Visual Modeling of Behavioral Properties in the LVM for XML Using XSemantic Nets

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

Abstract

Due to the increasing dependency on self-describing, schema-based, semi-structured data (e.g. XML), there exists a need to model, design and manipulate semi-structured data and the associated semantics at a higher level of abstraction than at the instance or document level. In this paper, we extend our research and propose to visually model (at the conceptual level) and transform dynamic properties of views in the Layered View Model (LVM) using the eXtensible Semantic (XSemantic) net notation. First, we present the modeling notation and then discuss the declarative transformation to map the dynamic XML view properties to XML query expressions, namely XQuery.

1. Introduction

Since the early software models, abstraction and conceptual semantics have proven their importance in software engineering methodologies. For example, Object-Oriented (OO) conceptual models offer 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 becoming the dominant standard for storing, describing and interchanging data among various Enterprises Information Systems (EIS) and databases. With the increased reliance on such self-describing, schema-based, semi-structured data languages, there is a need to model, design, and manipulate these data (namely XML) and the associated semantics at a higher level of abstraction, than at the instance level. However, in many real-world scenarios, existing OO conceptual modeling notations and languages provide insufficient modeling constructs for utilizing XML schema like data descriptions and constraints, while most semi-structured schema languages lack the ability to provide higher levels of abstraction (such as conceptual models) that are easily understood by humans [1-3]. To this end, it is interesting to investigate conceptual and schemata extensions as a means of providing higher level semantics in XML-related data modeling. In this context, many traditional database concepts and techniques (such as querying, DBMS, views, etc.) have been or are in the process of being transformed to support XML. For example, works [4, 5] are some of the good examples in this direction.

Similarly, in our work, we have proposed a Layered View Model (LVM) for XML with the notion of conceptual and schemata extension [3, 6]. There, we presented in a systematic way with *formal* semantics for: (a) a view formalism for XML with three levels of abstractions, namely, conceptual, schema, and instance; (b) detailed OO modeling primitives of static properties such as domains, attributes, constraints [3] and semantic relationships using two OO modeling languages and (c) the transformation methodology of such static properties between varying levels of abstraction. Since the OO models are capable of describing both static and dynamic properties of a domain in question, in this paper, we present a modeling notion to capture dynamic XML view properties in the LVM. It should be noted that the intention 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.

10th International Database Engineering and Applications Symposium (IDEAS'06)
0-7695-2577-6/06 $20.00 © 2006 IEEE

The motivation for the research presented in this paper includes: (i) Develop an approach that permits separation of implementation aspects and conceptual aspects of views so that there is a clear separation of concerns, thereby allowing analysis and design of views to be separated from their implementation, (ii) Define representations to express these views in a conceptual model as first-class citizens, (iii) Define a mechanism to permit construction of views at such a higher level of abstraction than being tied to a specific data manipulation language, (iv) To define a view development methodology for XML views that utilizes conceptual semantics and carries out schemata transformation to an XML schema for the views as well as transformation of the view construction to an appropriate XML query language, such as XQuery [7].

The rest of this paper is organized as follows. In section 2 we present some of the early work done in relation to XML views, followed by section 3, which describes our view model, view constructs and the view methods in detail. Section 4 describes a real-world case study example that is used to illustrate our concepts presented in this paper. This is followed by section 5 that provides a detailed discussion on modeling dynamic view properties in the LVM using the eXtensible Semantic nets notation. In section 6, a detailed discussion on the declarative transformation of the LVM view dynamic properties to query expressions is given followed by section 7 that concludes this paper with some discussion on our future research directions.

2. Related Work

We can group the existing view mechanisms into four categories, namely; (a) classical (or relational) views [8, 9], (b) views for Object-Oriented (OO) paradigm, (c) semi-structured (namely XML) view mechanisms and (d) view models for the Semantic Web [10] (SW). A comprehensive discussion on these view mechanisms can be found in our work [3, 11]. Here, we focus only on the view mechanism for XML.

One of the early discussion on XML views was by Serge Abiteboul [12] and later more formally by Sophie Cluet et al. [13] and Aguilera et.al [14]. 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 [12]. 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 [15] model) and a query language. In the paper [13], they discuss in detail on how abstract paths/DTDs are mapped to concrete paths/DTDs. These concepts, which are implemented in the Xyleme project [16, 17], 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) [18] 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 the construction of XML views, but it does not provide a mechanism for explicit XML view specification. 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. Also the view formalism is not independent of the MIX architecture.

One of the first XML view mechanism to utilize visual representation is based on Object-Relationship-Attribute model for Semi-Structured data (ORA-SS) and was proposed by authors in [19]. 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 models that support some of abstraction above the data language level.

