ALTERNATE REPRESENTATIONS FOR VISUAL CONSTRAINT SPECIFICATION IN THE LAYERED VIEW MODEL

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

Abstract
Extensible Markup Language (XML), with its rich set of semantics and constraints, 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 semi-structured data and schemas, there exists a requirement to model, design, and constrain semi-structured data and the associated semantics at a higher level of abstraction than at the instance or data level. But most semi-structured schema languages lack the ability to provide higher levels of abstraction, such as visual constraints, that are easily understood by humans. Conversely, though Object-Oriented (OO) conceptual models offers the power in describing and modeling real-world data semantics, constraints and their inter-relationships in a form that is precise and comprehensible to users, they provide insufficient modeling constructs for utilizing XML schema like data descriptions and constraints. Therefore, it is interesting to investigate conceptual and schema formalisms as a means of providing higher level semantics in the context of XML-related data engineering. In this paper, we present a visual constraint specification model for an XML layered view model. First we briefly outline the view model and then provide a detailed discussion on modeling issues related to view constraint specification using two OO modeling languages, namely OMG's UML/OCL and XML Semantics (XSemantic) nets. To demonstrate our concepts, we also provide an illustrative case study example based on a real-world application.

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, constraints 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] . These models span from early data centred models (e.g. Entity Relationship/Data Flow Diagrams (ER/DFD)) [2] to the modern Object-Oriented (OO) models [1, 3], where a software system is modeled at varying levels of abstractions.

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]. Conversely,

since the introduction of eXtensible Markup Language (XML) [4], 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 [5] 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.

Since the introduction of Model-Driven Architecture (MDA) [6] initiative by OMG, platform independent models play a vital role in system development and data engineering. Under the MDA initiative, first the model of a system is specified via an abstraction notation that is independent of the technical or deployment specifications (i.e. Platform Independent Model or PIM) and then the PIM is mapped or transformed into a deployment model (i.e. Platform Specific Model or PSM) by adding platform or deployment specific information. To support MDA initiatives in data engineering, data semantics, constraints and model requirements have to be specified precisely at a higher level of abstraction.

In the context of MDA solutions for XML domains, it is still a challenging task to produce PIMs despite the flexibility and the semantic richness of the semi-structured schema languages. This is mainly due to OO modeling languages such as OMG UML™ [7], Extended-ER [2] etc. provide insufficient modeling constructs for utilizing XML schema like data descriptions and constraints, while XML Schema lacks the ability to provide higher levels of abstraction (such as conceptual models, visual constraints, etc.) that are easily understood by humans. This is in the view that, models are often abstract representations which only keep so much of the detail as is relevant to the particular problem being considered [1, 3]. In this context, XML Schema generally is too low a representation to permit users to interact, visualize or understand it.To rectify this situation, many researchers in work such as [8-11], have applied intuitive techniques, notations and transformation methodologies to capture XML semantics at the conceptual level. This presents an opportunity to investigate data views as a means of providing data abstraction and semantics in PIMs for data intensive MDA solutions, such as XML document warehouses. But, existing OO paradigm and modeling languages provide minimal or no semantics to capture abstract view formalisms (at the conceptual and logical levels) [11] and existing semi-structured data technology standards and languages do not provide support for concrete view formalisms.

In our previous work, to address the above issue, we proposed a layered view model [11, 12] with three levels of abstraction, namely, (a) conceptual, (2) logical (or schema) and (3) document (or instance) level. In this paper, we elaborate on visual constraint specification in our layered view model using two OO modeling languages, namely; (a) OMG UML/OCL and (2) XML Semantic (XSemantic) nets [8, 13]. It should be ***noted*** that, our intention here is neither to provide view support for XML nor extensions to XML query languages such as XQuery or SQL 2003. Our proposed view formalism is independent of the instance/document level and focused on providing view formalism at the conceptual, logical and document levels by means of OO conceptual modelling and transformation formalisms.

