Semantic Modeling of e-Solutions Using a View Formalism with Conceptual & Logical Extensions

Rajugan, R.[1], MIEEE, Elizabeth Chang[2], MIEEE, Ling Feng[3], MIEEE, and Tharam S. Dillon[1], FIEEE

[1] eXel Lab, Faculty of IT, University of Technology, Sydney, Australia, e-mail: (rajugan, tharam)@it.uts.edu.au
[2] School of IS, 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@cs.utwente.nl

Abstract— **In industrial informatics, there exists a requirement to model and design views at a higher level of abstraction. Since the classical view definitions are only available at the query or instance level, modelling and maintaining such views for complex Enterprise Information Systems (EIS) is a challenging task. Further, the introduction of semi-structured data (namely XML) and its rapid adaptation by the commercial and industrial systems increased the complexity for view design and specification. To address such and issue, in this paper we present; (a) a layered view model for XML, (b) a design methodology for such views and (c) some real-world industrial applications of the view model. The XML view formalism is defined at the conceptual level and the design methodology is based on the XML Semantic (XSemantic) nets, a high-level Object-Oriented (OO) modelling language for XML domains.**

Index Terms— **XML, XML schema, conceptual views, logical views and document views.**

I. Introduction

In software engineering, many methodologies have been proposed to capture real-world problems into manageable segments, which can be communicated, modelled and developed into error-free maintainable software models and modules [1]. Conversely, as industrial production techniques move towards a distributed model, the need to exchange data between heterogeneous data sources in a seamless fashion is constantly increasing. These heterogeneous data sources could arise from server groups from different manufacturers or databases at different sites with their own schemas. Since the introduction of OMG's Model Driven Architecture (or MDA™) [2], it presents an interesting paradigm for Industrial Informatics and data modelling. Under the MDA initiatives, specifications of the system operations are separated from the details of the platform/implementation specific syntaxes and specifications. For such approach to be successful, all models have to be specified in an orderly manner, at a higher-level of abstraction, which in turn should be easily mapped to platform specific specifications.

Similarly, in the case of data models, the main objective of the conceptual model 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 [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. As a result, lately, XML has become the *defacto* standard for storing and manipulating self-describing information, which creates vocabularies in assisting information exchange between heterogenous enterprise data sources over the web [6, 7]. With enterprise content and EIS moving rapidly towards web-based e-Commerce/e-Business and Information Systems, XML contents add increased complexity for engineering data.

But, existing Object-Oriented (OO) modeling languages (such as UML, EER) 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. Also, existing OO paradigm and modeling languages provide minimal or no semantics to capture *abstract view formalisms* (at the conceptual and logical levels) and the existing XML technology standards have no support for *concrete view formalisms*.

To solve this problem, we proposed a layered view model and a design methodology for XML, that is generic enough to be modelled in any high-level modelling language that support OO paradigm (such as UML, XSemantic Nets [8] etc.). This view model, can provide the platform independent data abstraction for data intensive EIS and databases using conceptual and schema extensions [8, 9]. Also this can be applied in modelling data architectures under MDA initiatives, such as large-scale EIS, XML data/document warehouses, web and e-commerce systems. In this paper, we continue our discussion on our view formalism with emphasis on a semantic modelling methodology. We also provide some potential industrial applications where such view formalism has been utilized.

The rest of this paper is organized as follows. In section II we present some of the early work done in the view related domains followed by a discussion on our layered view model in section III. In section IV we introduce the XSemantic nets based view design methodology. Section V presents our work on conceptual operators, while section VI presents some of the real-world e-Solutions and applications of our view formalism. Section VII concludes the paper with some discussion on our future research directions.

II. Related Work

The discussion on view formalism for relational and OO paradigm for industrial applications have been extensively discussed in many forms [10-12]. Here we look at views for semi-structured data.

To solve the challenges associated with views for semi-structured data, many researchers have attempted to solve it by using; (a) graph based [13] and (b) semi-structured data models [14-16]. But, in their proposals, similar to that of relational and OO view models, they kept the view definitions at the lower levels abstraction. One of the early discussions on XML view was by Serge Abiteboul [17] and later more formally by Sophie Cluet et al. [18]. 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 [17]. 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 (e.g. ODMG [19] model) and a query language. In [18], authors discuss in detail on how abstract paths/DTDs are mapped to concrete paths/DTDs. These concepts, which are implemented in the Xyleme project [6, 20], 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. However, in relation to conceptual modeling, these view concepts provide no support. The view formalism 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.

