Keeping Pace with Evolving XML-Based Specifications

Marvin Tan and Angela Goh

School of Computer Engineering,
Nanyang Technological University, Singapore
{ps7726426d,asesgoh}@ntu.edu.sg

Abstract. Standards and specifications are essential to organizations that require exchange of information with other parties. XML is poised to be the definitive language of choice in Internet and Intranet applications and IT specifications are increasingly adopting XML schema. However, many specifications are still evolving. In order to cope with such rapid changes, extensions to the XML Schema Language are proposed. The objective is to ensure compatibility between different versions of standards. A framework which utilizes the XML Schema Language extensions is further proposed.

1 Introduction

In the past, standards and specifications for exchange of information between information systems were defined in proprietary languages and formats. The advent of the Internet has changed this trend. Today, specifications and standards used in information systems, within domains ranging from the financial sector to automobile industries, are increasingly formatted as XML documents. More significantly, many of these standards are described by XML schemas[10]. In the ideal world, commercial partnerships and alliances remain unchanged and final specifications of standards are indeed "final". In reality, commercial alliances grow and change; partnerships evolve and "final" specifications of a standard often have different versions.

With the ready availability of parsers and XML's extensible nature, XML is arguably the most popular data interchange language today. With the increasing adoption of XML Schema, it will be viewed as a natural way to resolve versioning issues for XML-based specifications. Examples of specifications that utilise XML Schema include W3C (World Wide Web Consortium) standards and OASIS's ebXML[7] for Supply Chain B2B transactions and WfMC's standards[8] in workflow systems. It was estimated in February 2000 that already more than 200 XML-based specifications exists [9]. Many new specifications would have been created since then.

The main problem with standards and interoperability today is the presence of multiple versions and variants of the same standard. In many domains, a universal common standard will never be possible. Hence, it is necessary to cope with changing standards as well as provide some means for similar standards to interoperate. To resolve the need to modify existing systems due to evolving standards, we propose to extend the XML Schema language to include elements and functionalities. Firstly, these extensions will provide a means to highlight the changes and differences between a preceding version or a variant and the original standard. Secondly, the new functionalities will establish

W. Lindner et al. (Eds.): EDBT 2004 Workshops, LNCS 3268, pp. 280–288, 2004.

a common version management definition so that frameworks or middleware may be developed to bridge the differences between different versions and variants of the standards/specifications.

In this paper, a common framework is proposed that allows objects or 'data exchanges' to be converted to a supported version based on the extensions found in them. The proposed extensions and framework will help manage compatibility issues with XML-based standards. The objectives of the work are fundamentally different from the version management solutions, proposed in document management[2] in system specification[3] and in documentation and software management[6]. These version management approaches focuses on the content while this work addresses changes in the meta-data, namely, schema evolution.

The organization of this paper is as follows. The next section will provide an example that highlights the benefits of the proposed extensions. Section 3 contains a description of the proposed extensions while Section 4 will present the proposed framework to aid and enforce the implementation of the proposed extensions. Section 5 discusses the possible issues arising from the use of the extensions. The last section concludes this paper and highlights possible future work.

2 An Example of Evolving Specifications and Standards

2.1 IMS Content Packaging Specifications

Chosen as example due to the rapid evolution and the availability of comprehensive revision history, eLearning standards have undergone tremendous change over the recent years. Take the example of the IMS Content Packaging (IMS CPS) Specifications version 1.1.3 and 1.1.2 [5]. The IMS CPS allows development of eLearning content in any format and to package it up with accompanying metadata so that other compliant systems may reuse the learning content.

As seen in Table 1, the Content Packaging Specifications contain numerous minor changes, such as 'renaming of an element', 'removal of an element' or 'shifting of a subelement from one element to another'. These might not have a major effect on the logical definition of the Schema document. However, implementation-wise, it would mean that the developer has to recode portions of the module, which handle the standards. Some systems may even need to maintain separate modules to ensure backward compatibility with the previous versions. This is a major problem with most standards. Systems adhering to previous versions of standards will not be able to read objects created using the later versions. A software upgrade is required.

