Studying the Evolution of XML Document Structures

Ivan Sun
School of Computer Science
and Information Technology
RMIT University
GPO Box 2476
Melbourne VIC 3001
Australia
isun@cs.rmit.edu.au

James A. Thom
School of Computer Science
and Information Technology
RMIT University
GPO Box 2476
Melbourne VIC 3001
Australia
jat@cs.rmit.edu.au

Abstract

The structure of XML (eXtensible Markup Language) documents often evolve over time, leading to reformulation and new version releases of the document structure. This occurs independently of the different ways in which XML document structure can be expressed. Major structural changes can cause version incompatibility issues and ensuing resource costs of updating existing documents. Software applications may be required to accommodate to both the old and new document structures. We present the case here for a study on the development process and evolution of XML document structures.

Keywords XML, XML Schema, DTD, Document Management

1 Introduction

Since the XML Recommendation [2] was released in 1998 as a simplified SGML (Standard Generalized Markup Language), it has enjoyed widespread usage as a standard for encoding and structuring documents. Many markup languages have emerged in the fields of industry, academia, government and science. Even the latest version of HTML, XHTML, sees it reformulated with XML rather than SGML.

Certainly one of the chief attractions of XML is its simplicity and flexibility of usage. To create a markup language, it is as simple (or so it would seem) as producing a Document Type Definition (DTD) according to the XML Recommendation. For example, the XHTML DTD is a *schema* for XHTML documents, where a schema refers to an expression of the document structure. As an alternative to the DTD, it is possible to use one of the other *schema types*, which are other ways of expressing XML document structure. These alternatives include XML Schema [6] and RELAX NG [4].

Perhaps because it is so apparently simple to create a markup language, and because there is not a widely recognised methodology for designing XML documents, there is a proliferation of markup languages, of which there are few mechanisms for controlling the quality of design. The worst fate that an ill-designed markup language may suffer is being ignored by the community of users.

However, poor design of document structure also leads to another undesirable outcome. Document structures often evolve over time, resulting in new version releases. Sometimes this is from unavoidable changes to the specifications controlling the usage of the XML documents, or changes to the environment in which the XML documents are used. Poor initial design may result in substantial revisions of the document structure with each version release, instead of minor revisions as the document structure evolves. Frequent major changes to document structure result in significant costs. There are version incompatibility issues and ensuing resource costs of updating existing documents. Software applications may be required to accommodate to both the old and new document structures.

Section 2 compares the DTD to another schema type and discusses how XML document design principles may apply across different schema types. Section 3 looks at different methodologies in the current literature for designing XML schema. In Section 4 we provide a case study of the 'XML Encoding for SMS Messages'. This derives from Koponen *et al.* [10], an IETF (Internet Engineering Task Force) draft for comment. It is an example of the evolution of XML document structure in its early stages. Finally in Section 5 we conclude the arguments for the benefits of further study in this area.

Proceedings of the 7th Australasian Document Computing Symposium,
Sydney, Australia, December 16, 2002.

```
<?xml version="1.0"
    encoding="ISO-8859-1"?>
<!DOCTYPE message SYSTEM
    "SMSmessage.dtd">
<message cid="1" id="000001">
  <ack status="ok"></ack>
</message>
```

Figure 1: XML document from Koponen *et al.* [10]

```
<!ELEMENT message  ack, to?,  from?)>
<!ATTLIST message cid CDATA #IMPLIED>
<!ATTLIST message id  CDATA #IMPLIED>
<!ELEMENT ack (said?,from?)>
<!ATTLIST ack status
    (in_progress|ok|error) #REQUIRED>
<!ELEMENT said (#PCDATA)>
<!ELEMENT from (#PCDATA)>
```

Figure 2: Simplified extract from the SMS Message DTD from Koponen *et al.* [10]

2 Background

The XML Recommendation [2] defines a class of data objects called *XML documents*. The distinguishing feature of these documents is the use of named tags (for example, `<year>2002</year>`) as markup that conveys information about content of the element `year`. The start tag `<year>` also has a corresponding end tag `</year>` and may contain text data or other element(s).

Also defined in the XML Recommendation are the conditions for which a document may be *well-formed* or *valid*. In summary, when a document is well formed, it means that the document obeys the XML specification including the proper nesting of markup tags and the appropriate use of characters within the document. When a document is well-formed and valid, it means that as well as being well-formed, the document conforms to a Document Type Definition (DTD), which is a prescription of the structure that a document must adhere to.

2.1 DTDs and other schema types

A DTD prescribes the elements, attributes and entities that XML documents must conform to. The DTD may also be embedded within the XML document.

Figure 1 is an example of an XML document that can be validated by the DTD in Figure 2. These examples originate from the markup language for SMS Messages that is a case study in Section 4 of this paper.