It should be note that, of all the view models discussed above, provides discussion only on static properties of views and provides no mechanism for capturing and/or modeling dynamic view properties. In a related work in Active XML [4, 20, 21], authors have provided an extensive support for dynamic properties for views in Active XML views, but those views are

based on active rules rather than data or document centric view specification and/or definitions.

3. Our Work: The Layered View Model (LVM) for XML

In our work with views for XML, we proposed a Layered View Model (LVM) for XML [6, 11] that is comprised of three different levels of abstraction, namely, *conceptual level, logical(or schema) level*, and *document (or instance) level*.

The top conceptual level describes the structure and semantics of views in a way that 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 modelled using some well-established modelling language such as UML/OCL [22], or our own XML-specific XSemantic net [1, 3], etc. The output of this level is a well-defined *valid* conceptual model in UML or XSemantic nets which can be either visual (such as UML class diagrams) or textual (in the case of XMI models). Also, at this level the views are referred to as *conceptual views*.

The middle level of the view model is the schema (or logical) level and describes the schema of views using the XML Schema (XSD) [23] definition language. Views at the conceptual level (conceptual views) are mapped into the views at the schema level (referred to as *logical views*) via the extended schemata transformation methodology described in the previous works such as [1, 2]. The output of this level will be in either textual (XSD) or some visual (graph) notations that comply from the schema language. In our previous works such as [3, 6, 11, 24] we have shown how conceptual views are mapped to XSD. This includes mapping UML (view specific) stereotypes, constraints (both UML and XSemantic nets) and constructional constructs (such as bag, set, list etc.) to XSD.

The third level is the document or 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* [3] (and other view dynamic properties) are mapped to language specific query expressions (e.g. XQuery, SQL '03 [25]), which are syntax specific. Formal semantics of the LVM for XML can be found in our work [3]. At this level views are referred to as *document views*.

There are two types of dynamic properties we address in our LVM, namely; (a) view constructs: These are the sequence of one ore more conceptual operators that constructs the views and (b) internal view methods (weak encapsulation) such as generic and update (or user) defined functions. In this paper we address only the view constructs, the generic methods and their declarative transformation to query expression.

3.1. Conceptual Operators

Conceptual operators are operators that operate on conceptual artefacts. They are grouped into set operators, namely; union, difference, intersection, Cartesian product, join and unary operators namely; projection, rename, restructure and selection. These conceptual operators can facilitate systematic construction of conceptual views (at the conceptual level) from a context [6] and can be easily transformed into query expressions, user-defined functions and/or procedures for implementation. By doing so, they help the modeler to capture view constructs at the abstract level without knowing or worrying about query language syntax. The set of binary and unary operators provided here (except intersection and join) are a *complete* or *primitive* set; i.e. other operators, such as division, join, intersection and compression operators can be derived from these complete set of operators. A detailed description and formal semantics of these conceptual operators are presented in [3].

3.2. Internal View Methods and Update Functions

Since the introduction of views in relational DBMS in early '80s, there have been constant discussions and research directions on data manipulation in views (including view updates) to date; view updatability is well-studied and implemented for relational environments in the context of materialized views and data warehouses. But, the concept of data manipulation in views, to date, has very few approved standards in both relational and OO models. Also, this is still a vendor/platform specific task (e.g. Oracle™ DBMS, IBM™ DB2, MySQL™ or O_2 OODBMS) and lacks consistency.

Our intention here is neither to address view updatability issues in the LVM nor to put forward a proposal for view updatability issues using XQuery. Our focus here is to enable conceptual modelling of a minimal set of dynamic view properties (e.g. generic methods) and the corresponding transformation of such

properties to document view query expressions. This, we argue, would provide some degree of *accessibility* and the *encapsulation* concept to views in the LVM. Our aim is to provide both static and dynamic modelling facility for the views in the LVM, at a higher-level of abstraction and the automated transformation its properties to view schema and query expressions.

However, it should be noted here that, in practice, view update issues are strongly coupled with the underlying (XML) database management system and the query language (i.e. supported operators, storage model etc.) and many restrictions and constraints may result from this. For the purpose of this research, we discuss only generic methods and use the newly proposed W3C XQuery update facility [26], which is a W3C working draft. Thus, given below are a set of *declarative* conditions that have to be met in order to perform querying and data accessibility in the LVM views for XML.

There are three kinds of views in the LVM based on their construction operator and the stored document/(s) type [11]. A summery of permitted generic operations for view types in the LVM is given in Table 1.