3 Proposed Extensions to XML Schema

Most "changes" to standards specifications may be captured formally using a modeling language. However, some changes are either too complex to be modeled clearly or are unimportant. Examples of the former include the 'overhaul' of existing documents and the expression of these elements in a different manner. An example of unimportant changes is the inclusion of comments, which have no technical implications to the

Table 1. Revision History Excerpt- IMS Content Packaging Specifications Ver 1.1.2 [4]

Version No.	Release Date	Comments
0.92	20 March 2000	Format updated with following changes: a) Move "isvisible" attribute from <resource> element to <item> element b) Add <title> to <tableofcontents> c) Revert back to the <resource type="webcontent"> approach introduced in the v0.9 document d) Rename <organization> to <organizations>
Public Draft 1.1	8 Dec. 2000	Made minor text changes and updated document to address the following issues: a) Replaced <tableofcontents> element with the <organization> element. b) Made <title> a sub-element of <item> rather than an attribute of it. c) Changed resource <item> element attribute "identifieref" to "resourceref". d) Made sub-level <manifest> a sub-element of <manifestref>, rather than <manifest>. e) Changed references from URL Base to XML Base. f) Reworded parts of section 2.1 in clarifying the definition of <organizations> and package. g) Added <dependency> element as a sub-element of <resource>. h) Updated XML samples.
Final 1.1	19 April 2001	Updated document to address the following open issues: a) Clarified the use of the <organization> and <item> elements. b) Added statement of recommendation to use PKZip v2.04g as the default Package Interchange File format in Section 1.2. c) Extended meta-data functionality to <organization>, <item>, and <file>. d) Changed the type attribute on <organization> to structure with a default value of hierarchical. e) Changed the href attribute on the <resource> element from Mandatory to Optional.

processing of the XML document. The updates or changes are formally declared through a language like XML Schema. These annotations are incorporated manually within the schema to reflect changes that have taken place. In other words, the revision history as illustrated in Table 1 are embedded in the schema.

3.1 Categorization of Changes

The possible changes that may occur between revisions of standards are categorized in Table 2.

Table 2. Three categories of version-to-version changes

Migratory Changes	Structural Changes	Sedentary Changes
• Morphing of an element to an attribute. • Migration of a sub-element from one element to another. • Modification of an attribute or element.	• Addition of new elements, sub-elements and attributes. • Removal of elements, sub-elements and attributes.	• Renaming of elements. • Renaming of attributes • Change of simple data type.

Migratory changes deal with the movement of elements or attributes to some other part of the document. This means that the elements or attributes existed in the previous version and are physically relocated within the current version. These changes also include the transformation of elements to attributes and vice versa. Such changes could require considerable modifications to be made to the original program reading the document, since the retrieval of attributes is fundamentally different from that of the elements.

Structural changes affect the document through the addition or removal of attributes or elements. Such changes mean that implementation of existing systems will have to be modified to support the new elements or developers must ensure that the removed elements will not have an impact on their system.

Sedentary changes, as the name suggests, involve no movement and have no effect on the structure of the XML standards document. It refers to the possible renaming of elements and attributes. Such changes can also be semantic in nature as the change in the attribute or element name might alter the original meaning completely.

One category of change that is difficult to model is **"semantic change"**. This category may not affect the structure of the schema at all. Such changes involve the change in the interpretation or the meaning of a term or terms used within the specification. For example, in a preceding version, the term **"entity"** in a "legal markup specification" may refer to businesses and companies and the recent version could expand the scope of **"entity"** to include people. Sometimes, semantic changes affect the transition between two versions of a specification or standard and sometimes they do not. There is no foolproof method of representing semantic changes convincingly with the aid of XML and XML Schema. However, the evolution of the Semantic Web [1] will greatly improve the representation of semantically-enriched information and provide possible means to indicate semantic changes within specifications. Section 5 highlights various means to alleviate the problem of representing semantic changes.

It is assumed that the original XML Schema avoids duplicate element names appearing in different local segments of element definition. As "duplicate names under different parent elements" is permitted in the declaration of an XML Schema, it makes the referencing of elements by their names impossible since they are sharing the same namespace. To work around this, the qualified path of the element may have to be used when referencing any particular element to avoid conflicts.