The rest of this paper is organized as follows. In section 2, we review some early work in view constraint specifications, followed by section 3, where we briefly outline our layered XML view model. In section 4, we provide a description of an illustrative, real-world case-study example used in this paper. Section 5, presents our constraints model for our layered view model, followed by the discussion on transformation of view constraints to view schema constraints in section 6. We conclude the paper section 7 with a brief discussion on our future work.

2. Related Work: View Constraints

Based on existing view formalisms and research directions, we can group them into four (4) categories, namely [12]; (1) classical or relational views, (2) OO view models, (3) views for XML, and (4) views for semantic web and Ontology bases. A detailed, critical discussion on these view models can be found in our work [12]. Here, in this paper, we focus mainly on view constraints.

In data modeling, specifications often involve constraints. In the case of views, view constraints fall into the category of data elaboration, which is one of the 2-Es of a view model; data Extraction and Elaboration [14]. Also, typically, these view constraints are specified using the data language in which the views are defined in. For example, in relational model, views are defined using SQL and a limited set of constraints can be defined using SQL [2, 15], 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). Due to such language dependent view specification mechanisms, view definitions and constraints still remains a data language and model dependent, lower level (implementation) construct.

In Object-Relational and OO models, views had similar constraints but they are more extensive and explicit due to the data model. The views here are constructed and specified by DBMS specific (such as OQL[16]) and/or external languages (such as C++, Java or O_2C [17]). 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 [18] or OO programming languages such as Java [19]. Today, in the case of ontology engineering (and in ontology views), this is still holds true, where constraints are specified using programming modules [14] 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 in [20] provides some form of higher-level view constraints (under ORA-SS model) for XML views, while the work [21] provides some form of logical level view constraints to be defined in views for in Sematic Web [22]/RDF [23] paradigm.

It is also important to note and distinguish the work of [24], where authors use Object Constraint Language (OCL) [25] to "model" relational views and utilizes OCL for data modeling and view specification, than specifying (view) constraints.

3. The Layered View Model: A Brief Introduction

Our view model for XML comprised of three different levels of abstraction (Fig. 1), 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.

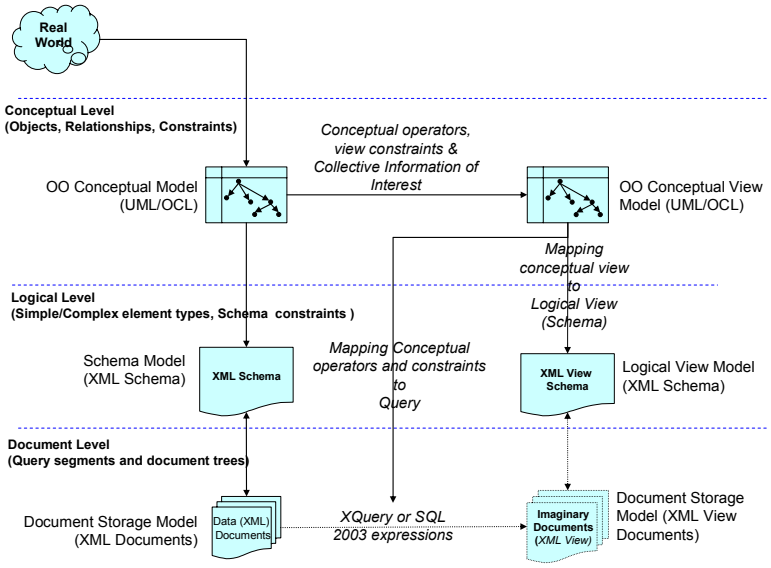

Figure 1: Layered XML View Model (context diagram)

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.