Table 1: Summary of the generic operations permitted in the LVM

View Types / Operation (or method)	Derived Imaginary Document (DID)	Constructed Imaginary Document (CID)	Triggered Imaginary Document (TID)
Get (or retrieve) methods	yes	yes	yes
Set (or insert) methods	conditional	conditional	no
Update methods	yes	conditional	no
Delete methods	conditional	conditional	no

4. Case Study Example

To demonstrate our concepts and formalisms presented in this paper, we use a real-world case study called e-Sol. The e-Sol 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 [27] 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 [27-29].

In WMS, 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 customers 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 have to be in different formats 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 use our XML conceptual, schema and instance views to design e-Sol at a higher level of abstractions to support changing business, environments, and data formats.

5. Modelling Views in the LVM Using XSemantic Nets

The XSemantic net modelling notion is an intuitive approach to conceptually model XML domains. The modelling efficiency and flexibility come from its structural similarity to an XML document structure and the ability to capture all the static properties of OO concept; objects, relationships (hierarchical and non-

hierarchical) and dependencies to name a few. Here, the concept of nodes and associated constrains are similar or more explicit than the notion of classes in OO models. Also, due to structural similarity, the transformation between XSemantic net and XML is single levelled (i.e. one step) and automatic [1]. The proposed methodology is comprised of three design levels: (i) semantic level, (ii) schema level, and (iii) instance level. The aim is to enforce conceptual modelling power to XML (and views) in order to narrow the gap between real-world objects and XML document structures. The XSemantic net notion used here is given in Fig. 1.

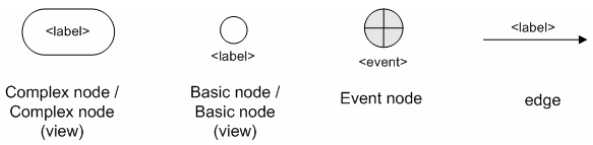

Figure 1: XSemantic net notation

The first level of the XSemantic net design methodology corresponds to the Object-Oriented (OO) conceptual level and is composed of two models, namely, the *domain* and the *view* models. This level is based on an (extended) modified semantic network [1], that provides semantic modelling of XML domains through five major components, namely:

(i) a set of *nodes* representing real-world *objects* and *view* objects,

(ii) a set of *directed edges* representing *semantic relationships* between these objects,

(iii) a set of *labels* denoting different types of semantic relationships, including; (a) **a**ggregation, (b) **g**eneralization, (c) a**s**sociation, (d) of-**p**roperty, (e) **v**iew and (f) **op**erator.

(iv) a set of *constraints*, such as; (a) defined over a node (e.g. uniqueness, referential integrity, etc.), (b) defined over an edge (e.g. cardinality, adhesion, etc.), (c) defined over a set of edges (e.g. exclusive disjunction, etc.)

(v) a set of *conceptual operators* to systemically construct conceptual views such as; (a) unary operators and (b) binary operators.

(vi) a set of *event nodes* representing *generic* and *user defined* methods (or triggers) in the *view* objects.

The second level of the proposed methodology is concerned with detailed XML schema design for both domain and view objects defined at the semantic level, including *element/attribute declarations* and *simple/complex type definitions*. The mapping between these two design levels are extension of the schemata transformation proposal stated in [1] and proposed to transform the semantic models into the XML Schema, based on which XML documents can be systematically created, managed, and validated.

The third level of the design methodology is concerned with a detailed query design for the views, defined at the semantic level including query language specific expressions and syntax declarations. The mapping between semantic level conceptual operators and the query language specific expressions is proposed to transform valid conceptual operators into executable native XML query expressions, such as XQuery FLOWR expressions or SQL 2003/SQLX statements. The resulting query expressions/statements are able to construct imaginary XML documents that can be validated against the XML (view) schemas generated at the schema level of the design methodology. At this level, it is also proposed to transform *generic* and *user* defined methods to query expressions in the form of triggers, user defined functions (UDF) and external procedure calls.

Example 1: The relationship between conceptual view `Logistics-Staff`, `Admin` and `Site-Manager` is generalization/specialization, as shown in Fig. 2.

***Example 2*:** The relationship between conceptual views `Site-Manager` and `Warehouse-Manager` is association, as shown in Fig. 2. `Site-Manager` *manages* `Warehouse-Manager`.

***Example 3*:** The relationship between conceptual view `Warehouse-Staff` and `Warehouse-Manager` is generalization/specialization, as shown in Fig. 2.

***Example 4*:** The relationship between conceptual views `Warehouse-Manager` and `Collaborative-Partner` is association, as shown in Fig. 2. `Warehouse-Manager` *deals-with* `Collaborative-Partner`.