The MIX (Mediation of Information using XML) view system [21] is a by-product of developing web scale mediator systems. Though MIX system provides support for XML views, it is not a formal view formalism as such. 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 it doesn't utilize XML Schema a semantically rich replacement for XML DTD and though it handle valid XML documents as input, it does not systematically validate resulting view documents [16]. As result, MIX systems just provide support for some of the XML view concepts and not completely satisfy requirements for an XML view model.

Object-Relationship-Attribute model for Semi-Structured data (ORA-SS) [16, 22] is an intuitive data model for XML based on Entity-Relationship (E-R) model and the static OO model. An object in ORA-SS is similar to that of an entity in E-R (and similar to that of an XML element), while a relationship is similar to that of a relationship between two entities in E-R. Attributes of ORA-SS describe the objects and relationships. This is one of the few view models that provide some form of abstraction to define and model views for XML.

In related work in Semantic Web (SW) [23] paradigm, some work has been done in views for SW [24, 25], where the authors proposed a view formalism for RDF document with support for RDF [26] schema (using a RDF schema supported query language called RQL). This is one of the early works focused purely on RDF/SW paradigm and has sufficient support for logical modeling of RDF views. The extension of this work (and other related projects) can be found at [27]. RDF is an object-attribute-value triple, where it implies object has an attribute with a value [28]. It only makes intentional semantics and not data modeling semantics. Therefore, unlike views for XML, views for such RDF (both logical and concrete) have no tangible scope outside its domain. In related area of research, the authors of the work propose a logical view formalism for ontology [29-31] with limited support for conceptual extensions, where materialized ontology views are derived from conceptual/abstract view extensions.

III. A View Model For XML

Our view model for XML 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.

A. 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 [28, 32, 33], or our developed XML-specific XSemantic net [8, 34], 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** $\boldsymbol{V^c}$ 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. $\boldsymbol{V^c} = (V^c_{name}, V^c_{obj}, V^c_{rel}, V^c_{constraint})$ 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 $\exists o_1, ..., o_n \in C_{obj}$, 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. [8, 33].
- 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}.

B. Logical/Schema Level

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 [28, 34-36]. 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 $\boldsymbol{V^c} = (V^c_{name}, V^c_{obj}, V^c_{rel}, V^c_{constraint})$, $\boldsymbol{V^s} = (V^s_{name}, V^s_{simpleType}, V^s_{complexType}, V^s_{constraint})$ is a **valid schema view of** $\boldsymbol{V^c}$, if and only if $\boldsymbol{V^s}$ is transformed from $\boldsymbol{V^c}$ by $\aleph^c_s$. That is, $\aleph^c_s : \boldsymbol{V^c} \rightarrow \boldsymbol{V^s}$.

In our previous works [37], 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.

C. Document/Instance Level

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 are mapped to query expressions (e.g. XQuery), which are syntax specific.

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

IV. Semantic Modeling of XML Views

We note that models are often abstract representations that only keep so much of the detail as is relevant to the particular problem being considered [1, 38]. XML Schema generally is too low a representation to permit users to interact, visualize or understand it. In [34], authors considered a modified form of semantic net to capture concepts and relationships in a form with clearly expressed semantics, which we call it here as XML Semantic nets (or XSemantic net). These XSemantic nets are at a higher level of abstraction than the XML Schema and we also define clear rules for transformation from the XSemantic nets to XML Schema, which do this automatically.

To be utilised as an expressive model to capture XML conceptual models, we modified the semantic nets [34] in a manner that allows us to move easily and automatically from and to XML.

The modification includes; (1) since classical semantic nets are usually cyclic, for the purpose of capturing XML structure, we resolve the cyclic links and made XSemantic Nets unidirectional; (2) one node has one and only one parent. Orphans nodes (except root node) are not allowed and (3) constraints (OO, XML) can be defined over nodes and edges to reflect an XML OO conceptual model.

The power of XSemantic nets comes 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 [34]. The XSemantic net notation used in this paper is shown in Fig. 1.