3.2 Identifying XML Schemas for Specifications

One of the key concepts in the use of these proposed extensions is the identity of the schema. The version and namespaces of the same family of specifications must be clearly specified along with information on the previous versions. There are many ways to do this. One way is to use attributes within the **XSD** tag. Parsers may easily access these attributes in order to determine the information. Subsequently, all new attributes or tags added will carry the 'vm' or 'version-management' namespace. Another namespace, 'pv', must also be declared for the previous version XSD.

3.3 Migratory Changes

In the version 1.1 (public draft) of the IMS Content Packaging Specifications, the element, '<manifest>' was moved out from the '<manifestref>' element to become a sub-element under the '<manifest>' element (nested self reference). This update should be reflected in the manner shown in Fig. 1. The original location in the previous version is stated with the type of the moved item in the previous version. This is to avoid confusion since attributes and elements may be given the same name. This category of changes could also involve modifications to attributes or elements and are more complex. Not

```
<xsd:complexType name="manifestType">
  <xsd:sequence>
    <xsd:element ref="metadata" minOccurs="0" />
    <xsd:element ref="manifest" minOccurs="0" maxOccurs="unbounded"
      vm:movedFrom="pv:manifestreftype" vm:type="sub-element"/>
    ... ...
  </xsd:sequence>
</xsd:complexType>
```

Fig. 1. An Example of the Migration of a Sub-element from One Element to Another

only may the name be altered but the structure and composition of the object in question may change including element occurrences and their use (optional or required).

Other changes in this category involve the transformation from an element to an attribute and vice versa. This entails the 'transformation' of the basic type of a tag (from an element to an attribute or vice versa). While the name of the tag may remain the same, the fundamental properties of the tag have changed. These changes should only involve elements of 'simple' type since the transformation of 'complex' elements or 'mixed content' elements to attributes is complicated and cannot be performed without excessive loss of information. Instead, they may be modeled as a series of removal and modification changes. In addition, during the change from a simple element to an attribute, the data type defined for the element will be lost as attributes may only contain literal string values. These changes may result in incompatibility problems or information loss as the information expressed in one version may not be relevant in another version.

3.4 Structural Changes

Structural changes add to or remove from the overall structure of the Schema. Additions are simply represented with **"vm:newAddition='true"** tags. Removal of elements and attributes will be reflected differently since the removed items will not exist in the new specifications. To cater for this, a new segment of XML elements must be declared in the XML Schema. The definition in Fig. 2 highlights the removal of both elements as

```
<vm:nonVisibleChanges>
  <vm:change type="removal">
    <vm:object type="element" name="pv:InLineBlock"/>
  </vm:change>
  <vm:change type="removal">
    <vm:object type="inner attribute" name="maxOccurs" from="pv:Activities"/>
  </vm:change>
</vm:nonVisibleChanges>
```

Fig. 2. An Example of the Removal of Elements

well as attributes. In this case, the attribute being a standard attribute is identified as an 'inner_attribute' and the parent element is stated. The namespace for the previous version is adopted to distinguish the elements.

3.5 Sedentary Changes

In the transition from IMS CPS Public Draft 0.91 to 0.92 (as seen in Table 1), the element '<organization>' was renamed to '<organizations>'. To reflect the renaming, the declaration will be updated as shown in Fig. 3.

```
<xsd:element name="organizations" type="organizationsType" vm:renamedFrom="pv:organization"/>
```

Fig. 3. An Example of the Renaming of an Element

How can these extensions be used in the real world? There are two possible scenarios. Firstly, there may be a wish to upgrade existing specifications and incorporate these extensions to enable version compatibility. One may start with the most recent version of the XML Schema document and insert the relevant extensions. Previous versions may be updated in an iterative manner. These extensions will not affect the existing data exchanges and applications. The existing compliant systems will simply ignore the extensions. The second scenario deals with the fairly straightforward establishment of new specifications from scratch. When a later version is created, the developers of the standards Schema 'translate' these changes from the previous version into the relevant extensions. Table 3 summarizes the proposed XML extensions.

