

Conceptual Design of XML Document Warehouses

Vicky Nassis¹, R. Rajugan², Tharam S. Dillon², and Wenny Rahayu¹

¹Dept. of CS-CE, La Trobe University, Melbourne, Australia
{vnassis, wenny}@cs.latrobe.edu.au

²Faculty of Information Technology, University of Technology, Sydney, Australia
{tharam3, rajugan}@it.uts.edu.au

Abstract. EXtensible Markup Language (XML) has emerged as the dominant standard in describing and exchanging data among heterogeneous data sources. XML with its self-describing hierarchical structure and its associated XML Schema (XSD) provides the flexibility and the manipulative power needed to accommodate complex, disconnected, heterogeneous data. The issue of large volume of data appearing deserves investigating XML Document Warehouses. But due to XML's non-scalar, set-based semi-structured nature, traditional data design models lack the ability to represent XML design level constructs in an abstract and implementation-independent form which are crucial for designing complex domains such as data marts and data warehouses, but also during their operational and maintenance phase. We utilize Object Oriented (OO) concepts to develop a conceptual model for XML Document Warehouses. In this paper we propose a conceptual design formalism to build meaningful XML Document Warehouses (XDW). Our focus includes; (1) conceptually design and build meaningful XML (warehouse) repository (xFACT) using OO concepts in integration with XML Schema constructs, (2) conceptually model and design *virtual dimensions* using *XML conceptual views* [10a] [10b] to satisfy warehouse end-user requirements and (3) use UML *package* diagrams to help logically group and build hierarchical *conceptual views* to enhance semantics and expressiveness of the XDW.

1 Introduction

Data Warehousing (DW) has been an approach adopted for handling large volumes of historical data for detailed analysis and management support. Transactional data in different databases is cleaned, aligned and combined to produce good data warehouses. Since its introduction in 1996, eXtensible Markup Language (XML) has become the *defacto* standard for storing and manipulating self-describing information (meta-data), which creates vocabularies in assisting information exchange between heterogenous data sources over the web [22]. Due to this, there is considerable work to be achieved in order to allow electronic document handling, electronic storage, retrieval and exchange. It is envisaged that XML will also be used for logically encoding documents for many domains. Hence it is likely that a large number of XML documents will populate the would-be repository and several disparate transactional databases.

The concern of managing large amounts of XML document data arises the need to explore the data warehouse approach through the use of XML document marts and XML document warehouses.

Since the introduction of dimensional modeling which, evolves around facts and dimensions, several design techniques have been proposed to capture multidimensional data (MD) at the conceptual level. Ralph Kimball's Star Schema [11] proved most popular, from which well-known conceptual models Snowflake and Starflake were derived. More recent comprehensive data warehouse design models are built using Object-Oriented concepts (structural relationships, Object cubes or data cubes) on the foundation of Star Schema. In [7] [8a] [8b] and [18] two different OO modeling approaches are demonstrated where a data cube is transformed into an OO model integrating class hierarchies. The Object-Relational Star schema (O-R Star) model [9] aims to envisage data models and their object features, focusing on hierarchical dimension presentation, differentiation and their various sorts of embedded hierarchies.

These models both object and relational have a number of drawbacks namely they are; (a) data-oriented without sufficient emphasis or capturing user requirements, (b) extensions of semantically poor relational models (star, snowflake models), (c) original conceptual semantics are lost before building data warehouses as the operational data source is relational, (d) further loss of semantics resulting from oversimplified dimensional modeling, (e) time consuming if additional data semantics are required to satisfy evolving user requirements and (f) complex query design and processing is needed, therefore maintenance is troublesome. When applying these approaches to the design of XML document warehouses, it is important to consider XML's non-scalar, set-based and semi-structured nature. Traditional design models lack the ability to utilise or represent XML design level constructs in a well-defined abstract and implementation-independent form.

One of the early XML data warehouse implementation includes the Xyleme Project [16]. The Xyleme project was successful and it was made into a commercial product in 2002. It has well defined implementation architecture and proven techniques to collect and archive web XML documents into an XML warehouse for further analysis. Another approach by Fankhauser et al. [5] explores some of the changes and challenges of a document centric (such as EDI messages, eBooks) XML warehouse. Both these approaches offer some powerful implementation and architectural design insights into XML warehouses but, coupling them with a well defined conceptual and logical design methodology may help future design of such XML warehouse for large-scale XML systems. For these given reasons, in this paper we propose a conceptual modelling approach to the development of an XML Document Warehouse (XDW).