The DTD is the only schema type specified in the XML Recommendation [2]. However it has several limitations as a means of defining document structure. First, elements have limited expression of cardinality. An element may occur once, or any one of the following three cardinalities: one or none (?), none to many (*), and one to many (+). There is no straightforward means of expressing certain integer occurrences, such as 2 to 6 occurring elements, within a single element declaration. Second, there is no data type constraint on the data that may be held by each element in the structure.

In XML Schema, a number of predefined data types are supported. In terms of cardinality of elements, specific minimum and maximum integer values are also supported. Schemas are expressed as well-formed XML documents, unlike the DTD. Furthermore hierarchical relationships of elements are expressed via nested elements. However, this results in more verbose declarations and is contrasted to the more compact DTD schema.

From the above comparison of the DTD and XML Schema, it can be seen that due to the different features of each schema type, it is not always possible to have a complete equivalence between two schemas of different types. For complex documents, this is even less likely.

This suggests that in designing XML documents, the intended document structure should be considered separately to what can be expressed in the different schema types. Schemas of these different types are merely textual representations of the intended document structure.

A study of the evolution of XML document structures may lead to the identification of design principles that can be applied to the various schema types. Furthermore, it may also be advantageous to model and to design XML document structure independent of the schema type. The model can then be translated to a schema of the schema type that best suits the application.

3 Existing Methodologies for Designing XML Documents

In this section we review methodologies based on Entity Relationship (ER) modeling, on information modeling via UML, and some other approaches.

3.1 Information Modeling from Database Modeling Theory

Information models, such as relational database information modeling, have well developed principles and processes ensuring good schema design and fault-avoidance. A translation from a relational database schema to an XML schema is a valid strategy of ensuring sound design of XML document structure.

Thom [11] proposes that the structure of XML documents may be derived from information models, such as those used for relational databases.

This is supported by an illustrative example of a conversion from an ER Schema to a DTD. Other information models such as the Relationship Management Methodology, the Unified Modeling Language (UML) and the Object-Oriented Hypermedia Design method (OOHDM) may be similarly extended to derive XML DTDs.

In a similar but more restrictive vein, Fong *et al.* [7] propose a methodology where a DTD is derived from an Extended Entity Relationship (EER) model. Relational data is used to build an EER model, which is then used to derive an XML structure. The purported advantage of this is that the EER model would be a stable, static representation of both XML and relational data.

3.2 Information Modeling via UML

Carlson [3] and Birbeck *et al.* [1] both propose similar methodologies for designing XML documents. There is an initial stage of modeling the information using UML diagrams, followed by a process of translating the Class diagrams into DTDs or XML Schema documents. Such a methodology uses the substantial body of literature in object oriented analysis and design as leverage in the process of designing XML document structure.

3.3 Other ways of designing XML Documents

Other ways of designing XML documents include Erdmann *et al.* [5] which describe an approach of using *ontologies* to derive DTDs to structure XML documents, and to query XML documents using a conceptual level of understanding.

An ontology "provides a formal specification of concepts, their relationship, and other realities of some domain [5]". DTDs are considered limited because they do not have a standard means of providing information about what the markup tags mean, and how they relate to the document structure. Erdmann *et al.* propose a software tool called ONTOBROKER, which outputs a DTD for a class of XML documents given. The input required is ontological information expressed in 'Frame Logic', a formal notation for representing ontology.

Erdmann *et al.* [5] recognise that their methods are not meant to completely replace other ways of modeling XML data. However it does provide an interesting alternative to ensuring a purposeful and strategic design of DTDs for structuring XML documents.

Other authors, such as Harold [8], advocate a more 'intuitive' approach. This involves listing the necessary data information for a particular class of XML documents, before culling unnecessary data components, and translating this to a DTD.

4 Case Study: XML Encoding for SMS Messages

Koponen *et al.* [10] is an IETF draft for comment that specifies a simple protocol for submitting SMS messages to mobile phones and other mobile terminals through the Internet. This draft for comment also provides a DTD for encoding the SMS encoded message to be transferred between the SMS Service Provider (SP) and the SMS Gateway. The SP hosts the SMS service, and provides content for the SMS service. The SMS Gateway acts as a proxy for interfacing with the hardware device (SMS Centre) that submits and receives SMS message via complex telecommunication protocols. The purported advantage of using an XML encoded message for the transfer between the SP and the SMS Gateway is that it replaces five existing protocols for communicating with the SMS Centre.

Koponen *et al.* [10] describe two of the SMS Message DTD versions: draft version 00 [9] and draft version 3 [10]. In the evolution of the SMS Message DTD between the two versions, the change items relating to the `messageID` and the `mref` elements stand out as indicative of a possible XML document design principle.

