XML Document Versioning, Revalidation and Constraints*

Jakub Malý and Martin Nečaský

Faculty of Mathematics and Physics
Charles University in Prague, Czech Republic
maly@ksi.mff.cuni.cz

Abstract. One of the prominent characteristics of XML applications is their dynamic nature. When a system grows and evolves, old user requirements change and/or new requirements accumulate. Apart from changes in the interfaces used/provided by the system or its components, it is also necessary to modify the existing documents with each new version, so they are valid against the new specification. In this doctoral work we will extend an existing conceptual modeling approach with the support for multiple versions of the model. Thanks to this extension, it will be possible to detect changes between two versions of a schema and generate revalidation script for the existing data. By adding integrity constraints to the model, it will be able to revalidate changes in semantics besides changes in structure.

Keywords: XML schema, schema evolution, conceptual modeling, constraints.

1 Introduction and Motivation

Recently, XML has become a corner stone of many information systems. It is a de facto standard for data exchange and it is also a popular data model in databases [1]. XML applications are very dynamic in their nature. Requirements change during the life cycle of the system and so do the XML schemas. Without any tools to help, the old and new schema need to be examined by a domain expert. Each change must be identified, analyzed and all the relevant components of the system modified accordingly. Moreover, all the existing documents must be updated. This can be a time-consuming and error-prone process, but, in fact, a significant portion of the operations could be performed automatically.

Applications usually utilize XML in two scenarios: either 1) XML documents are used for data exchange in intra/inter-system communication and XML schemas define interfaces of the individual components and systems themselves, while the data itself are stored in another (usually relational) data storage or 2) XML documents are also used to store the physical data and XML schemas are used to describe the structure and check the validity of these documents.

* This work was supported in part by the Czech Science Foundation (GAČR), grant number P202/10/0573, and by the grant SVV-2011-263312.

A. Harth and N. Koch (Eds.): ICWE 2011 Workshops, LNCS 7059, pp. 317–321, 2011.

As schemas change with a new version of the system, the system needs to be updated, but it is also usually required to accept data valid against the old version, at least for some transitional period of time. In the first scenario, the affected component can be equipped with some *adapter* component that modifies the structure of the document. In the second scenario, the existing documents stored in the system need to be augmented to conform to the new version of the schemas (this process is usually called *revalidation*). In both cases, the problem can be solved by accompanying the new version of the schema with some kind of "revalidation script" every time schemas change and using this script to either preprocess incoming data or update the existing internal documents. The revalidation script can be either a script in an implementation language (XSLT, XQuery Update Facility), or a sequence of formalized update operations.

2 Current Approaches

For the goal of determining whether documents are no longer valid against the new version, the system must recognize and analyze the differences between the old version ($\mathcal{S}'$) and the new version ($\widetilde{\mathcal{S}}'$) of the schema. There are two possible ways to recognize changes:

a) Recording the changes as they are conducted during the design process (and propagate each change immediately to the documents [2,3] or propagate all changes in one batch [4])
b) Comparing the two versions of the diagram [5,6]

All the existing evolution frameworks work only in the scope of one schema. However, in a complex system, the specification can be comprised of hundreds of schemas with interrelated changes. When new data needs to be added to the existing documents (e.g. when new mandatory element is added to the schema) the existing frameworks offer only trivial solutions (creating only the empty structure, the content must be filled by the user). But in a sound and consistent model, the content can be added automatically, as we will outline later. In some schema evolution scenarios, elements, attributes or whole subtrees are moved from one location in the document to another. These so-called *migratory* operations are not supported in many frameworks or the support is insufficient. None of the existing frameworks deal with the semantics of the changes (e.g. when `time-spent` attribute is moved from `Task` element to `Project` element, its value should be equal to the sum of all the values in `Task` elements in the old version) or integrity constraints.