1.1 Our Work

UML, a widely adopted standard for Object-Oriented (OO) conceptual models is the foundation to build this conceptual model for XML document warehousing, considering that its graphical notation complies with the user(s) and domain expert(s) understanding. Our proposed methodology provides; (a) a conceptual design, using UML,

for the XDW meaningful XML FACT repository (xFACT), (b) conceptually model and design *virtual dimensions* using *XML conceptual views* [10a] [10b] to satisfy warehouse end-user requirements and (c) use UML *package* diagrams to logically group and build hierarchical *conceptual views* to enhance semantics and expressiveness of the XDW.

The main aspects to our methodology are as follows; (1) *User requirements (UR)*: Assist in determining different perspectives of the document warehouse rather than data/documents, (2) *XML Document structure* [13]: Using XML document capability in accommodating and explicitly describing heterogeneous data accompanied with their inter-relationships semantics (unlike flat-relational data), (3) *XML Schema*: Describes, validates and provides semantics for its corresponding instance document (XML document) [15]. Also, it can capture all OO concepts and relationships [4] [6] as well as intuitive XML specific constructs, such as ordering explained in Section 1.2 and (4) *Conceptual Views* [10a][10b]: A *conceptual view* describes how a collection of XML tags relates to the direct use of a domain user at the conceptual/abstract level. A typical XML domain may contain XML documents ranging from few to many thousands of semantically related clusters of XML documents depending on the real world requirement. Only a subset of the XML tags cluster, its specification and their data values (information), which collectively form a conceptual view, may be of interest to the domain user at a point in time.

The rest of the paper is organized as follows; In Section 1.2, we outline some unique XML Schema concepts that are captured and modeled using UML, while Section 1.3 provides a brief description of the example case study used in this paper, of which, we will be gradually building XML warehouse model (using UML). Section 2 is dedicated to in-depth discussion of the XDW conceptual model (with examples), followed by the conclusion in Section 3.

1.2 XML Schema and OO Concepts

A widely used approach for conceptual modeling to developing XML Schemas is the OO conceptual model, frequently expressed in UML [6]. In order to understand the semantics that must be captured in abstract models of XML Data Warehouses, two significant issues are raised (1) ordered composition and (2) homogeneous composition. An illustration of each case is provided in the sections that follow using the proposed mapping techniques in [6] for the transformation of the OO diagrams to XML Schema. The examples are extracted from the complete UML diagram in Figure 5.

1.2.1 Ordered Composition

Consider the following XML Schema fragment:

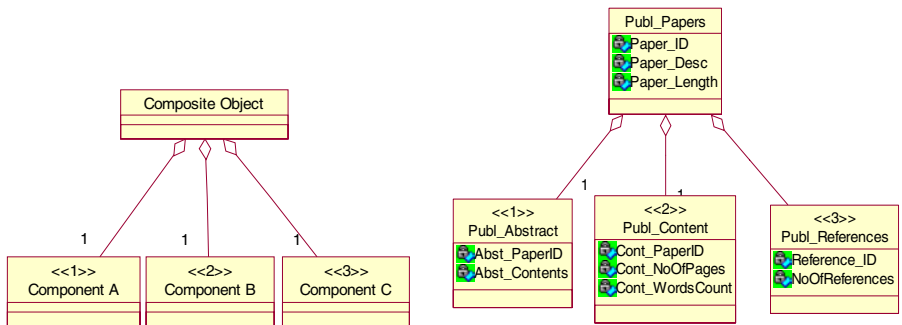
```
<xs:element name="Publ_Papers" type=" Publ_PaperType"/>
<xs:complexType name="Publ_PaperType">
  <xs:sequence>
    <xs:element name="Paper_ID" type="xs:ID"/>
    <xs:element name="Paper_Desc" type="xs:string"/>
    <xs:element name="Paper_Length" type="xs:short"/>
    <xs:element name="Publ_Abstract" type="Publ_AbstractType"/>
  