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 [12, 26], or our developed XML-specific XSemantic net [13], 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 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** $\boldsymbol{V^c}$ is a 4-ary tuple $\boldsymbol{V^c} = (V^c_{name}, V^c_{obj}, V^c_{rel}, V^c_{constraint})$, 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 [27]. $\boldsymbol{V^c} = (V^c_{name}, V^c_{obj}, V^c_{rel}, V^c_{constraint})$ is called a *valid conceptual view of the context* $\boldsymbol{C}$, if and only if the following conditions satisfy;

- For any object $\forall o \in V^c_{obj}$, there exist objects $\exists o_1, \ldots, o_n \in C_{obj}$, such that $o = \lambda_1 \ldots \lambda_m (o_1, \ldots, o_n)$ where $\lambda_1 \ldots \lambda_m \in \hat{\lambda}$. That is, o is a newly derived object from existing objects $o_1, \ldots, o_n$ in the context via a series of conceptual operators $\lambda_1, \ldots \lambda_m$ like select, join, etc .
- 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}.
- 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}.
- 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}.

The middle level of the view model is the scheme (or logical) 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 [8, 9]. 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* $\boldsymbol{V^s}$ is a triple $\boldsymbol{V^s} = (V^s_{name}, V^s_{simpleType}, V^s_{complexType}, V^s_{constraint})$, 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 [4].

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, $\aleph^c_s : V^c \rightarrow V^s$.

The bottom instance level implies a fragment of instantiated XML data, which conforms to the corresponding view schema defined at the upper level. At this level, XML data instantiation is achieved by transforming conceptual operators (and constraints, if any) to any native XML query language (e.g. XQuery) query language. A detailed discussion that describes this level and the mapping methodology can be can be found in our work [12, 27].

4. An Illustrative Example Case-study

The e-Sol Inc. aims to provide logistics, warehouse, and cold storage space for its global customers and collaborative partners (Fig. 2). 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 [28] 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 [28, 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 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. A context diagram of the system is given in Fig. 2.

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.

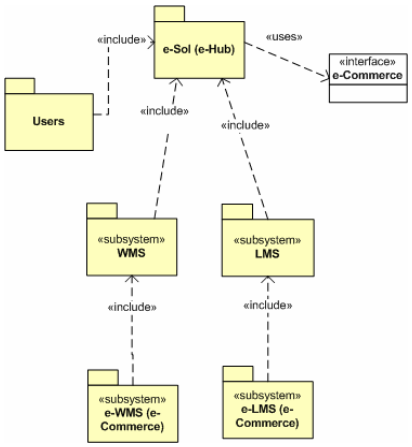

Figure 2: e-Sol context diagram

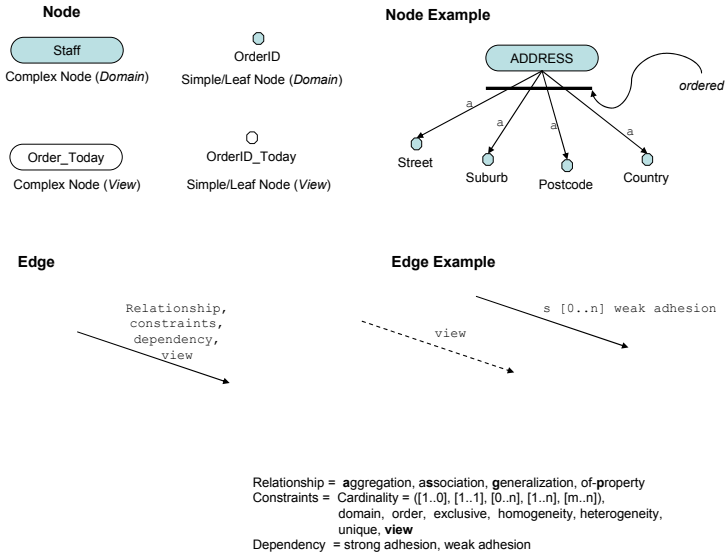

Figure 3: A XML Semantic (XSemantic) net notations

Example 1: Context, "`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. 8(a)/8(b), "`Warehouse-Manager`" is a valid XML conceptual view, named in the context of "`Staff`". Its constructed using the conceptual SELECT operator [27], which can be shown as; $\sigma_{warehouse\text{-}Staff.Role=\text{“}manager\text{”}}(Core\text{-}Users)$