3 Conceptual Modeling with Versioning and Revalidation

This doctoral work will follow the work on conceptual modeling of XML data. In [7], a two-layered model *XSEM* was introduced, with platform-independent

(PIM) and platform-specific (PSM) layers. A PIM schema (UML class diagram) models a problem domain at the conceptual level. A PSM schema is an extended (necessary for modeling hierarchical XML data) UML class diagram that models one XML schema. It is proven that a PSM schema is equivalent to regular tree grammars (RTG). Components of the PSM schemas in the system are linked to concepts in the PIM and thus correctness and coherence can be maintained during initial design and further evolution phase (e.g. change in a concept `Purchase` in the PIM schema can be easily propagated to all the PSM schemas where `Purchase` is referenced) – the two-layered design is made to measure to the scenarios, where there are multiple XML schemas (modeled by PSM schemas) sharing a common problem domain (modeled by PIM schema), each XML schema representing a different view on some part of the domain (e.g a PIM concept `Purchase` is referenced in a `purchase-request` and `yearly-report` schema, both using different attributes and associations of `Purchase`).

Also, having separate PIM and PSM makes possible to add additional models (e.g. model of a relational database) and linked them to PIM too. This way it is possible to depict the relation between XML schema constructs and RDB tables and columns via the links to common PIM.

To date, XSEM was enhanced with support for multiple versions in order to support schema evolution (*XSEM-Evo* [8]). XSEM-Evo uses combination of the methods mentioned in Section 2. The core of the algorithm uses schema comparison, but besides the two versions of the schema, it requires the set of *version links*, which connect the same concept in different versions (e.g. the old version and the new version of the concept `Purchase` will be linked). These version links can be maintained automatically as user edits the schema (this idea comes from the change recording approach), entered manually or by heuristics matching the similar concepts. Combined approach of schema comparison with version links sufficiently handles the addition, removal and also migratory changes in schemas, additional annotations enable it to handle non-trivial migratory operations mentioned in the previous section. The experimental implementation was incorporated into *XCase* editor [9]).

4 Research Objectives and Methodology

The aim of this doctoral work is to further enhance capabilities of XSEM-Evo in two main perspectives: 1) increase the power of XSEM model via introducing constraints at both PIM and PSM layers and 2) fully utilize the links to PIM layer during document revalidation, especially for adding missing content to the revalidated documents.

Constraints in XML UML allows the designer to specify constraints and invariants in the model via *Object Constraint Language* (OCL) in those situations, where classes and associations do not describe the model precisely enough. At the level of XML schemas, constraints are required too. And some types of constraints are impossible to define via languages based on RTGs, such as DTD and classic XML Schema. Examples of such constraints are choices between groups

of attributes or so called co-occurrence constraints (e.g. element E_1 must occur only if the value of element E_2 is v_2 – classic XML Schema cannot do better than to declare E_1 as optional). To allow such constraints, XML Schema was extended with the possibility to declare non-RTG based constructs *assert* and *test* and even a separate schema validation language *Schematron* was designed for this purpose.

As we modified UML to serve us in XML modeling, we plan to modify OCL to serve us to define constraints in XML schemas. Our PSM schemas can be translated to XSDs and it will be possible to translate the PSM level constraints to Schematron schemas analogously.

From the evolution point of view, with OCL constraints, it will be possible to track changes in semantics. For example, the request for customer history returned the list of all purchases in the old version, but in the new version, the list will contain only realized purchases. The structure of the schema will remain unchanged, but in the new version, a new constraint will be added. The evolution algorithm will be able to revalidate the document accordingly via deleting all the unrealized purchases. Since all the existing evolution frameworks only deal with structure and do not recognize semantics, none of them is even capable to detect such change, let alone revalidate it.

Adding content. To date, XSEM-Evo is able to deal with changes that modify the structure and data present in the document. However, sometimes, new data need to be added to the document (e.g. when new mandatory attribute is added to some document). The existing approaches also offer only insufficient solutions. They either only a) create the minimal empty structure (elements and attributes without values) or b) use default values (same in all instances) or c) require the new content to be provided by the user.