```

```

        <xs:element name="Publ_Content" type="Publ_ContentType"/>
        <xs:element name="Publ_Reference" type="Publ_ReferenceType"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="Publ_AbstractType">
    <xs:sequence>
        <xs:element name="Abst_PaperID" type="xs:IDREF"/>
        <xs:element name="Abst_Contents" type="xs:string"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="Publ_ContentType">
    <xs:sequence>
        <xs:element name="Cont_PaperID" type="xs:IDREF"/>
        <xs:element name="Cont_NoOfPages" type="xs:short"/>
        <xs:element name="Cont_WordsCount" type="xs:short"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="Publ_ReferenceType">
    <xs:sequence>
        <xs:element name="reference_ID" type="xs:ID" nillable="false"/>
        <xs:element name="NoOfReferences" type="xs:string" nillable="false"/>
    </xs:sequence>
</xs:complexType>

```

The composite element `Publ_Papers` is an aggregation of the sub-elements namely `Publ_Title`, `Publ_Abstract` and `Publ_Content`. Interpreting this XML Schema section we observe that the tag `<xs:sequence>` signifies that the embedded elements are not only a simple assortment of components but these have a specific ordering. Hence we add to UML an annotation that allows capturing of the ordered composition as shown in Figure 1, utilizing stereotypes to specify the objects’ order of occurrence such as `<<1>>`, `<<2>>`, `<<3>>`, ..., `<<n>>`. Figure 1b shows the XML Schema segment above modeled applying this UML notation.



a. UML Stereotype for an Ordered Composition b. The XML Schema Fragment for an Ordered Composition shown in UML

Fig. 1. Figures 1(a) and 1(b): Ordered composition example

1.2.2 Homogeneous Composition

In a homogeneous aggregation, one “whole” object consists of “part” objects, which are of the same type [19]. Two important cases are considered when applied to the homogenous composition, which are as follows:

1.2.2.1 Case of One-to-Many Relationship

Consider the XML schema segment below of the relationship between the two elements `Publ_Chapter` and `Publ_Section`.

```
<xs:element name="Publ_Chapter" type="Publ_ChapterType"/>
<xs:complexType name=" Publ_ChapterType">
  <xs:sequence>
    <xs:element name="Ch_No" type="xs:short"/>
    <xs:element name="Ch_Title" type="xs:string"/>
    <xs:element name="Ch_Keywords" type="xs:string" maxOccurs="unbounded"/>
    <xs:element name="Publ_Section" type="Publ_SectionType" minOccurs="1"
      maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="Publ_SectionType">
  <xs:sequence>
    <xs:element name="Section_Title" type="xs:string"/>
    <xs:element name="Section_Contents" type="xs:byte"/>
    <xs:element name="Section_Figures" type="xs:anyType"/>
    <xs:element name="Section_Tables" type="xs:byte"/>
    <xs:element name="Section_SubSection" type="Publ_SectionType" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

It is declared that the object `Publ_Chapter` can consist of one or more `Publ_Sections`. We specify the **maxOccurs** value of `Publ_Section` to “unbounded” (default value is ‘1’) enabling the element `Publ_Section` to occur from one to many times within `Publ_Chapter`. We consider the assumption that a given section **cannot** be contained in any other chapter from the papers submitted. This characteristic is shown in Figure 2a using the proposed UML notation while Figure 2b shows the model corresponding to the XML Schema fragment below.



a. UML Notation for a Homogeneous Composition featuring One to Many relationship

b. Homogeneous Composition example

Fig. 2. Figures 2(a) and 2(b): Homogeneous composition (Case 1)

1.2.2.2. Case of Many-to- Many Relationships

Consider the XML schema segment below of the relationship between `Publ_Year` and `Publ_Month` elements:

```
<xs:element name="Publ_Year" type="Publ_YearType" maxOccurs="unbounded"/>
<xs:complexType name="Publ_YearType">
  <xs:sequence>
    <xs:element name="Yr_Year"/>
```