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 Cartesian Product conceptual operator, where x = `Warehouse-Manager` and y = `Management-Memo`; $\times_{(x,y)} = \mathcal{R} = x \times y$

Example 5: In the case of conceptual view "`Income`" (shown in Fig. 9(a)-9(b)), the conceptual construct is a conceptual JOIN operator with join conditions, where x = `Staff`, y = `Salary-Pkg` and z = `Benefit-Pkg`: $(x \rightarrow_{(x.staffID=y.staffID)} y)$ *AND* $(x \rightarrow_{(x.staffID=z.staffID)} z)$

5. Specifying Constraints for Conceptual Views

Constraint specification for conceptual views can be explicit and extensive in comparison to relational or OO view constraints as, the conceptual views are designed at a higher-level. Here, the constraint specifications are restricted only by the modelling language semantics and not by the view definition language as in the case of relational (e.g. SQL) and/or OO (e.g. OQL) views. In addition, further constraints can be defined for conceptual views including; (i) domain constraints (range of values, min, max, pattern etc), (ii) constructional contents (set, sequence, bag, ordered-set), (iii) ordering (iv) explicit homogenous composition/heterogeneous compositions, (v) adhesion and/or dependencies (vi) exclusive disjunction and many more. A detailed discussion on constraint specification for (stored) XML domains in UML can be found in the work [9], while detailed discussion on XSemantic net constraints in [8]. Specifying these constraints in UML/OCL or XSemantic nets for conceptual views are similar to that of stored domain object constraints.

5.1. Constraint Specification Using UML/OCL

In UML, the Object Constraint Language (OCL) [25], which is now a part of the UML 2.0 standard [7], can support unambiguous constraints specifications for UML models including specification of static and dynamic (e.g. messages constraints) model elements . In our conceptual view model, we incorporate OCL (in addition to built-in UML constraint features such as cardinality constraints [1, 9]) as our view constraint specification language to explicitly state view constraints. It should be noted that, we do not use OCL to *define* or *specify* views, rather state additional constraints using OCL. OCL supports defining *derived* classes [25, 30], which is close to a view concept [24]. It is of the form of;

```
context:    <derived-class-name>::<new-attribute-name>: Type
derive:     <source-stored-class>.
            [some expression representing the derivation rule]
            / [<source-stored-class-attribute>
```

To model our conceptual views, we show view classes visually, with the `<<view>>` stereotypes and the relationship between the stored class and the view as `<<construct>>` stereotype [26]. Therefore, we do not require non-visual OCL view specification as shown above, but may be used to show some of the derivations rule for the attributes and/or operations to make the view definition more explicit and precise. OCL also supports specifying derived values and attributes in already existing views (and stored classes) and specified in the form of;

```
context Typename::assocRoleName: Type
derive: -- some expression representing the derivation rule
```

5.2. Constraint Specification Using XSemantic Net

XML Sematic (XSemantic) net is a modified semantic network model notation (Fig. 3), to model XML domains using the OO concepts [8, 13]. In XSemantic nets, due to its structural similarity to XML, most XML data/schema specific constraints are build-in to the model. There exist no need to have additional (textual) constraint specification language (e.g. such as OCL). These constraints are grouped into three categories [8], namely; (a) constraints over an edge, (b) constraints over a set of edges and (c) constraints over an edge. Since XSemantic nets and XML share structural similarities, the constraints are very explicit and transformation of XSemantic net model to XML Schema is straight forward and automatic [8]. The modelling primitives (or notation) used in XSemantic net is given in Fig. 3.