***Example 5*:** In the case of the conceptual view `Income` (shown in Fig. 3), the conceptual construct is a conceptual JOIN operator with join conditions, where `x = Staff`, `y = Salary-Pkg` and `z = Benefit-Pkg`:

$$\triangleright\triangleleft_{(x,y,z)} = (x \rightarrow_{(x.staffID=y.staffID)} y)\ AND\ (x \rightarrow_{(x.staffID=z.staffID)} z)$$

***Example 6*:** There exists an object-attribute relationship (i.e. of-property) `staffID`, `baseSalary`, `totalBenefits` etc. of the conceptual view `Income` as shown in Fig. 3.

***Example 7*:** For example (Fig. 3), we have shown how a simple join conceptual operator is represented

using an event node. In Fig. 4, the same example (`Income`) is shown with some generic methods.

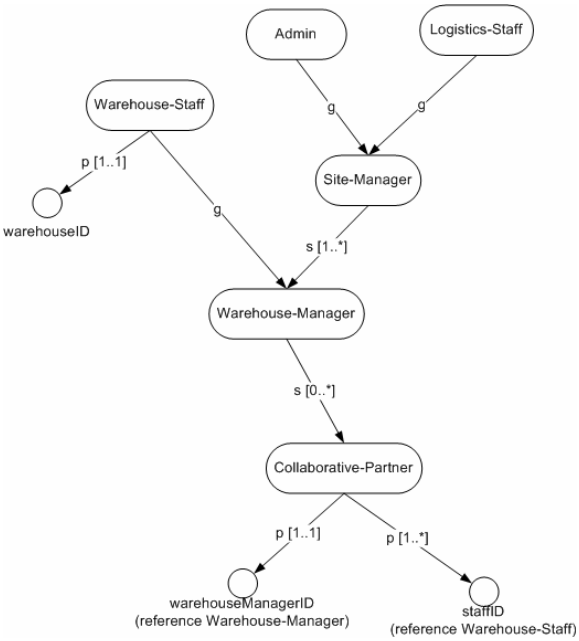

Figure 2: Conceptual view semantic relationship examples (generalization, association)

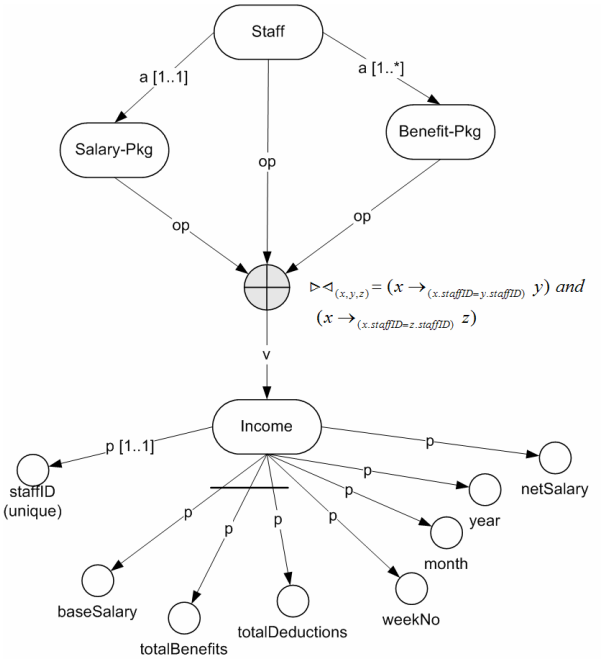

Figure 3: A conceptual view example in the e-Sol ("`Income`")

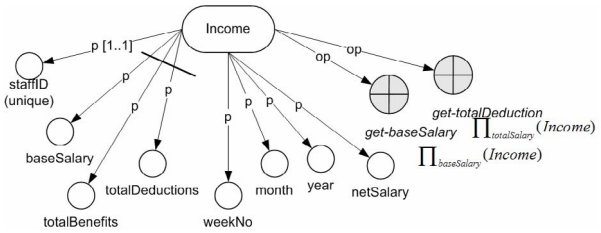

Figure 4: Conceptual view (`Income`) and generic methods

6. Declarative Transformation of LVM Dynamic Properties to Document View Expressions

In our work, we selected XQuery as the document view expression as it is well-suited for our purpose and better than other XML query languages (such as XPath or XSLT) because: (i) XQuery is easy to read and write in comparison to XPath and XSLT, (ii) XQuery has powerful For-Let-Where-Order-Return (FLWOR) expression, in comaprions to XPath/XSLT are (purely) presentation oriented expressions, (iii) XQuery provides User-Defined Functions (UDF) and variable binding, (iv) XQuery support XML Schema and (v) XQuery provide extension mechanism to support new functionalities and data models such as RDF, OWL, etc., such as in [5, 30].

