Data & Knowledge Engineering 72 (2012) 1–30

Contents lists available at SciVerse ScienceDirect

Data & Knowledge Engineering

journal homepage: www.elsevier.com/locate/datak

When conceptual model meets grammar: A dual approach to XML data modeling

Martin Nečaský *, Irena Mlýnková, Jakub Klímek, Jakub Malý

Department of Software Engineering, Faculty of Mathematics and Physics, Charles University, Malostranske nam. 25, 118 00 Praha 1, Czech Republic

ARTICLE INFO

Article history:
Received 8 December 2010
Received in revised form 7 September 2011
Accepted 9 September 2011
Available online 22 September 2011

Keywords:
XML schema
Conceptual modeling
Regular tree grammars
Conceptual to XML schema transformation

ABSTRACT

In this paper we introduce a novel approach to conceptual modeling for XML schemas. Compared to other approaches, it allows for modeling of a whole family of XML schemas related to a particular application domain. It is integrated in a well-established way of software-engineering, namely Model-Driven Development (MDD). It allows software-engineers to naturally model their application domain using a conceptual schema at the platform-independent level of the MDD hierarchy. From there they can design the desired XML schemas in a form of conceptual schemas at the platform-specific level of MDD hierarchy. Schemas at the platform-specific level are then automatically translated to particular XML schemas. Beside this forward-engineering direction, reverse-engineering direction integrating existing XML schemas into the MDD hierarchy is supported as well.

We provide several theoretical results which ensure correctness of the introduced approach. We exploit regular tree grammars to formalize XML schemas. We formalize the bindings between the schemas at the two MDD levels and between schemas at the platform-specific level and XML schemas. We prove that conceptual schemas specify the target XML schemas unambiguously. We also prove the expressive power of the conceptual schemas. And, finally, we prove correctness of the introduced translation algorithms between platform-specific and XML schema levels.

1. Introduction

The eXtensible Markup Language (XML) [18] is currently one of the most popular meta-formats for data exchange on the Web. To enable data exchange, it is crucial to restrict the allowed structure of exchanged XML documents so that each communicating party is able to understand them. The structure is restricted by a set of rules called an *XML schema*. In this paper, we aim at the problem of correct and user-friendly design of XML schemas. At first glance the problem may seem to be solved. There are *grammar-based XML schema languages* (DTD [18], XML Schema [37] or RELAX NG [24]) which enable expressing XML schemas. Formally, they are based on regular tree grammars as described in [61]. Furthermore, languages called *constraint-based XML schema languages* (Schematron [43]) enable specification of integrity constraints. And, if we are not satisfied with their textual notation, we can use tools for XML schema visualization, e.g. *Altova XML Spy* [8].

Research literature shows that XML schema languages are not very friendly and, hence, extends them with conceptual modeling techniques. The idea is similar to the one in the world of relational databases. A designer creates a conceptual schema of the problem domain. It is then automatically converted to an XML schema. Some works extend the UML class model [2] (e.g. [15,25,30,62,75,77]), ER model [22] (e.g. [5,11,50,54,55,72]) or Object-Role Modeling Notation (ORM) [38] (e.g. [56]); others

* Corresponding author. Tel.: +420 221 914 250; fax: +420 221 914 323.
E-mail addresses: necasky@ksi.mff.cuni.cz (M. Nečaský), mlynkova@ksi.mff.cuni.cz (I. Mlýnková), klimek@ksi.mff.cuni.cz (J. Klímek), maly@ksi.mff.cuni.cz (J. Malý).

0169-023X/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.datak.2011.09.002