5.3. An Illustrative Walkthrough

In this section, we describe some constraints that frequently occur in real-world with the help of illustrative case-study examples. Our aim here is not to compare and contrast or to promote one particular notation (i.e. UML/OCL or XSemantic nets) over the other. Rather, we provide discussion on different types of constraints that are observed in real-world scenarios and how they can be modelled here, in the context of conceptual views, not just one but two OO modelling languages.

1) Ordered/Unordered Composition

In real-world, composite objects being in an aggregation with one or more sub-objects, they also can be in a pre-defined order. For example in XML Schema construct such as with `<xsd:sequence>`, we regularly observe that the tag `<xsd:sequence>` signifies that the embedded elements are not only a simple assortment of components but these have a specific ordering. This signifies an important OO concept, *ordered composition*. Simply said, to capture ordering, we add an UML stereotype that allows capturing of the ordered composition utilizing stereotypes to specify the objects' order of occurrence such as `<<1>>, <<2>>, <<3>>, .… ,<<n>>`, as shwon in Fig. 4(a). In related work [31], authors have shown similar appraoch and extensively discussed defining such ordered composition and mapping it to XML Schema. In the case of XSemantic nets, ordering can be modeled as shown in Fig. 4(b).

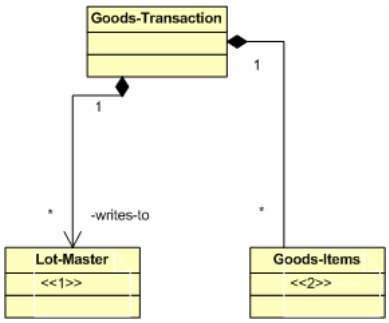

Figure 4(a): *Ordered* composition (UML/OCL)

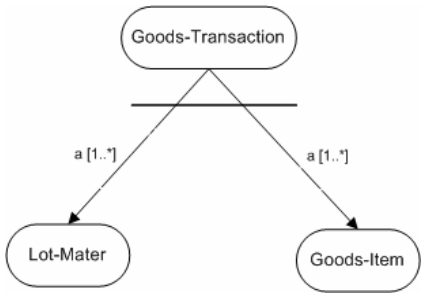

Figure 4(b): *Ordered* composition XSemantic net)

2) Unique Constraint

In many OO conceptual models and diagrams, though the concept of OID is assumed to be an implicit concept (unlike primary keys in E/ER), in our work, with *conceptual views*, we have a need to explicitly state the OIDs and should be available to visualize at that highest level of abstraction. Therefore, here, we provide a means of using OIDs for the purpose of IDs, similar to that of primary/foreign key or unique constraints available in E/ER models. We argue that, just utilizing OID (a unique concept to OO systems) in our conceptual model provides additional semantics, such as providing Id/keys, referential and integrity constraints that are visually lacking in many OO conceptual modelling technique. Fig. 5(a) (with the OCL expression in example 7) shows how this is used in UML/OCL, while Fig. 5(b) shows how this is done in XSemantic nets.

Example 6: In the case of conceptual view "`Warehouse-Manager`" (Fig. 5(a) and 5(b)), we indicate the unique `staffID` by the following OCL expression;

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

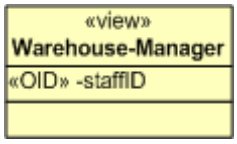

Figure 5(a): OID / Unique Constraint (UML/OCL)

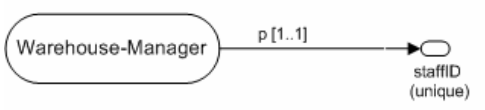

Figure 5(b): OID / Unique Constraint (XSemantic net)

3) Exclusive disjunction

Example 7: In the case of conceptual views "`Lot-Movement`", 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. 6(a)) and by an arc in XSemantic net (as shown in Fig. 6(b)).