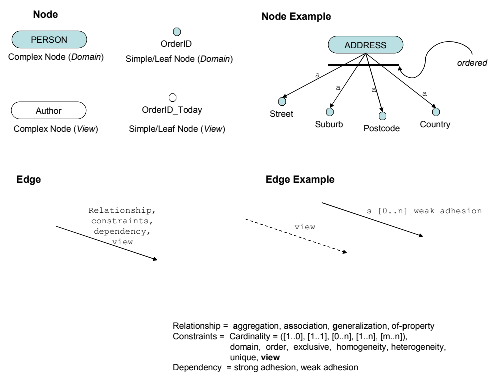
Node
PERSON
Complex Node (Domain)
OrderID
Simple/Leaf Node (Domain)
Author
Complex Node (View)
OrderID_Today
Simple/Leaf Node (View)
Node Example
ADDRESS
ordered
a
Street
Suburb
Postcode
Country
Edge
Relationship, constraints, dependency, view
Edge Example
s [0..n] weak adhesion
view
Relationship = aggregation, association, generalization, of-property
Constraints = Cardinality = ([1..0], [1..1], [0..n], [1..n], [m..n]), domain, order, exclusive, homogeneity, heterogeneity, unique, view
Dependency = strong adhesion, weak adhesion

Figure 1: XML Semantic (XSemantic) net notation

The proposed methodology comprised of three design levels: (1) semantic level, (2) schema level and (3) instance level. The aim is to enforce conceptual modeling power to XML (and views) in order to narrow the gap between real-world objects and XML document structures.

The first level corresponds to the Object-Oriented (OO) conceptual level and composes of two models, namely, the XML domain and the XML view models. This level is based on a modified semantic network[34], referred here as XML Semantic net (or XSemantic net) that provides semantic modeling of XML domains through five major components;

i. a set of *atomic* and *complex* 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 *aggregation, generalization, association*, *of-property* and *view* relationships;
iv. a set of constraints defined over nodes and edges to constrain semantic relationships and object domains and finally,
v. a set of binary and unary conceptual operators to systemically construct conceptual views from a given collection set of nodes and edges.

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 [34] 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 concern with 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 are proposed to transform valid conceptual operators into executable native XML query expressions, such as XQuery [39] FLOWR expressions or SQL 2003/SQLX [40] statements. The resulting query expressions/statements be able to construct imaginary XML documents that can be validated against the XML (view) schemas at the schema level of the design methodology. A detailed discussion on such transformation can be found in our work in [37] and [41].

V. 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* $\tilde{\lambda}$. 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 binary and unary 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 contexts. 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 [33] can be derived from these basic set of operators.

In this paper, to illustrate our concepts, we use the description of a simple Conference publishing system (CPS) for managing, distributing and archiving conference proceedings (as an e-commerce solution) for various International conferences held in different cities throughout the year (Fig. 2). The main component of a conference publication comprises of a collection of papers (past and present), stored in various geographically distributed conference databases/systems, in varying proceedings format such as ACM, LNCS, IEEE etc. The system is similar to that of existing systems such as ACM Portal [42], SpringerLink [43] or IEEE Xplore® [44]. Logically, we treat all the different conferences and their proceedings as one big (logical) conference proceeding on the web (similar to the concept of a "global view" in enterprise systems).

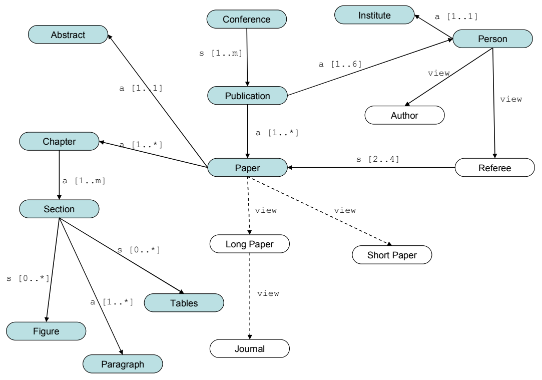

Figure 2: Simplified XSemantic net model of the CPS

A. 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 context objects to produce conceptual views. Here, the context objects are represented in XSemantic nets.

Let x, y be two set of objects, ($x, y \in C_{obj}$; called O_{perand1} and O_{perand2} 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 (Fig. 3). This can be shown as; $\bigcup_{(x,y)} = x \cup y'$, 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 [41].