```

<xs:element name="Yr_SpecialEvents"/>
<xs:element name="Publ_Month" type="Publ_MonthType" nillable="false"
maxOccurs="unbounded"/>
</xs:sequence>
</xs:complexType>
<xs:complexType name="Publ_MonthType">
<xs:sequence>
<xs:element name="Month_Name"/>
</xs:sequence>
</xs:complexType>

```

In this category a composite object (`Publ_Year`) may have many components (`Publ_Months`) and each component may belong to many composite objects. The `maxOccurs` value equaling to “unbounded” in both elements forms the many-to-many relationship between `Publ_Year` and `Publ_Month` elements. The UML notation illustrating this case is shown in Figure 3a while Figure 3b shows the model corresponding to the XML Schema fragment below.

1.3 An Example Case Study

As a motivating example for this paper, we intent to gradually build a conceptual model of an XDW based on a simplified (academic) conference publication system (CPSys). To illustrate the purpose of this paper we highlight only a few, simplified warehouse user requirements. These requirements include (a) chronological listing of author details, sorted by conferences, (b) alphabetical listing of all proceedings arranged according to the conference date and year. Each listing should include proceedings title, ISBN, conference, editor listing and the table of contents, (c) alphabetical listing of all proceedings together with abstract lists with cross-reference to author/(s) details, (d) chronological listing of conferences and workshops listed with their associated publications, and (e) chronological listing of conferences and workshops listed with their associated abstract list, together with their authors.



a. UML Notation for a Homogeneous Composition featuring Many to Many relationship

b. Homogeneous Composition example (n:m)

Fig. 3. Figures 3(a) and 3(b): Homogeneous composition (Case 2)

Due to page limitations, in this paper, we provide only a very brief discussion on capturing and modeling user requirements, as it can be found in many published literatures in this area such as [21]. In the next section, we outline our proposed XDW conceptual with embedded examples to highlight some important model concepts.

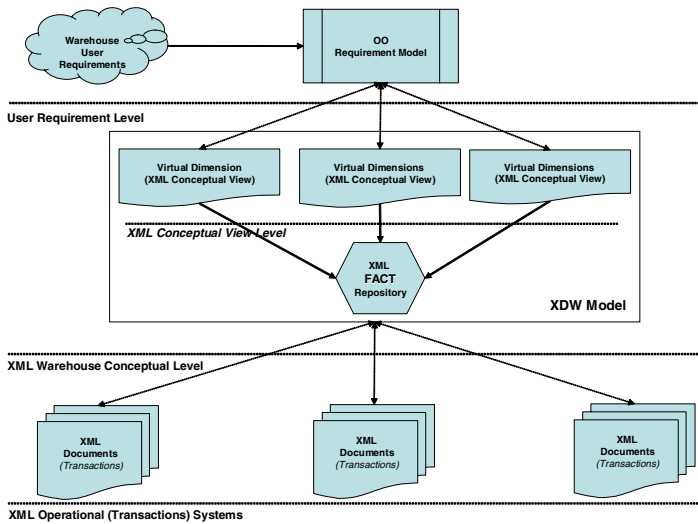


Fig. 4. XDW Context Diagram

2 XML Document Warehouse Model

The XDW model outlined below, to our knowledge, is unique in its kind as it utilizes XML itself (together with XML Schema) to provide; (a) structural constructs, (b) metadata, (c) validity and (d) expressiveness (via refined granularity and class decompositions). The proposed model is composed of three levels (1) User Requirement Level, (2) XML Warehouse Conceptual Level and (3) XML Schema Level. A context diagram of this model is given in Figure 4, below. The first level, the user requirement level composes of two elements; (a) warehouse user requirement document and (b) OO requirement model which includes UML use-case diagrams and descriptions. The second level of the XDW model is composed of the XML FACT repository (xFACT, discussed in Section 2.4 below) and the dimensions that satisfy warehouse user requirements captured. The (XML) Schema level of the model involves transformation of the conceptual model into XML Schema.

2.2 User Requirement Level

The first level of the XDW model captures the warehouse end-user requirements. As opposed to the classical data warehouse models, this requirement model does not consider the transactional data as the focal point in the design of the warehouse. The XDW model is designed based on the user requirements. The transactional data/documents are considered only to refine existing user requirements, if such a need arises. This level is further divided into two more sub-components, which are discussed below.

2.2.1 Warehouse User Requirements