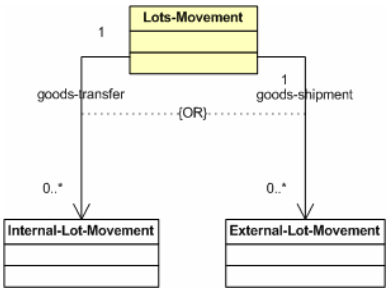

Figure 6(a): Exclusive disjunction (UML/OCL)

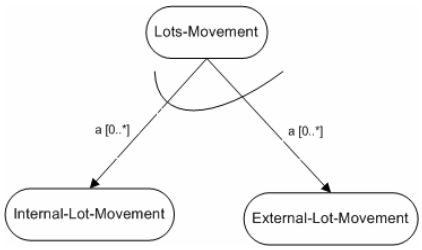

Figure 6(b): Exclusive disjunction (XSemantic net)

4) Cardinality Constraint

The cardinality constraint shows the number of instances of one class that may relate to single instance of another. This is similar to that of relational and OO data models, as shown in Fig. 7(a) and Fig. 7(b).

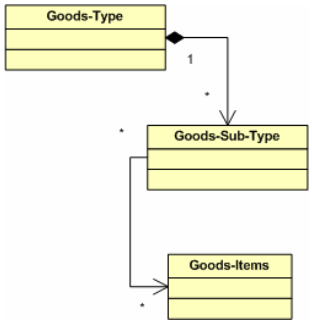

Figure 7(a): Cardinality constraints (UML/OCL)

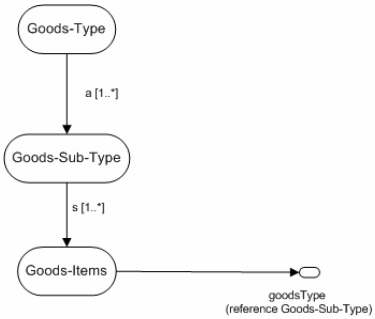

Figure 7(b): Cardinality constraints (XSemantic net)

5) Dependency / Adhesion Constraint

Example 8: In the case of conceptual views "`Warehouse-Manager`" and "`Warehouse-Staff`", in the context of "`Staff`" (Fig. 8(a)-8(b)), 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 9: In the case of conceptual view "`Income`" (Fig. 9(a) & 9(b)), 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
```

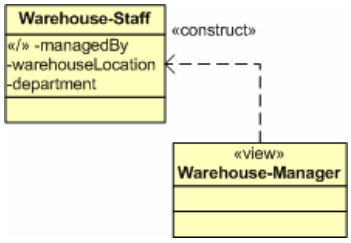

Figure 8(a): Dependency / Adhesion constraint (UML/OCL)

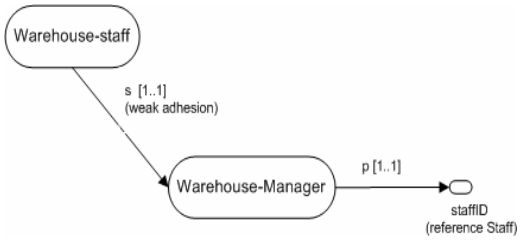

Figure 8(b): Dependency / Adhesion constraint (XSemantic net)

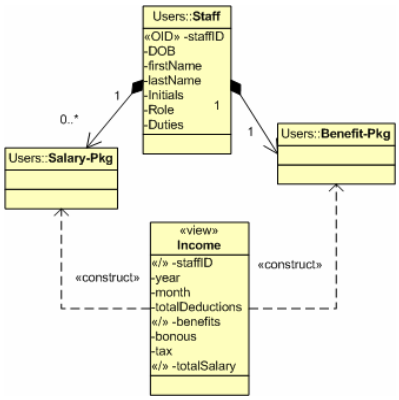

Figure 9(a): Dependency / Adhesion constraint (UML/OCL)

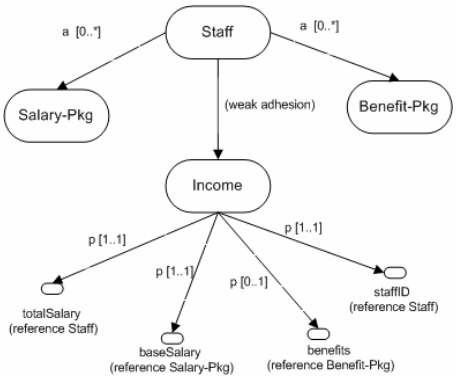

Figure 9(b): Dependency / Adhesion constraint (XSemantic net)

6. Transformation of Conceptual View Constraints to Logical View Constraints

At the logical level, as shown in Fig. 1, conceptual view properties are transformed to XML schema, thus generating the (XML) view schema. In our previous works such as [12, 13, 26], 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. Here, we briefly demonstrate mapping some conceptual view constraints to XML schema.

Example 10: The ordered/unordered compositions (e.g. in Fig. 4(a)-4(b)), "`Lot-Master`" & "`Goods-Items`"), shown using the stereotype (`<<1>>, <<2>>,…`), are mapped using the `<xs:sequence>` construct; (a) at the conceptual view attribute level and (b) at the conceptual view class level. This shown below in the code listing;