Table 3. Extensions to XML Schema

Version identification	vm:familyId vm:version vm:previousVersionCompatible vm:previousVersionURI
Migratory Changes	vm:movedFrom
Structural Changes	vm:newAddition vm:nonVisibleChanges vm:changetype
Sedentary Changes	vm:renamedFrom

However, these extensions alone are insufficient unless some form of implementation is provided to support their use. Hence, Section 5 proposes a framework to employ these extensions.

4 Proposed Framework to Utilise Compatibility Extensions

Since these extensions do not employ the standard XML Schema namespace, applications will be unable to read these extensions. Hence, it is necessary to ensure that existing systems make use of the proposed extensions to achieve effective compatibility between versions and variants of standards specifications.

With reference to the elearning example, most learning content is packaged in XML documents (known as learning objects). These learning objects may be available over the Internet or even via electronic mail. As shown in Fig. 4, this proposed framework has the following functions:

- Reads the incoming XML documents and retrieves the relevant schemas (may be a iterative process depending on the number of versions elapsed between the two versions: stated in the XML document and the referenced Schema)
- Reads all the schemas and consolidates the 'compatible' segments of the documents based on the presented extensions.

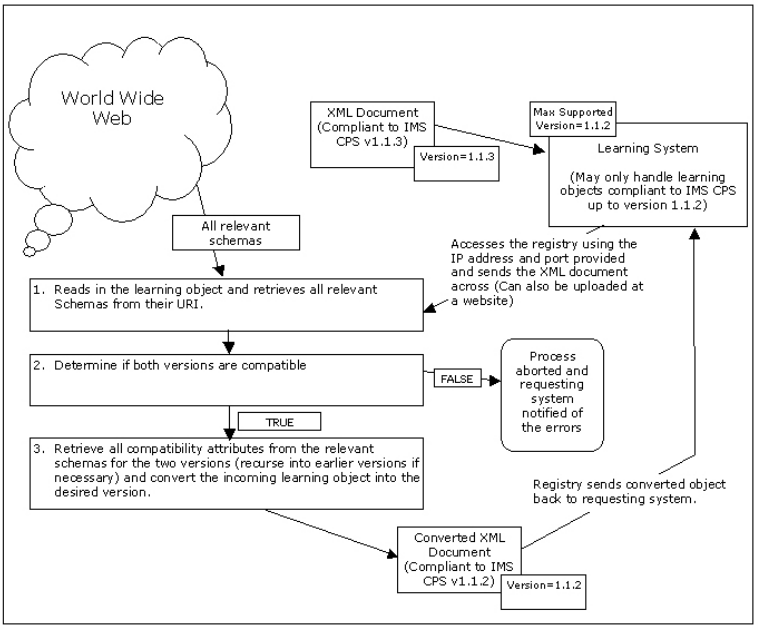

Fig. 4. Overview of Proposed Framework to provide Compatibility in the eLearning Domain

Based on the 'compatibility report' produced in the previous step, convert the document to the desired version and send it back to requesting system. The most feasible way is to provide a generic online registry (preferably by a standards body) that receives standards-based XML documents and instructions on the particular version desired. The registry may then retrieve the relevant schemas, do a check on the extension tags found in both versions and based on the compatibility instructions, convert the received XML document to a version desired by the initiating system. The document is then sent back to the system. This centralized solution may not be feasible in the long run because distributing APIs (Application Programming Interfaces) is a slow and tedious process. However, it is a necessary overhead. It is also necessary to avoid additional implementation work before existing systems can use the framework. With the aid of this framework and the proposed extensions, different versions of standards specifications will be more easily managed at a machine level.

5 Possible Problems and Issues

5.1 Representing Semantic Changes

As mentioned in Section 3.1, 'Semantic changes' cannot be represented in the proposed extensions using XML and XML Schema. However, some effects produced by 'Semantic Changes' may be simulated using the existing extensions. For example, if the meaning of a particular element has completely changed and there is no mapping between that element in a preceding version and the current one (even though the name is unchanged), we could model this effect by using the "removal change" as demonstrated in Fig. 2 and "addition" extension. This series of changes has the effect of removing the element from the current version and adding back an element with the same name. In that way, that particular element will not be treated the same way across the two versions.