The warehouse user requirements correspond to written, non-technical outline of the XML warehouse. These are usually the typical or the predictable results expected of an XDW. Also, these user requirements can be further refined and/or enhanced by the participation of domain experts or the operators of the transactional system in question. In addition, further refinement can be added to this model by using approaches that generate user requirement documents such as analysing frequent user query patterns [21] or other automated approaches.

2.2.2 OO Requirement Model

The OO requirement model transforms all non-technical user requirements into technical, software model specific concepts using UML (actors, use-case, and objects) to represent. Again, the OO requirement model can be further refined through the involvement of domain experts, system analysts, programmers or the operators of the transactional system.

2.3 XML Warehouse Conceptual Level in UML

The process of deriving the XDW conceptual model involves taking the formalized user requirements expressed in UML and then validating them against the XML transactional system to check for data availability. In the case of no or lack of transactional data highlights ambiguous the user requirements (which has to be re-defined and/or modified) for future warehouse requirements. At a later stage during the transactional system maintenance, these will then be considered as the future user requirements.

The XDW conceptual model is composed of; (1) an XML FACT repository (xFACT) and (2) a collection of logically grouped *conceptual views*. The xFACT is a snapshot of the underlying transactional system/(s) for a given *context*. A *context* is more than a measure [1] [7] or an item that is of interest for the organization as a whole. In classical data warehouse models, a context is normally modeled as an ID packed FACT and associated data perspectives as dimensions. Usually, due to constraints of the relational model, a FACT will be collapsed to a single table, with IDs of its dimension/(s), thus emulating (with combination of one or more dimension/(s)) a data cube (or dimensional data). A complex set of queries is needed to extract information from the FACT-Dimension model. But, in regards to XML, a context is more than a flattened FACT (or simply referred to as *meaningless* FACT) with embedded semantics such as those explained Section 1.2 as well as with non-relational constructs such as set, list, and bag. Therefore, we argue that, a *meaningless* FACT does not provide semantic constructs that are needed to accommodate an XML *context*.

The role of conceptual views is to provide perspectives to the document hierarchy stored in xFACT repository. Since conceptual views can be grouped into logical groups, each group is very similar to that of a *subject area* (or class categories) [2] [20] in Object-Oriented conceptual modeling techniques. Each subject-area in the

XDW model is referred to as *Virtual Dimension* (VDim) to keep in accordance with dimensional models. VDim is called *virtual*; since it is modeled using XML *conceptual views* [10a] (which is an *imaginary* XML document) and behaves as a dimension in the given perspective. The following sections discuss in detail the modeling of VDims, xFACT repository and the issues associated with them.

2.4 XML FACT Repository (xFACT)

XML FACT repository (xFACT) is composed of an XML Schema, which is constructed from the XDW conceptual model and its XML document (formed from the transactional XML document sources). It is a snapshot of the underlying transactional system/(s) for a given *context*. A context is a meaningful collection of classes and relationships, which is designed to provide perspectives (with the help of one or more *conceptual views*) for business intelligence. Due to page limitation, we only presented some fragments (*see* Section 1.2) of the xFACT (XML) Schema, which provides the metadata and validity for the xFACT repository.