```
<!-- additional nesting -->
<xs:complexType name="Composite_Object/attrbute">
   <xs:sequence>
      <xs:element name="Componant-A/attribute-a" type="xs:OID"/>
      <!-- additional nesting -->
      <!-- additional nesting -->
   </xs:sequence>
</xs:complexType>
<!-- additional nesting -->
```

Example 11: The unique constraint (the <<OID>>) in Staff can be mapped to the view schema as shown in the code fragment below (Fig. 5(a) & 5(b)).

```
<!-- additional nesting -->
<xs:complexType name="staffType">
      <xs:sequence>
         <xs:element name="staffID" type="OIDType"/>
         <!-- additional nesting -->
         <xs:element name="managedBy">
            <xs:key name="manager">
               <xs:selector/>
               <xs:field xpath="staffID"/>
            </xs:key>
         </xs:element>
      </xs:sequence>
   </xs:complexType>
   <!-- additional nesting -->
   <!-- additional nesting -->
   <xs:complexType name="OIDType">
      <xs:sequence>
         <xs:element name="ID">
            <xs:unique name="OID">
               <xs:selector xpath="OIDType"/>
               <xs:field xpath="ID"/>
            </xs:unique>
         </xs:element>
      </xs:sequence>
   </xs:complexType>
<!-- additional nesting -->
```

Example 12: As shown in example 8, the exclusive disjunction constraint (Fig 6(a)-6(b)) between "Internal-Lot-Movement" and "External-Lot-Movement" can be mapped XML Schema using the <xs:choice> schema construct, as shown below in the code listing;