The formal semantics of XQuery can be found in [31] and the new working draft on XQuery update facilities in [26]. In this research, we briefly discuss XQuery as the document view language and for specifying view specific generic methods (namely the *get, set, delete* or *retrieve* methods). This is because XQuery standards do not fully support XML data manipulation yet. But, we choose XQuery as the document view constructor as it is gaining momentum as the language of choice for XML databases and repositories and in the future it will support many of the data manipulation features.

6.1. Transformation of Conceptual Operators to Document View Expressions

The transformation of conceptual operators to XQuery is a 2-step transformation: (a) the declarative transformation of conceptual operator definitions to W3C XQuery expressions; and (b) refinement and validation of XQuery expressions to query engine specific executable code (outside the scope of this research and not addressed in this research). The transformation is done in two steps so as: (a) to keep the conceptual operators and textual XQuery expressions separated from the actual executable XQuery expressions (including both standalone and

embedded code), to keep the transformation functions simpler (almost one-to-one declarative mapping between the conceptual operators to XQuery expressions) and independent of one specific query engine (and one predefined XQuery specification); (b) to achieve MDA like PIM to PSM transformation with more emphasis on reducing vendor specific XQuery engine (or platform) specific syntax and maintain close proximity to the original W3C XQuery syntax; (c) to support forthcoming (and new) XQuery standards (e.g. such as those proposed in [26] and extensions (e.g. XQuery support for Semantic Web meta languages, such as OWL [30]); (d) to achieve portability and cross-platform interoperability between various present and future implementations of XQuery engines.

In addition, here we provide the basic (or skeletal) transformation, that is, it is declarative. This is because: (i) XQuery standards are still evolving and providing a definitive non-declarative transformation may restrict the utilization of new XQuery standards, (ii) if the transformation is generic (i.e. only the resulting skeletal syntax is defined), thus, the document view construction is left as part of the deployment option, that is; (a) document view expression may be stored as predefined functions within a database management system (e.g. Oracle, Tamino, SQL Server, etc., such as stored procedures, triggers or user-defined functions. Also, a given operator may also be mapped to a generic XQuery function template (analogues to User Defined Functions (UDF) [32] in relational and Object-Relational (O-R) models). However, given the evolving nature of the XQuery standards, it is desirable to have the mapping to be as simple as possible, an approach used in this research, UDF syntax, notation, and definition may vary from programmer to programmer and between query engines, (b) document view expression may be deployed as embedded code within a script enabled page or external program modules (e.g. Oracle PL/SQL) and (c) document view expression may be made part of the customized XQuery extensions, such as in [5] or [30], where there exists a need to support application specific or domain specific LVM view construction and (iii) Since the transformations produce skeletal XQuery expression, it can be customized and/or optimized by allowing it be extended (or restricted) to add platform specific requirements and/or environment settings.

It should be noted that the intention of this section is not to introduce and elaborate on XQuery syntax and functions, but rather to address the declarative mapping of conceptual operators to XQuery functions. The transformation described below also includes some of the proposed XQuery extensions by the working draft in [26]. Also, in real-world scenario, conceptual operators used will be a combination of one more basic set as described above, and can be mapped to a sequence of XQuery expressions using the core mappings described in the following sections. Table 1 illustrates a brief summary the proposed transformation described here.

The transformation of the conceptual binary operators to XQuery is to map the conceptual operator to XQuery set operators. Let `node-1` and `node-2` be two node sequences. Here, the nodes can be from one document or from two documents. The transformation of unary operators can be mapped directly to an XQuery FLWOR expressions, except for the rename and restructure operators. For the transformation of rename and restructure operators, we use the proposed XQuery extensions in the W3C working draft [26] to XQuery *rename* and *transform* operators.

***Example 8*:** As shown in Fig. 5, the conceptual selection operator of the view `Warehouse-Staff` can be mapped to the document view construct as shown below in the Code Listing 1.

```
for     $staff in document ("staff.xml")
where
  $staff//StaffMember/Work/@workGroup       =
"warehouse"
order by $staff-member/lastName
return <Warehouse-Staff> {$staff/*
       } </Warehouse-Staff>
```

Code Listing 1: Transformation of $\sigma_{workGroup="warehouse"}(Staff)$ to XQuery FLWOR expression

***Example 9*:** As shown in Fig. 5, the conceptual selection operator of the view `Warehouse-Manager` can be mapped to the document view construct as shown in Code Listing 2.

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

Code Listing 2: Transformation of $\sigma_{Role="manager"}(Staff)$ to XQuery

***Example 10*:** The Code Listing 3 illustrates an order (`staffID`, `lastName`, `firstName`, `deptNo`) constraint applied to the conceptual view `Warehouse-Staff`, as shown in Fig. 5.