(3) Difference Operator: (Fig. 4) 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 $dom(\mathcal{R}) = \mathcal{D}(x) \times \mathcal{D}(y)$

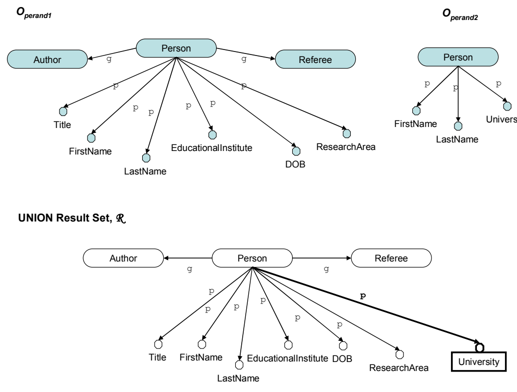

Figure 3: Union operator example

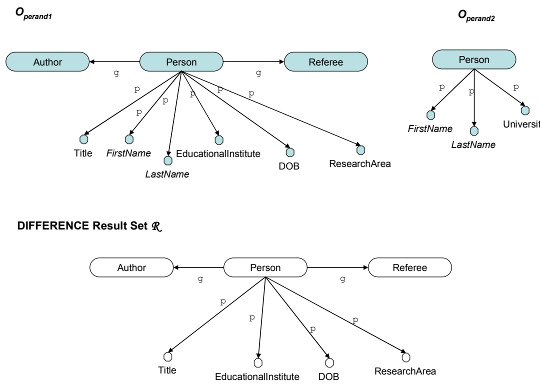

Figure 4: Difference operator example

(5) *Join Operator*: A *Join* operator can be shown in its general form as; $\bowtie_{(x,y)} = \mathcal{R} = x \bowtie_{[j_{condition}]} y$

where, optional join-condition provides meaningful merger of objects in a given context C; where $dom(\mathcal{R}) = \mathcal{D}(x) \times \mathcal{D}(y)$

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 $\rhd\lhd_{(x,y)}$ 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; $\rhd\lhd_{(x,y)} = \mathcal{R} = x \rhd\lhd y = \times_{(x,y)}$

(ii) Conditional Join: (Fig. 5) A join operator $\rhd\lhd_{(x,y)}$ 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; $\rhd\lhd_{(x,y)} = \mathcal{R} = x \rhd\lhd_{(j_{condition1}[AND.....])} y$

(iii) Pattern Join: A join by pattern $\rhd\lhd_{(x,y)}$ is a join by condition operator where the join-condition $j_{condition}$ is of type pattern-condition.

$\rhd\lhd_{(\text{Paper, Refree})} = \text{Paper} \rhd\lhd_{(\text{Paper/PaperID=Referee/PaperID})} \text{Referee}$

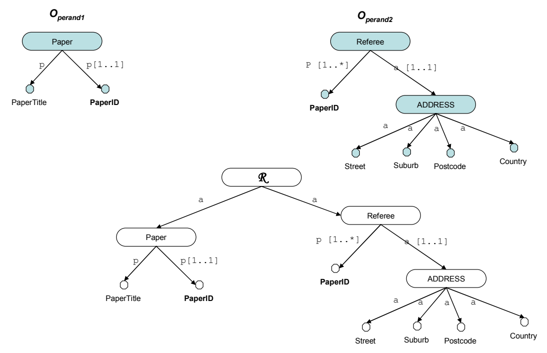

Figure 5: Join operator example

$\prod_{(\textit{Author/FirstName, Author/LastName, Paper/Abstract/*})}(\text{X})$

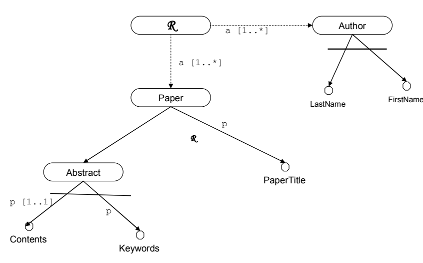

Figure 6: Project operator example

B. Conceptual Unary Operators

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

(1) PROJECT Operator (Fig. 6): 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 [45] 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 [8]; $\sigma_{(x)} = \mathcal{R} = \sigma_{s_{condition}}(x)$

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} 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 [33] for Ontology extraction) can be derived by combining one or more of these binary and unary operators.