2.4.1 Meaningful FACT

In building the xFACT, we vary from modeling traditional data warehouse flat FACT tables such that, we add meaning. That is to say, the xFACT is more semantically descriptive due to its interconnected relationships and decomposed hierarchical structures (Figure 5). The relationships considered in xFACT are not restricted to association (1:1, 1:m, n:m) relationships, but also homogenous composition (with shared aggregation with cardinality 1:n and n:m) specialization/generalization and ordering as discussed in Section 1.2. We utilize XML schema definition language to describe the xFACT semantics (classes, relationships, ordering and constraints). Therefore, modeling of the xFACT is constrained only by the availability of XML schema elements and constructs.

Figure 5 shows the complete model of the xFACT designed for our case study, where the real-world object `Publications` is hierarchically de-composed into `Publ_Papers` and `Publ_Conference`. `Publ_Papers` is further decomposed (with ordering) into `Publ_Abstract`, `Publ_Contents` and `Publ_References`. Such decompositions are necessary to provide granularity to a real-world object and if needed, additional semantics can be added at the design time at different levels of hierarchy. For example, the `Publ_Conference` class hierarchy is decomposed with additional semantics such as `Publ_Region`, `Publ_Country` and `Publ_City`. Another example is the relationship between the parent class `Publ_Person` and the derived sub-classes `Publ_Referee` and `Publ_Author`, which form a generalization relationship (ISA).

Since the flexibility of XML Schema in handling complex structures is greater than any other model, it provides extra intuition to make the xFACT table as expressive as possible. In addition, the process of iterative refinement of user requirements conducted at the design stage helps in modeling such a FACT. Building the xFACT provides the well-defined repository for designing/constructing of expressive VDims.

tional document or data fragment exists to satisfy it. If this does occur, further enhancements are made to the user requirement to make it feasible to model with the existing xFACT. Such situation indicates that there is no transactional document or document fragment to satisfy the end user requisite. Therefore modeling of VDim is an iterative process where user requirements are validated against the xFACT in conjunction with the transactional system.

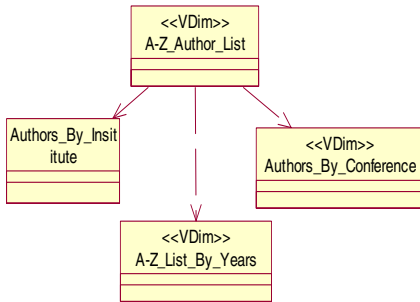


Fig. 6. A conceptual view hierarchy with <<construct>> stereotype

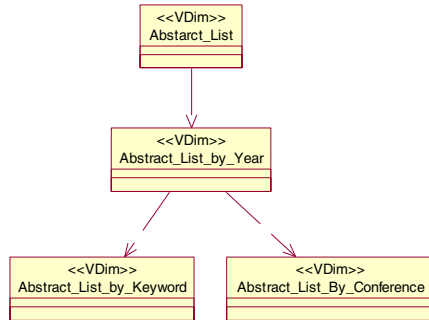


Fig. 7. VDim “Abstract” Package Contents

A VDim is modeled at the XDW conceptual level using a new UML stereotype called <<VDim>>. This stereotype is similar to a UML class notation with a defined set of attribute and methods. The method set can be either constructors (to construct a VDim) or manipulators (to manipulate the VDim attribute set). Also note that we model the relationship between an xFACT and a VDim with a dashed, directed line, denoting the <<construct>> stereotype (shown in Figure 6-7). Though VDims can have additional semantic relationships such as generalization, aggregation, association [10a] [10b], such relationships can be shown using standard UML notations. In addition to this, two VDims can also have <<construct>> relationships depending on dependencies (e.g. Figure 7, between VDim Abstract_List_by_Year and VDim Abstract_List_by_Keyword).

Definition 1: One Virtual Dimension composes of one (or more logically grouped) conceptual view, thus satisfying one (or more logically related) user document warehouse requirement/(s).

2.5.1 UML Packages as Dimensions