***Example 11*:** A *Cartesian product* conceptual operator (Fig. 6) can be mapped to the XQuery

expression, at the document level as illustrated in Code Listing 4.

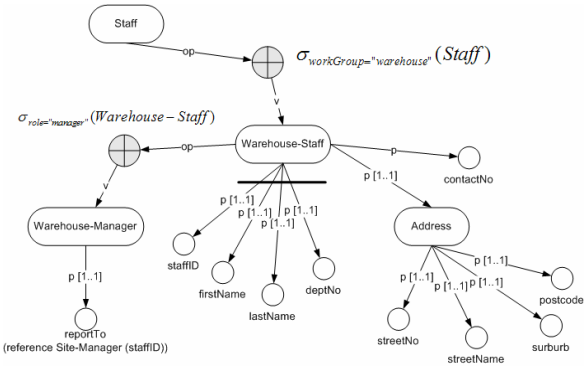

Figure 5: Conceptual view example (Warehouse-Staff)

***Example 12*:** The Code Listing 5 illustrates the transformation of the `Income` conceptual view construct (Fig. 3) to document view expression.

The *rename* conceptual operator is mapped to the XQuery update facility rename operator. The syntax is given in Code Listings 6 - 7. It should be noted here that the rename operator can be also used with projection and selection operator, using the new extensions provided for the XQuery FLWOR expression to support the rename operator.

The *restructure* conceptual operator can be mapped to XQuery using: (i) Implicit mapping: Here, the restructure operator is mapped to a sequence of FLWOR expressions together with optional order and/or where clause and (ii) Explicit mapping: Here, the restructure operator is mapped the new XQuery update facility transform operator to create new structure of an existing document/(s). The syntax is given in Code Listing 8.

***Example 13*:** Code Listing 9 illustrates a generic *restructure* operator mapped to document view expression using the XQuery transform operator.

6.2. Transformation of Generic Methods to Document View Expressions

To transform the generic methods/functions to document view expressions, unlike SQL in relational data model, XQuery standards do not fully support XML data manipulation yet. However, with the prospect of new extensions being proposed for XQuery, transformation described below also includes some of the proposed XQuery extensions by the working draft in [26]. Given the declarative conditions stated in Table 1 above are satisfied for a given view type in the LVM, we can summarise the transformation of its generic methods to XQuery expressions as given in Table 2, which summaries some of the mapping to generic XQuery expressions. Let node-1 and node-2 be two node sequences.

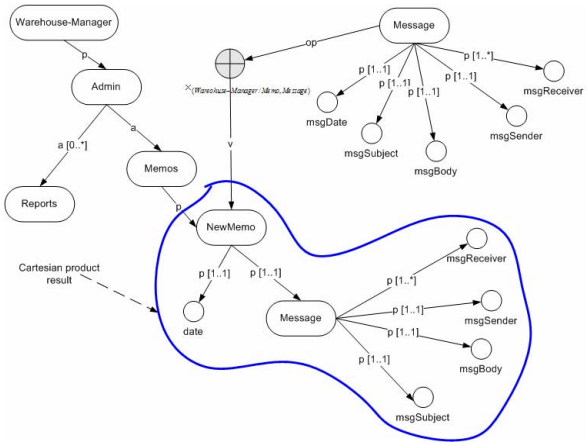

Figure 6: Conceptual view construct (Cartesian product) example

It should be noted here that, generic methods described above (and summarised in Table 2) may also be mapped to XQuery using elaborate UDFs in XQuery [33]. An example of such UDF XQuery syntax is shown in Code Listing 10. However, here we do not use such transformation. Another similar transformation methodology, to map generic and user defined methods to SQL, in the context of Object-Relational paradigm can be found in [32].

Table 2: Summary of the transformation of LVM generic methods to document view expressions (XQuery)

Generic Methods	Comments	XQuery expression
Get methods	Simple FLWOR expression (simple project operators for attribute or elements)	`doc ("node-1.xml")//node/*` or `doc ("node-1.xml")//node/@*`
Update methods	XQuery `replace` operator * Note: here only replacing element values are considered.	`replace` **`value of`** `{fn:doc ("node-1.xml")//element-a[1] with $new-value`
Set methods	XQuery `insert` operator * Note: here the new items are always assumed to be inserted as last	`insert {$new-sub-node} as last into fn: doc("node-1.xml")//nodes /node[OID=$param-oid]`
Delete methods	XQuery `delete` operator	`delete { fn: doc("node-1.xml") //nodes[element=$del-value] }`

```
for     $staff in document ("staff.xml")
where   $staff//StaffMember/Work/@workGroup = "warehouse"
order by $staff-member/lastName, $staff-member/firstName
return
        <Warehouse-Staff>
                <personal-info>
                (: ordered :)
                        <staffID> {$staff/@staffID} </staffID>
                        {$staff/lastName}
                        {$staff/@firstName}
                        <deptNo> {$staff/@deptNO} </deptNo>
                </personal-info>
                <Address> $staff/address/* </Address>
                <Contact-No> $staff/contactNo/* </Contact-No>
        </Warehouse-Staff>
```