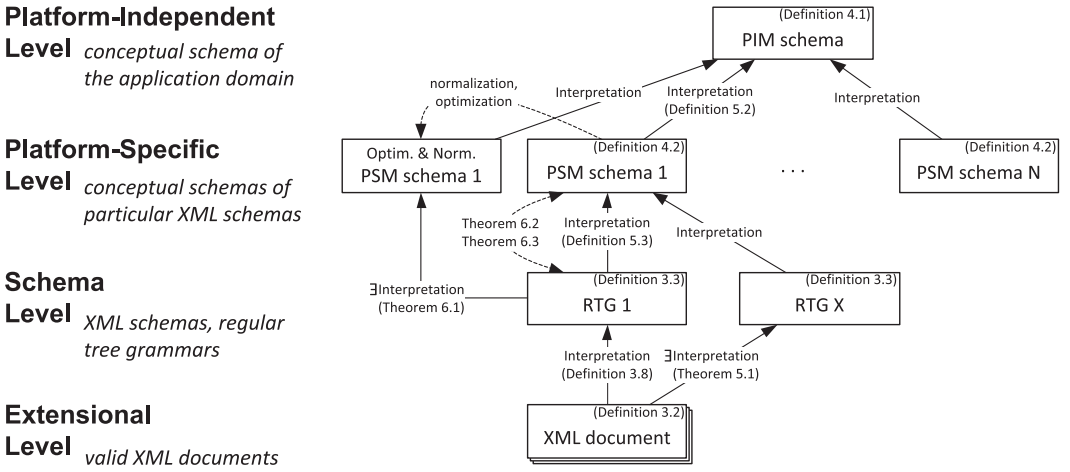

Fig. 1. Schematic visualization of paper contributions.

use ontologies [86](e.g. [13,85]). There are also papers which introduce their own models (e.g. [28,74]). And, naturally, commercial tools exist as well, e.g. *Enterprise Architect* [81]. However, we have identified several drawbacks of these approaches:

1. They require a separate conceptual schema for each single XML schema. In most cases, however, a single XML schema is not suitable for all components of a particular software system. It is necessary to design a whole family of XML schemas. In these cases, a designer needs to create many conceptual schemas not explicitly related to each other.
2. They introduce a set of rules which define an XML schema modeled by a conceptual schema. However, they do not prove that each conceptual schema always models an XML schema. They also do not prove that an XML schema is specified unambiguously. If this is ensured, we say that the *conceptual model is defined correctly*. Finally, they do not prove the *expressive power* of their introduced conceptual model.
3. They do not prove that their algorithm for translating a conceptual schema to an XML schema follows the introduced rules. When this is proved, we say that the *translation algorithm is defined correctly*. Moreover, they focus solely on a single translation direction, but they do not show whether it can be reversed. In that case, it is also necessary to show that the composition of both translation directions always leads to the original XML schema. When this is ensured, we say that *both directions are mutually consistent*.
4. They do not sufficiently consider how designers work when they create conceptual or XML schemas. A designer may introduce unreachable components into the schemas (i.e. components which cannot be instantiated in XML documents). Since some parts maybe unnecessarily complex, a technique which prevents from these situations would make the schemas simpler and more readable.
5. They do not sufficiently study the problem of designing integrity constraints for XML.

To overcome the first of the mentioned drawbacks, we have already introduced a conceptual model for XML called *XSEM* [64,65,70]. It follows the principles of the Model-Driven Development (MDD) [57]. MDD is a well-established software-engineering methodology and, therefore, most software engineers are familiar with it. First, a designer creates a conceptual schema in a platform-independent model (called *PIM schema*). Second, (s)he designs a schema in a platform-specific model (called *PSM schema*) for each desired XML schema on the basis of the PIM schema. It specifies how a selected part of the PIM schema is represented in the target XML schema and how it is automatically converted to an expression in a selected XML schema language.

In this work, we extend our approach with a complete formal model. On its basis we focus on the second, third and fourth mentioned drawback. (Note that we do not focus on the last one. We leave it for our future work.) We call our approach a *dual approach*, because it formally interconnects conceptual and schema level by applying the MDD principles. This provides a possibility to switch from one level to the other in both directions. As a consequence, it also naturally allows us to apply the well-known MDD methodologies called *forward-engineering* and *reverse-engineering* in the process of designing XML schemas. The results presented in this paper are fully implemented in a new tool *eXolutio*.[1]

Particular contributions of this work are outlined in Fig. 1. It shows the two MDD levels — platform-independent and platform-specific. The former one contains a single PIM schema. The latter one contains a PSM schema for each desired XML schema. They represent the conceptual level. The figure also depicts the logical level. It consists of a schema level with XML schemas and

[1] http://www.eXolutio.com.