriented Conceptual Models to XML," *Int. Journal of Computer Systems Science & Engineering*, vol. 18, No. 1, pp. 45-60, 2003.
[41] R. Xiaou, T. S. Dillon, E. Chang, and L. Feng, "Modeling and Transformation of Object-Oriented Conceptual Models into XML Schema," 12th Int. Conf. on Database and Expert Systems Applications (DEXA '01) 2001, 2001.
[42] R. Xiaou, T. S. Dillon, E. Chang, and L. Feng, "Mapping Object Relationships into XML Schema," Proc. of OOPSLA Workshop on Objects, XML and Databases, 2001.
[43] R.Rajugan, E. Chang, T. S. Dillon, and L. Feng, "XML Views, Part III: Modeling XML Conceptual Views Using UML," 7th Int. Conf. on Enterprise Information Systems (ICEIS '05), Miami, USA, 2005.
[44] W3C-XPath, "XML Path Language (XPath) Version 1.0," in *XML Path Language*, vol. November 1999: The World Wide Web Consortium (W3C), 1999.
[45] E. Chang, et al., "A Virtual Logistics Network and an e-Hub as a Competitive Approach for Small to Medium Size Companies," 2nd Int. Human.Society@Internet Conf., Seoul, Korea, 2003.
[46] E. Chang, et al., "Virtual Collaborative Logistics and B2B e-Commerce," e-Business Conf., Duxon Wellington, NZ, 2001.
[47] ITEC, "iPower Logistics (http://www.logistics.cbs.curtin.edu.au/)," 2002.
[48] W3C-XQuery, "XQuery 1.0: An XML Query Language," in *XML Query Language (XQuery)*: The World Wide Web Consortium (W3C), 2004.
[49] ANSI and ISO, "ANSI - SQL 2003," ANSI / ISO 2003.
[50] D. D. Chamberlin and H. Katz, *XQuery from the experts : a guide to the W3C XML query language*. Boston: Addison-Wesley, 2003.