In version 00, we have the `message` element containing a variety of different child elements representing the different types of SMS messages that may be carried by the XML document:

```
<!ELEMENT message (submit*   |
            deliverstatusreport* |
            deliver*  |
            statusreportrequest* |
            statusreport* |
            delete*  |
            deletereport* |
            ack*  | nack* )>
```

Many of these child elements contain the `messageID` and `mref` elements, which represent character data identifying particular SMS messages sent. The following DTD extract illustrates this.

```
<!ELEMENT submit (da+, oa?, ud?, udh?,
       dcs?, pid?, vp?, timing?
       QoS?, messageID, cost?,
       rsr?)>
<!ELEMENT deliverystatusreport
       (mref, statuslist*)>
<!ELEMENT deliver (oa, da?, ud?, udh?,
       dcs?, pid?, scts?,
       location?, QoS?, mref)>
<!ELEMENT statusreportrequest
       (mref, messageID, da*)>
<!ELEMENT statusreport
       (mref, messageID, statuslist*)>
<!ELEMENT delete (mref, messageID, da*)>
<!ELEMENT deletereport (mref, messageID,
```

```
      deleteOK, deleteNOTOK)>
<!ELEMENT nack
      ((messageID | mref), errormessage)>
```

Here are the contents declared for the `messageID` and `mref` elements:

```
<!ELEMENT messageID (#PCDATA)>
<!ELEMENT mref (#PCDATA)>
```

In version 3, the `messageID` and `mref` elements have been removed from the different child elements of `message`. Instead, the information they represent are captured in the addition of two new attributes `id` and `cid` in the `message` element.

The justification of these changes to the DTD is that the "messageID and mref elements [are] used extensively in version 00"[10]. The above changes are illustrated in the following DTD extract from version 3.

```
<!ELEMENT message ((submit* | deliver* |
      ack* | deliverystatus* |
      statusreportrequest* |
      delete*  | deletereport*),
      to?,  from?)>
<!ATTLIST message cid CDATA #IMPLIED>
<!ATTLIST message id  CDATA #IMPLIED>
```

However an evaluation of whether this evolution represents a move to a better DTD design may include further consideration of whether:

- `messageID` and `mref` elements are required in all various types of SMS messages,
- the new DTD provides more visual clarity for the end-users, and
- the new DTD represents a closer match to the protocol specification.

The above example from the case study demonstrates the possibility that patterns may exist in the causes of change to XML document structure and that extensive case studies may unearth design principles that have not been previously identified.

5 Conclusion

We propose that further studies into the evolution of XML documents will contribute to:

- the formulation of design principles to inform future design of XML documents;
- a better methodology for evaluating the quality of design of document structure, such as fault categorisation; and
- a standard approach to designing XML documents independent of schema type.

We propose that such studies will incorporate an evaluation and synthesis of methodologies for the design of XML document structure in current literature. In formulating design principles, we advocate case studies involving real-life examples of markup languages characterised by significant evolution of document structure.

References

[1] M. Birbeck, J. Diamond, J. Duckett, O. G. Gudmundsson, P. Kobak, E. Lenz, S. Livingstone, D. Marcus, S. Mohr, N. Ozu, J. Pinnock, K. Visco, A. Watt, K. Williams and Z. Zaev. *Professional XML.* Wrox Press Ltd, Birmingham, UK, second edition, 2001.

[2] T. Bray, J. Paoli, C. M. Sperberg-McQueen and E. Maler (editors). Extensible Markup Language (XML) 1.0 (Second Edition) , October 1998. [Online] Available: `http://www.w3.org/TR/2000/REC-xml-20001006`.

[3] D. Carlson. *Modelling XML Applications with UML - Practical E-business Applications.* Addison-Wesley, New Jersey, USA, 2001.

[4] J. Clark and Makoto (editors). RELAX NG Specification. [Online] Available: `http://www.oasis-open.org/committees/relax-ng/spec-20011203.html`, December 2001.

[5] M. Erdmann and R. Studer. How to structure and access XML documents with ontologies. *Data and Knowledge Engineering*, Volume 36, Number 3, pages 317–335, March 2001.

[6] D. C. Fallside. XML Schema Part 0: Primer. [Online] Available: `http://www.w3.org/TR/xmlschema-0`, May 2001.

[7] J. Fong and F. Pang. Converting relational database into XML document. In *Database and Expert Systems Applications, 2001. Proceedings, 12th International Workshop on*, pages 61–65, September 2001.

[8] E. R. Harold. *XML Bible.* IDG Books Worldwide, Foster City, California, 1999.

[9] J. P. T. Koponen, T. Ikonen and L. Ziegler. XML Encoding for SMS messages. [Online] Work in Progress. Available: `http://www.xml.coverpages.org/draft-koponen-sms-xml-00.txt`, May 2001.

[10] J. P. T. Koponen, T. Ikonen and L. Ziegler. XML Encoding for SMS messages. [Online] Work in Progress. Available: `http://www.ietf.org/internet-drafts/draft-koponen-sms-xml-03.txt`, April 2002.

[11] J. A. Thom. Information models for document engineering. In M. Rossi and K. Siau (editors), *Information Modeling in the New Millennium*, pages 259–267. Idea Group Publishing, 2001.