VI. SOME POTENTIAL APPLICATIONS OF THE XML VIEW MODEL

Since XML and XML driven solution frameworks are on the increase, it is important to provide models and techniques for XML, which is at a high enough level of abstraction but with rigorously defined standards that are to be more widely understood by both developers and non-technical users. We adopt this ideology in our work with e-solutions designed by our views. Here, we briefly present some of the potential real-world applications of our XML view model in the context of EIS and e-solutions.

A. Views for XML Databases and Repositories

In relational DBMS systems, views are used in the context of access control, query refinement, performance enhancement and providing data perspectives for complex aggregate data. In the Enterprise Content Management (ECM) [46] framework, is a data intensive task, especially when handling large volumes of distributed heterogeneous data, such as data warehousing. Yet, with increasing heterogeneous database schemas (relational, XML and other formats) and contents challenge the traditional database and view techniques. Therefore for XML database and repository designers can develop their platform independent view formalism using higher-level modeling languages and use automated tools to implement such views in their multi-site, multi-platform DB/Repository systems. In doing so, the view definitions are not coupled a specific query syntax and/or specification where they are exposed rapid to changes.

B. XML-View based UAC Middleware Design

The proposed growth for XML repositories in the ECM framework and their use in either to store data or as an interoperability layer for legacy applications provides the need to investigate user access control in such repositories [47]. The widespread use of XML highlights the need for flexible and expressive access control models for XML documents to protect sensitive and valuable information from unauthorized access (both by humans and machines/agents). Traditionally, views provided user access control mechanism in many DBMS. In the work [47], authors present an XML view-based access control model, which supports access control for both human and machine data users. The design methodology proposed is based on XML views [1, 2] and support conceptual level design of UAC constraints and acts as a middleware layer for XML repositories and the databases alike.

C. Document Warehouse Design

Enterprise Content Management (ECM) [46] is a data intensive task, especially when handling large volumes of distributed heterogeneous data, such as data warehousing. To address such an issue, the authors of the work [48], proposed an XML-view based design methodology for modeling and designing XML document warehouses (XDW), using our view formalism (conceptual and logical views) for dimensional (XML) data modeling [48, 49]. Later they extended their work to accommodate web content in Web (XML) document warehouse design [50]. The proposed XDW (and WDW) contain three levels, namely (1) user requirement level [51, 52], (2) XDW conceptual model level, (3) warehouse logical (or schema) level and (4) document (or instance/query) level.

D. Collaborative Web Engineering

The increase in enterprise web content in XML and storage of data in XML document format will provide greater semantic clarity and enable easier access and evaluation of the semantically rich web contents. For example, in an Industrial setting, a web solution may comprise of partner companies/franchise, where they have similar web content but varying user interface and/or web design. In order to keep the web content descriptive among business partners, yet discrete, where a particular user/staff may want to get an appropriate view of data (warehouse storage information, bookings, supplier information, storage capacity etc) at a given location or level of the solution network hierarchy. One way to handle such a complex task is to model and build semantic-aware enterprise websites [53] and web portals [54], using views, where the web content and their associated user interface definitions are captured at the conceptual level (using conceptual views) and mapped to logical view schemas where, additional presentation constraints (such as local company web styles/formats) are applied.

E. Views for Semantic Web (SW) Paradigm

Views for SW [33] requires some form of abstraction [41], as SW documents and querying are done at the logical level or with logical syntaxes to handle heterogeneous schemas such as in multi-site ontology bases. Therefore we argue that a view formalism for SW requires 2-Es (data Extraction and Elaboration) [33]. Though there exists some work in regards to a logical view formalism for SW, most of them are tied to RDF specific schema/syntaxes that do not provide conceptual extensions In related work [33], our view formalism provide support for ontology extraction in the form of materialised ontology views in Ontology Extraction Methodology (OEM) [31, 55]. In the work, authors investigated how our layered view model can be applied to Ontology extraction under the OEM framework.

VII. Conclusion and Future Work

Though very useful, existing view formalisms (for all data models including XML) lack higher level modelling techniques and abstraction that are needed to describe, model, and communicate complex systems such as data warehouse and e-commerce systems. Therefore, in this paper, we presented an informal and formal discussion on an XML view formalism for with conceptual and logical extensions. We also include some of the real-world examples of such views.

For future work, a few of issues deserve investigation. First is the investigation into dynamic aspects of the XML-view formalism. Second a well-formulated empirical study to focus on validating the view formalism.