Earlier, we stated that semantically related conceptual views could be logically grouped together as grouping classes into a subject area. Further, a new view-hierarchy and/or constructs can be added to include additional semantics for a given user requirement. In the XDW conceptual model, when a collection of similar or related *conceptual views* are logically grouped together, we called it aggregated or grouped *Virtual Dimension* (Figures 6-7), implying that it satisfies one or more logically related user requirement/(s). In addition, we can also construct additional con-

ceptual view hierarchies such as shown in Figures 6-7. These hierarchies may form additional structural or dependency relationships with existing conceptual views or view hierarchies (grouped VDims) as shown in Figures 8-9. Thus it is possible that a cluster of dimensional hierarchy/(ies) can be used to model a certain set of user requirement/(s). Therefore we argue that, this aggregate aspect can give us enough abstraction and flexibility to design a user-centered XDW model.

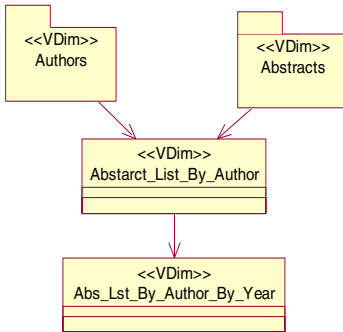


Fig. 8. A VDim hierarchy (derived from grouped VDims)

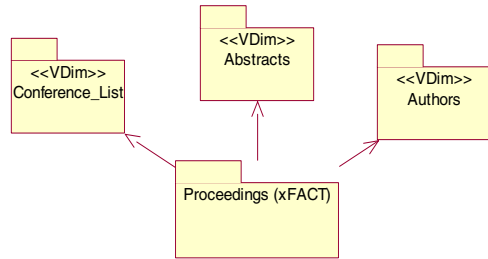


Fig. 9. “Authors”, “Abstracts” & “Conference_List” package

In order to model an XML conceptual view hierarchy or VDim and capture the logical grouping in among them, we utilize the *package* construct in UML. According to OMG specification;

“A *package* is a grouping of model elements. Packages themselves may be nested within other packages. A package may contain subordinate packages as well as other kinds of model elements. All kinds of UML model elements can be organized into packages.” [12].

This in practice describes our logical grouping of XML conceptual views and their hierarchies. Thus we utilize packages to model our connected dimensions (Figure 8). Following the similar arguments above, we can show that, the xFACT (shown in Figure 5) can be grouped into one logical construct and can be shown in UML as one package. In Figures 9 & 10, we show our case study XDW model with xFACT and VDims connected via <<construct>> stereotype.

In summary, we gradually introduced the XDW conceptual model in UML and fragments from sections 2.3 – 2.5. First the complete xFACT model is shown in Figure 5 while the xFACT stereotype of ordering is shown in Figure 1. The xFACT semantic relationships are shown in Figures 2, 3 & 4. Then we introduced virtual dimensions (VDim, shown in figures 7,8 & 9) with the <<construct>> stereotype. Later the complete XDW model (similar to that of the Star Schema for relational model) is shown in figures 9, & 10 using the UML *package* construct. Please note that, due to space limitations, the attributes set of the VDim and that of some xFACT classes are not shown. Also, in the xFACT class hierarchy, the cardinality of the composition and the association relationships should be treated as 1, unless specified.

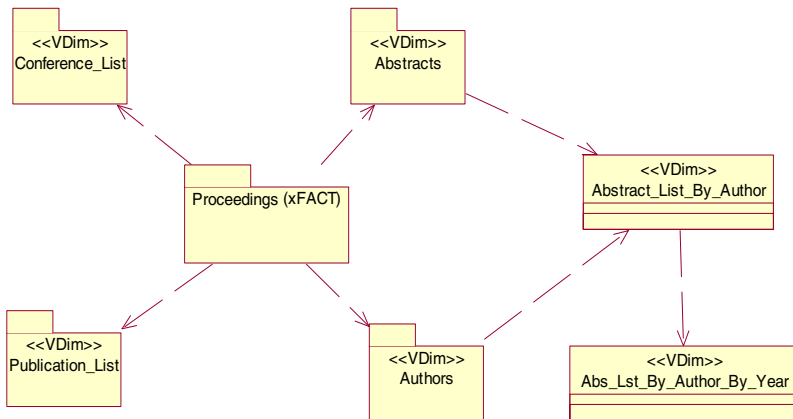


Fig. 10. XDW Conceptual Model (in UML)

3 Conclusion and Future Work

XML has become an increasingly important data format for storing structured and semi-structured text intended for dissemination and ultimate publication in a variety of media. It is a markup language that supports user-defined tags and encourages the separation of document content from presentation. In this article, we present a coherent way to integrate a conceptual design methodology to build a native XML document warehouse. The proposed XML design methodology consists of user requirement level, warehouse conceptual level and XML Schema level, which capture warehouse user requirements and the warehouse model respectively. The only model we did not discuss here in detail is the OO user requirement model, as many literatures exist for that purpose.