Possible link to other models, in particular relational database model (RDM), besides PSM was suggested in Section 3. With RDM linked to PIM, one possible solution (for the first scenario from Section 1) for adding content suggest itself – the required values for the content can be retrieved via a query from the database. E.g. when new attribute `date-of-birth` is added to element `student`, the system can trace the attribute being linked to PIM attribute `date-of-birth` of class `Person` and this can be traced to be stored in a column `PERSON_BIRTHDATE` of table `T_PERSON`. From this table, the value can be retrieved via a query during revalidation.

Another solution for the same problem would be to provide the algorithm with an additional input data document D_i (for the previous example that would be the list of birth dates of the people in the system) and generate the revalidation script so that it will query the document D_i when assigning values for `date-of-birth` attributes. The improvement brought by either of the two solutions is that the revalidation script will again be able to process all the existing documents automatically without requiring user's input.

Both extensions described above will be based on a strictly formal model.

Another possibility of adding data not already present in any form in the system, is by retrieving it from the external sources, e.g. from the Web.

5 Conclusion

Our approach to XML schema evolution and data revalidation can considerably simplify the process of transition to the new version. With the proposed enhancements, XSEM-Evo framework will be able to detect changes in the revalidation schema, decide, whether the detected changes may invalidate existing documents and in that case it generate a revalidation script.

Thanks to the two-layer architecture of XSEM, it is possible to define conceptual changes in one place on the PIM level and let the system to consistently propagate them to all the PSM schemas, where they may have impact. Constraints at both the PIM and PSM levels will complete the structural consistency with proper semantics and consistency of content/values.

The two-layer architecture also enables us to link XML schemas to other components of the system (e.g. relational database). With the introduction of constraints, the ability to detect changes in semantics and provide proper revalidations will be further improved. Together, XSEM framework will facilitate both initial design and further evolution with (special emphasis on consistency and coherence) of the systems, applications and specifications, especially in the area of web engineering, where XML, XML schemas and related technologies are utilized to a large degree.

References

1. Bourret, R.: XML and Databases (September 2005), `http://www.rpbourret.com/xml/XMLAndDatabases.htm`
2. Guerrini, G., Mesiti, M., Sorrenti, M.A.: XML Schema Evolution: Incremental Validation and Efficient Document Adaptation. In: Barbosa, D., Bonifati, A., Bellahsène, Z., Hunt, E., Unland, R. (eds.) XSym 2007. LNCS, vol. 4704, pp. 92–106. Springer, Heidelberg (2007)
3. Su, H., Kramer, D.K., Rundensteiner, E.A.: XEM: XML Evolution Management, Technical Report WPI-CS-TR-02-09 (2002)
4. Klettke, M.: Conceptual xml schema evolution — the codex approach for design and redesign. In: Workshop Proceedings Datenbanksysteme in Business, Technologie und Web (BTW 2007), Aachen, Germany, pp. 53–63 (March 2007)
5. Domínguez, E., Lloret, J., Rubio, Á.L., Zapata, M.A.: Evolving XML Schemas and Documents Using UML Class Diagrams. In: Andersen, K.V., Debenham, J., Wagner, R. (eds.) DEXA 2005. LNCS, vol. 3588, pp. 343–352. Springer, Heidelberg (2005)
6. Kwietniewski, M., Gryz, J., Hazlewood, S., Van Run, P.: Transforming xml documents as schemas evolve. Proc. VLDB Endow. 3, 1577–1580 (2010)
7. Nečaský, M., Mlýnková, I.: When Conceptual Model Meets Grammar: A Formal Approach to Semi-Structured Data Modeling. In: Chen, L., Triantafillou, P., Suel, T. (eds.) WISE 2010. LNCS, vol. 6488, pp. 279–293. Springer, Heidelberg (2010)
8. Malý, J.: XML Schema Evolution Master Thesis (2010), `http://www.jakubmaly.cz/master-thesis.pdf`
9. XCase – tool for XML data modeling, `http://www.ksi.mff.cuni.cz/xcase/`