Code Listing 3: Transformation of conceptual selection operator (with order constraint) to XQuery expression

```
for $memo in document ("Warehouse-Manager.xml")//Admin,
       $msg in document ("Messages.xml")//Msg
where $msg/@date= today()
return <Memo>  {
               $memo/Memos}
               <newMemo> {$msg/@*} {$msg/MsgBody} </newMemo>
        </Memo>
```

Code Listing 4: Transformation of $\times_{(Wareohuse-Manager/Memo,Message)}$ to XQuery expression (Cartesian product)

```
for $staff in document ("staff.xml")//Staff-member,
     $sal in document ("staff.xml")//Salary-Pkg,
     $benefits in document ("staff.xml")//Benefit-Pkg
let $totBenefits := $sal/Family-Support/totalAmount +
                       $sal/Executive-Support/totalAmount
let $netSal := $sal + $totBenefits - $sal/deductionAmount
where $staff/@staffID = $sal/@staffID and
          $staff/@staffID = $benefits/@staffID
return <Income>  {$staff/@staffID}
   {$staff/FirstName}
   {$staff/LastName}
   {$staff/Tax-SSN}
   <baseSalary> {$sal/base} </baseSalary>
   <totalBenefits> {$totBenefits} </totalBenefits>
   <totalDeductions> {$sal/deductionAmount} </totalDeductions>
   <payMonth> {month (), year()} </payMonth>
   <netSalary > {$netSal} </netSalary>
</Income>
```

Code Listing 5: Transformation of the `Income` conceptual view construct $\triangleright\triangleleft_{(x,y,z)} = (x \rightarrow_{(x.staffID=y.staffID)} y)$ *and* $(x \rightarrow_{(x.staffID=z.staffID)} z)$ to XQuery expression

```
rename
        {fn:document ("node-1.xml")/old-element-name}
to "new-element-name"
```

Code Listing 6: Transformation conceptual rename operator to XQuery expression (renaming an element)

```
rename
        {fn:document ("node-1")/nodes[1]/element-old-value[1]}
to $value0in-variable
```

Code Listing 7: Transformation conceptual rename operator to XQuery expression (renaming a value using a variable)

```
for     $a document ("node-1.xml")
where $a//selection-condition = value-or-nested-query
order by $a/element-2  (: new ordered structure :)
return <new-structure-1>
        {$a//element-4/*
        <new-structure-2>{$a//element-3/*
                <new-structure-3> {$a//element-1/*
                }</new-structure-3>
     } </new-structure-2>
   } </new-structure-1>
```

Code Listing 8: Transformation of conceptual restructure operator to XQuery expression (option (i))

```
for $a in doc ("node-1.xml")//nodes
return
        transform
                copy $trans-a := $a
                do delete {$trans-a/node-1}
return $trans-a
```

Code Listing 9: Transformation of restructure operator to XQuery expression (option (ii))

```
define function get-Warehouse-Staff-firstName ($param-staffID)as element ()*
{
   for  $staffMem in document ("staff.xml")//StaffMember
   where some $staffID in $staffMem/@staffID satisfies
        ($staffID = $param-staffID)
        and
        ($staffMem/Work/@workGroup = "warehouse")
   return $staffMem/@firstName
}
```

Code Listing 10: An example transformation of a generic get method (getFirstName()) in Warehouse-Staff to an XQuery UDF

7. Conclusion & Future Work

In this paper, we presented a modeling notation to capture dynamic properties in the layered view model using XSemantic nets. We also provided a declarative transformation of such dynamic properties into document view (query) expressions.

For future work, some further issues deserve investigation. First, the investigation of a formal mapping approach to conceptual view (dynamic) properties to query expressions and the automation of such transformation. Second, is the investigation into dynamic perspectives of such conceptual view formalism that can be applied to traditional, Semantic Web and web service data to conceptually model the domain in question.

8. References

[1] 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.

[2] 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.

[3] R.Rajugan, "A Layered View Model for XML with Conceptual and Logical Extension, and its Applications," in *Faculty of Information Technology*. Sydney: University of Technology, Sydney (UTS), Australia, 2006, pp. 460.