```
<!-- additional nesting -->
<xs:element name="Lot-Movement">
   <xs:complexType>
      <xs:complexContent>
         <xs:extension base="LotMovementType">
         <!-- additional nesting -->
            <xs:choice>
               <xs:element name="Internal-Lot-Movement" type="InternalLotMovementType"/>
               <!-- additional nesting -->
               <xs:element name="External-Lot-Movement" type="ExternalLotMovementType"/>
               <!-- additional nesting -->
            </xs:choice>
            <!-- additional nesting -->
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
</xs:element>
<!-- additional nesting -->
```

7. Conclusion and Future Work

In this paper, we presented a visual view constraint specification at the conceptual level (and corresponding logical level) to support data engineering under the MDA initiatives. The visual view constraint specification is demonstrated using two OO modelling languages, namely (a) UML/OCL and (b) XSemantic nets, where the views are defined using conceptual semantics. We also briefly elaborated on a transformation methodology to map conceptual view constraints to (XML) schema constraints.

For future work some issues deserve investigation. First, is to develop an empirical study to validate the view constraint model. Second is the investigation of automating the mapping process between the conceptual constraint models to schematic constraints. Third, is the investigation into specifying constraints for dynamic perspectives of the view model.

8. References

[1] T. S. Dillon and P. L. Tan, *Object-Oriented Conceptual Modeling*: Prentice Hall, Australia, 1993.
[2] R. Elmasri and S. Navathe, *Fundamentals of database systems*, 4th ed. NY York: Pearson/Addison Wesley, 2004.
[3] I. Graham, et al., *Object-oriented methods : principles & practice*, 3rd ed. Harlow: Addison-Wesley, 2001.
[4] W3C-XML, "Extensible Markup Language (XML) 1.0, (http://www.w3.org/XML/)," 3 ed: The W3C , 2004.
[5] W3C-XSD, "XML Schema," vol. 2004, 2 ed: W3C, 2004.
[6] OMG-MDA, "The Architecture of Choice for a Changing World®, MDA Guide Version 1.0.1 (http://www.omg.org/mda/)," OMG, 2003.
[7] OMG-UML™, "UML 2.0 Final Adopted Specification (http://www.uml.org/#UML2.0)," 2003.
[8] 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.
[9] 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.
[10] R. Conrad, D. et al., "XML conceptual modeling using UML," 19th Int. Conference on ER '00, USA, 2000.
[11] R.Rajugan, E. Chang, T. S. Dillon, and L. Feng, "XML Views: Part 1," 14th International Conference on Database and Expert Systems Applications (DEXA '03), Prague, Czech Republic, 2003.
[12] 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.
[13] R.Rajugan, E. Chang, L. Feng, and T. S. Dillon, "Semantic Modelling of e-Solutions Using a View Formalism with Conceptual & Logical Extensions," Int. IEEE Conf. on Ind. Informatics. (INDIN '05), Perth, Australia, 2005.
[14] C. Wouters, R.Rajugan, 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. J. Date, *An introduction to database systems*, 8th ed. New York: Pearson/Addison Wesley, 2003.
[16] R. G. G. Cattell, et al., "The Object Data Standard: ODMG 3.0," Morgan Kaufmann, 2000, pp. 300.
[17] S. Abiteboul and A. Bonner, "Objects and Views," ACM SIGMOD Record, Proceedings of the International Conference on Management of Data (ACM SIGMOD '91), 1991.
[18] 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.
[19] S. Abiteboul, et al., "Active XML: A Data-Centric Perspective on Web Services," BDA, 2002.
[20] 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.
[21] 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.
[22] W3C-SW, "(http://www.w3.org/2001/sw/)," W3C, 2005.
[23] W3C-RDF, "Resource Description Framework (RDF), (http://www.w3.org/RDF/)," 3 ed: The W3C, 2004.
[24] 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.
[25] OMG-OCL, "UML 2.0 OCL (http://www.omg.org/cgi-bin/doc?ptc/2003-10-14)," OMG, 2003.
[26] R.Rajugan, E. Chang, T. S. Dillon, and L. Feng, "XML Views, Part III: Modeling XML Conceptual Views Using UML," 7th International Conference on Enterprise Information Systems (ICEIS '05), Miami, USA, 2005.
[27] R.Rajugan, E. Chang, T. S. Dillon, and L. Feng, "A Layered View Model for XML Repositories & XML Data Warehouses," The 5th Int. Conf. on Computer and Information Technology (CIT '05), Shanghai, China, 2005.
[28] 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.
[29] ITEC, "iPower Logistics (http://www.logistics.cbs.curtin.edu.au/)," 2002.
[30] J. B. Warmer and A. G. Kleppe, *The object constraint language : getting your models ready for MDA*, 2nd ed. Boston, MA: Addison-Wesley, 2003.
[31] V. Nassis, R.Rajugan, T. S. Dillon, and W. Rahayu, "Conceptual and Systematic Design Approach for XML Document Warehouses," *International Journal of Data Warehousing and Mining*, vol. 1, No 3, 2005.