5.2 Incompatibility Issues

As mentioned in Section 3.3, there will be problems if segments of different versions are incompatible, resulting in information loss. One scenario is as follows: a given document contains an element of a complex type. However, the number of components within the element has been reduced from 10 to 3. Therefore, there is redundant information provided in the original version. The conversion process will completely remove traces of such information.

Another scenario is the reverse. A later version can contain elements, which are not present in an earlier one. Therefore, the conversion of a document from a later to earlier version will result in the loss of information.

Additional extension tags for such information mismatch are required. To retain information from the preceding version that were removed in the current one, a new element must be included within XML Schema and a new attribute is placed within the element to indicate it as an element to store the incompatible information. The list of incompatible elements can be obtained from the "removal" change information (shown in Fig. 2). The incompatible information will be stored as a block of string type (inclusive of the name tags).

Often it is the lack of information that will cause immediate problems, since the missing information may be critical to the system functions. Though uncommon, this situation may occur during the conversion of an earlier version to a later one. The lack of information may not be resolved at a machine level until full-fledged semantics are available. In addition, backward compatibility is frequently not supported in most upgrades of applications and systems.

5.3 Effects of a Version Management Mechanism

Even though changes to standards and specifications are occurring rapidly, there is still a lapse between two versions of a specification. Changes usually arise from the continual research and analysis performed by the standards working groups. Feedback from usage within a specific domain is also extremely important in improving standards and specifications. With the proposed mechanism, changes can be more easily applied to specifications, leading to a higher adoption of specifications and standards. The time

required to formulate the next version through feedback and analysis is significantly higher than the time required to incorporate the specifications to the application domain.

6 Conclusion

XML-based specifications may evolve relatively quickly over a short period of time. Even though some changes may be minor, systems which exchange information based on these specifications will still have to be modified. These systems may have to carry multiple legacy modules for backward compatibility. Hence, a proper version management framework is necessary to reduce the efforts required to ensure compatibility. Since an increasing number of standards are drafted in the form of XML Schemas, it would be natural to extend the XML Schema to provide constructs for compatibility between different versions of standards documents. The proposed extensions and framework will help to ease compatibility problems as standards and specifications evolve. Future work involves the building of a prototype to prove this concept of version management. The scalability and feasibility of such a solution will also be evaluated.

References

1. T. Berners-Lee. Semantic web road map, 1998. Internal note, World Wide Web Consortium, `http://www.w3.org/DesignIssues/Semantic.html`.
2. S. Chien, V. Tsotras, and C. Zaniolo. A comparative study of version management schemes for xml documents, 2000.
3. Franz Huber, Bernhard Schatz, Alexander Schmidt, and Katharina Spies. Autofocus: A tool for distributed systems specification. In *FTRTFT*, pages 467–470, 1996.
4. IMS Global. *IMS Content Packaging Specification Version 1.1.2 (Final Specification)*. IMS Global, `http://www.imsglobal.org/content/packaging/cpv1p1p2/imscp_infov1p1p2.html`, 2001.
5. IMS Global. *IMS Content Packaging Specifications*. IMS Global, `http://www.imsglobal.org/content/packaging/index.cfm`, 2002.
6. S. Mauw, M.A. Reniers, and T.A.C. Willemse. Message Sequence Charts in the software engineering process. In S.K. Chang, editor, *Handbook of Software Engineering and Knowledge Engineering*. World Scientific, 2000.
7. OASIS. ebxml. `http://www.ebxml.org/`.
8. WFMC. Workflow standards. `http://www.wfmc.org/standards/docs.htm`.
9. Mike Willis. Vertical industry standards: Legal xml and xfrml. In *The DIXON Conference 2000*, 2000.
10. World Wide Web Consortium (W3C), `http://www.w3.org/TR/xmlschema-0/`. *XML Schema part 0: Primer*, 2000.