[4] S. Abiteboul, O. Benjelloun, I. Manolescu, T. Milo, and R. Weber, "Active XML: Peer-to-Peer Data and Web

0-7695-2577-6/06 $20.00

Services Integration," Proceedings of the 28th International Conference on VLDB, HK, China, 2002.
[5] L. Feng and T. S. Dillon, "An XML-Enabled Data Mining Query Language XML-DMQL," *International Journal of Business Intelligence and Data Mining*, 2005.
[6] R.Rajugan, E. Chang, T. S. Dillon, and L. Feng, "Modeling Views in the Layered View Model for XML Using UML," *International Journal of Web Information Systems (IJWIS), Troubador Publisher Ltd.*, vol. 2(2), pp. 95-117, 2006.
[7] W3C-XQuery, "XQuery 1.0: An XML Query Language (http://www.w3.org/TR/xquery)," in *XML Query Language (XQuery)*: The World Wide Web Consortium (W3C), 2004.
[8] C. J. Date, *An introduction to database systems*, 8th ed. New York: Pearson/Addison Wesley, 2003.
[9] E. F. Codd, *The Relational Model for Database Management: Version 2*: Addison Wesley Publishing Company, 1990.
[10] W3C-SW, "The Semantic Web (http://www.w3.org/2001/sw/)," W3C, 2005.
[11] R.Rajugan, E. Chang, T. S. Dillon, and L. Feng, "A Three-Layered XML View Model: A Practical Approach," 24th International Conference on Conceptual Modeling (ER '05), Klagenfurt, Austria, 2005.
[12] 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.
[13] 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.
[14] V. Aguilera, S. Cluet, T. Milo, P. Veltri, and D. Vodislav, "Views in a Large-Scale XML Repository," *The VLDB Journal — The International Journal on Very Large Data Bases*, vol. 11(3), pp. 238-255, 2002.
[15] 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.
[16] Xyleme, "Xyleme Project (http://www.xyleme.com/)," 2001.
[17] 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.
[18] 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.
[19] 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.
[20] S. Abiteboul, O. Benjelloun, I. Manolescu, T. Milo, and R. Weber, "Active XML: A Data-Centric Perspective on Web Services," BDA, 2002.
[21] S. Abiteboul, B. Amann, S. Cluet, A. Eyal, L. Mignet, and T. Milo, "Active Views for Electronic Commerce," Proceedings of the 25th International Conference on VLDB, Edinburgh, Scotland, 1999.
[22] OMG-UML™, "UML 2.0 Final Adopted Specification (http://www.uml.org/#UML2.0)," vol. 2005: OMG, 2003.
[23] W3C-XSD, "XML Schema (http://www.w3.org/XML/Schema)," vol. 2004, 2 ed: W3C, 2001.
[24] R.Rajugan, E. Chang, T. S. Dillon, and L. Feng, "Alternate Representations for Visual Constraint Specification in the Layered View Model," The Seventh International Conference on Information Integration and Web Based Applications & Services (iiWAS '05), Kuala Lumpur, Malaysia, 2005.
[25] ANSI and ISO, "ANSI - SQL 2003," ANSI / ISO 2003.
[26] W3C-XQuery-UF, "XQuery Update Facility (http://www.w3.org/TR/2006/WD-xqupdate-20060127/) - W3C Working Draft 27 January 2006," in *XML Query Language (XQuery)*: The World Wide Web Consortium (W3C), 2006.
[27] E. Chang, T. Dillon, W. Gardner, A. Talevski, R.Rajugan, 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.
[28] E. Chang, W. Gardner, A. Talevski, E. Gautama, R.Rajugan, T. Kapnoullas, and S. Satter, "Virtual Collaborative Logistics and B2B e-Commerce," e-Business Conference, Duxon Wellington, NZ, 2001.
[29] ITEC, "iPower Logistics (http://www.logistics.cbs.curtin.edu.au/)," 2002.
[30] P. Lehti and P. Fankhauser, "SWQL – A Query Language for Data Integration Based on OWL," First IFIP WG 2.12 & WG 12.4 International Workshop on Web Semantics (SWWS '05), In conjunction with On The Move Federated Conferences (OTM '05), Agia Napa, Cyprus, 2005.
[31] W3C-XqFM, "XQuery 1.0 and XPath 2.0 Formal Semantics (http://www.w3.org/TR/xquery-semantics/)," W3C Candidate Recommendation 3 November 2005 ed: The World Wide Web Consortium (W3C), 2005.
[32] J. W. Rahayu, "Object-Relational Transformation Methodology," in *Department of Computer Science & Computer Engineering*: La Trobe University, Melbourne, Australia, 2000.
[33] D. D. Chamberlin and H. Katz, *XQuery from the experts : a guide to the W3C XML query language*. Boston: Addison-Wesley, 2003.