VIII. References

[1] T. S. Dillon and P. L. Tan, *Object-Oriented Conceptual Modeling*: Prentice Hall, Australia, 1993.
[2] OMG-MDA, "The Architecture of Choice for a Changing World®, MDA Guide Version 1.0.1 (http://www.omg.org/mda/)," OMG, 2003.
[3] Jacek Blazewicz, et al., *Handbook on Data Management in Information Systems*: Springer, Berlin ; New York, 2003.
[4] W3C-XML, "Extensible Markup Language (XML) 1.0, (http://www.w3.org/XML/)," 3 ed: WWW Con. (W3C), 2004.
[5] W3C-XSD, "XML Schema," vol. 2004, 2 ed: W3C, 2004.
[6] Lucie-Xyleme, "Xyleme: A Dynamic Warehouse for XML Data of the Web," Int. IDEAS '01, Grenoble, France, 2001.
[7] J. Pokorn'y, "XML Data Warehouse: Modelling and Querying," Proc. of the Baltic Conf. (BalticDB-IS '02), 2002.
[8] Rajugan R., et al., "XML Views, Part II: Modeling Conceptual Views Using XSemantic Nets," Wrkshp & SS in IEEE IECON '04, S.Korea, 2004.
[9] Rajugan R., E. Chang, T. S. Dillon, and F. Ling, "XML Views: Part 1," 14th Int. Conf. on DEXA '03, Prague, Czech Republic, 2003.
[10] C. J. Date, *Relational database : selected writings*. Reading, Mass.: Addison-Wesley, 1986.
[11] C. J. Date, *An introduction to database systems*, 8th ed. New York: Pearson/Addison Wesley, 2003.
[12] R. Elmasri and S. Navathe, *Fundamentals of database systems*, 4th ed. New York: Pearson/Addison Wesley, 2004.
[13] Y. Zhuge and H. Garcia-Molina, "Graph structured Views and Incremental Maintenance," Proceeding of the 14th IEEE Conf. on ICDE '98, USA, 1998.
[14] S. Abiteboul, et al., "Views for Semistructured Data," Workshop on Management of Semistructured Data, USA, 1997.
[15] H. Liefke et al., "View Maintenance for Hierarchical Semistructured," Proc. of 2nd Int. Conf. on DaWak '00, London, UK, 2000.
[16] Y. B. Chen, T. W. Ling, and M. L. Lee, "Designing Valid XML Views," Proc. of the 21st Int. Conf. on ER '02, Tampere, Finland, 2002.
[17] S. Abiteboul, "On Views and XML," Proc. of the eighteenth ACM SIGMOD-SIGACT-SIGART PODS '99, USA, 1999.
[18] 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.
[19] R. G. G. Cattell, et al., "The Object Data Standard: ODMG 3.0," Morgan Kaufmann, 2000, pp. 300.
[20] Xyleme, "Xyleme Project (http://www.xyleme.com/)," 2001.
[21] C. Baru, A. et al., "XML-based information mediation with MIX," Proc. of the ACM Int. Conf. on SIGMOD '99, United States, 1999.
[22] Y. B. Chen, T. W. Ling, and M.-L. Lee, "A Case Tool for Designing XML Views," Second Int. Workshop on DiWeb '02, Toronto, Canada, 2002.
[23] W3C-SW, "Semantic Web, (http://www.w3.org/2001/sw/)," W3C, 2005.
[24] R. Volz, et al., "Views for light-weight Web ontologies," Proc. of the ACM Symposium on Applied Computing (SAC '03), USA, 2003.
[25] R. Volz, et al., "Implementing Views for Light-Weight Web Ontologies," Seventh Int. IDEAS'03, Hong Kong, SAR, 2003.
[26] W3C-RDF, "Resource Description Framework (RDF), (http://www.w3.org/RDF/)," 3 ed: WWW Con. (W3C), 2004.
[27] KAON, "KAON Project (http://kaon.semanticweb.org/Members/rvo/Folder.2002-08-22.1409/Module.2002-08-22.1426/view)," 2004.
[28] L. Feng, E. Chang, and T. S. Dillon, "Schemata Transformation of Object-Oriented Conceptual Models to XML," *Int. J. of Comp Sys. Science & Engineering*, vol. 18, No. 1, pp. 45-60, 2003.
[29] C. Wouters, et al., "A Practical Walkthrough of the Ontology Derivation Rules,": 13th Int. Conf. DEXA '02, Aix-en-Provence, France, 2002.