For future work, a lot of issues deserve investigation. The first subject matter is the mapping of the conceptual model in XML Schema level. This includes converting the xFACT and VDim into XML Schema using the transformations discussed in the papers [6] and [23]. In the case of VDims, this is actually the transformation of between the *conceptual views* into *XML View* [10a] [10b] schemas. Other steps in our future work include; (a) the techniques to map operational XML documents to xFACT (the equivalent of the classical data warehouse Extract-Load-Transform process), (b) access the data warehouse including XML queries for OLAP processing and (c) aggregate functions.

References

1. M Golfarelli, D Maio, S Rizzi; "*The Dimensional Fact Model: A Conceptual Model for Data Warehouses*"; Int. Journal of Cooperative Information Systems (Invited Proceeding), 7-2-3, 1998.
2. T S Dillon, P.L Tan; "*Object-Oriented Conceptual Modeling*"; Prentice Hall, Australia, 1993.

3. Ramez Elmasri, Shamkant B Navathe; "*Fundamentals of Database Systems*"; 3rd Ed, Addison-Wesley '00.
4. Ling Feng, Tharam Dillon & E Chang; "*A Semantic Network Based Design Methodology for XML Documents*"; ACM Trans. on Info. Sys., Volume 20, No. 4, August 2002; pp 390-421.
5. Peter Fankhauser, Thomas Klement; "*XML for Data Warehousing Changes & Challenges*"; Proc of DaWaK 2003 Prague, Sept, pp 1-3.
6. Ling Feng, Elizabeth Chang, Tharam Dillon; "*Schemata Transformation of Object-Oriented Conceptual Models to XML*"; Int. Jou. of Com. Sys Sci & Eng., Vol 18 No. 1 Jan 2003. pp. 45-60.
7. Juan Trujillo, Manuel Palomar, Jaime Gomez, Il-Yeol Song; "*Designing Data Warehouses with OO Conceptual Models*"; "Computer", IEEE Computer Society (12): 66-75. December 2001.
- 8a. Sergio Lujan-Mora, Juan Trujillo, Il-Yeol Song; "*Multidimensional Modeling with UML Package Diagrams*"; ER 2002: pp 199-213.
- 8b. Sergio Lujan-Mora, Juan Trujillo, Il-Yeol Song; "*Extending the UML for Multidimensional Modeling*"; UML 2002: pp 290-304.
9. Rahayu, J.W., Dillon, T.S., Mohammad, S., and Taniar, D., "*Object-Relational Star Schemas*", Proc. of the 13th IASTED Int. Conf. Parallel and Distributed Comp. & Sys. (PDCS 2001), 2001.
- 10a. R Rajugan, E Chang Thram S Dillon & Ling Feng; "*XML Views: Part I*"; 14th Int. Conf. on DB & Expert Systems Applications (DEXA 2003), Prague, Czech Republic, September 1-5, 2003.
- 10b. R Rajugan, E Chang Thram S Dillon & Ling Feng; "*XML Views, Part II: Modeling Conceptual Views Using XMSEmantic Nets*"; (to be submitted).
11. Ralph Kimbal, Margy Ross; "*The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling*"; 2nd Edition, Wiley Computer Publishing, 2002.
12. OMG; "*Unified Modeling Language Specification*", March 2003, Ver 1.5 formal/03-03-01; <http://www.omg.org/technology/documents/formal/uml.htm>
13. W3C Consortium; "*EXtensible Markup Languagees*"; <http://www.w3.org/XML/>; 2000.
14. W3C Consortium; "*XML Query (XQuery) Requirements*"; <http://www.w3.org/TR/xquery-requirements/>; Nov 2003.
15. W3C Consortium; "*XML Schema*"; <http://www.w3c.org/XML/Schema>; 2 May 2001.
16. Xyleme Project; Publications; <http://www.xyleme.com/>
17. OMG; "Data Warehousing, CWMTM and MOFTM Resource Page" <http://www.omg.org/cwm/>
18. Alberto Abelló, Josè Samos, and Fèlix Saltor "*Understanding Facts in a Multidimensional Object-Oriented Model*"; 4th Int. Wrks.on DW and OLAP (DOLAP '01) Atlanta, Nov '01, pp 32-39.
19. Rahayu, W. S., E. S. Chang, et al.; "*Aggregation versus Association in Object Modeling and Databases*"; ACS Conf. on IS, Australian Computer Society Inc. 2: pp 521 - 532.
20. Coad P., Yourdon, E. "*Object-oriented Analysis*"; Prentice Hall, 1991.
21. Ji Zhang, Tok Wang Ling, Robert M. Bruckner & A Min Tjoa; "*Building XML Data Warehouse Based on Frequent Patterns in User Queries*"; Proc of DaWaK 2003 Prague, Sept. '03; pp 99-108.
22. Jaroslav Pokorny; "*XML Data Warehouse: Modelling and Querying*"; Kluwer Academic Publishers, The Netherlands, 2002, pp 67-80.
23. Renguo Xiaou, Tharam S. Dillon, Elizabeth Chang, Ling Feng; "*Modeling and Transformation of Object-Oriented Conceptual Models into XML Schema*"; DEXA 2001, pp 795-804.