[30] C. Wouters, et al., "A Practical Approach to the Derivation of a Materialized Ontology View," in *Web Information Systems,* USA: IGP, 2004.
[31] C. Wouters, et al., "Ontologies on the MOVE," 9th Int. Conf. on DASFAA '04, Jeju Island, Korea, 2004.
[32] Rajugan R., E. Chang, T. S. Dillon, and L. Feng, "XML Views, Part III: Modeling XML Conceptual Views Using UML," 7th Int. Conf. on ICEIS '05, USA, 2005.
[33] C. Wouters, Rajugan R., T. S. Dillon, and J. W. Rahayu, "Ontology Extraction Using Views for Semantic Web," in *Web Semantics and Ontology*, USA: IGP, 2005.
[34] L. Feng, E. Chang, and T. S. Dillon, "A Semantic Network-based Design Methodology for XML Documents," *ACM Trans. on IS (TOIS)*, vol. 20, No 4, pp. 390 - 421, 2002.
[35] R. Xiaou, et al., "Modeling and Transformation of Object-Oriented Conceptual Models into XML Schema," 12th Int. Conf. on DEXA '01, 2001.
[36] R. Xiaou, et al., "Mapping Object Relationships into XML Schema," Proc. of OOPSLA Workshop on Objects, XML and Databases, 2001.
[37] Rajugan R., 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), Klagenfurt, Austria, 2005.
[38] G. Booch, *Object-oriented analysis and design with applications*, 2nd ed. Redwood City, Calif. Reading, Mass.: Benjamin/Cummings Pub. Co.; Addison-Wesley, 1993.
[39] W3C-XQuery, "XQuery 1.0: An XML Query Language," in *XML Query Language (XQuery)*: WWW Con. (W3C), 2004.
[40] ANSI and ISO, "ANSI - SQL 2003," ANSI / ISO 2003.
[41] C. Wouters, Rajugan R., et al., "Ontology Extraction Using Views for Semantic Web," in *Web Semantics and Ontology*, USA: IGP, 2005.
[42] A. Portal, "(http://portal.acm.org/)," ACM, 2005.
[43] Springer, "SpringerLink: http://www.springerlink.com," Springer, 2005.
[44] IEEE, "IEEE Xplore®: http://ieeexplore.ieee.org," Rel 1.8 ed: IEEE, 2004.
[45] W3C-XPath, "XML Path Language (XPath) Version 1.0," in *XML Path Language*, vol. November 1999: WWW Con. (W3C), 1999.
[46] AIIM, "The ECM Association (http://www.aiim.org/index.asp)," AIIM, 2005.
[47] R. Steele, et al., "Design of an XML View Based User Access Control (UAC) Middleware," IEEE Int. Conf. on EEE-05, Hong Kong, 2005.
[48] V. Nassis, Rajugan R., T. S. Dillon, and W. Rahayu, "XML Document Warehouse Design," 6th Int. Conf. on DaWaK '04, Zaragoza, Spain, 2004.
[49] V. Nassis, Rajugan R., 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.
[50] V. Nassis, et al., "A Systematic Design Approach for XML-View Driven Web Document Warehouses," Int. Workshop on UWSI '05, Singapore, 2005.
[51] V. Nassis, R. Rajugan, T. S. Dillon, and J. W. Rahayu, "A Requirement Engineering Approach for Designing XML-View Driven, XML Document Warehouses," The 29th Annual Int.COMPSAC '05, Edinburgh, Scotland, 2005.
[52] V. Nassis, et al.., "Goal-Oriented Requirement Engineering for XML Document Warehouses," in *Processing and Managing Complex Data for Decision Support*,.: Idea Group Publishing, 2005.
[53] Rajugan R., et al., "xWeb: An XML View Based Web Engineering Methodology," Int. Workshop on UWSI '05, Singapore, 2005.
[54] W. Gardner, Rajugan R., E. Chang, and T. S. Dillon, "xPortal: XML View Based Web Portal Design," 17th Int. Conf. on ICSSEA '04, Paris, France, 2004.
[55] C. Wouters, et al., "A Practical Approach to the Derivation of a Materialized Ontology View," in *Web Information Systems,* USA: IGP, 2004.