# Towards Model-Driven Software Language Modernization*

Patrick Neubauer

Business Informatics Group,
TU Wien, Austria
neubauer@big.tuwien.ac.at
http://www.big.tuwien.ac.at

**Abstract.** The introduction of Extensible Markup Language (XML) represented a tremendous leap towards the design of Domain-Specific Languages (DSLs). Although XML-based languages are well adopted and flexible, their generic editors miss modern DSL editor functionality. Additionally, artifacts defined with such languages lack comprehensibility and, therefore, maintainability, because XML has primarily been designed as a machine-processable format with immutable concrete syntax. While there exist techniques to migrate XML-based languages to modeling languages, they are composed of manual steps demanding complex language engineering skills that are usually not part of a domain engineer's skill set. To tackle these shortcomings, we propose a bridge between XML-based languages and text-based modeling languages. This includes the automated and customizable generation of Xtext-based editors from XML schema definitions (XSDs) providing advanced editor functionality, individualized textual concrete syntax style, and round-trip transformations enabling the exchange of data between the two languages. For the evaluation of the approach, we plan to conduct case studies as well as user studies based on industrial-strength markup languages that will be transformed to textual modeling languages including editors that are intended to be at least as powerful as those manually built for XML-based languages.

**Keywords:** Domain Specific Language, Model-Driven Engineering, Language Engineering, Markup Language, Language Modernization

## 1 Problem

With the introduction of machine-processable XML [11] in 1998, the World Wide Web Consortium (W3C) accomplished a tremendous leap towards easing the design of software languages, leveraging the idea of having a generic editor, parser, and validation methodology. Although for prominent XML-based languages — which are themselves DSLs —, such as Business Process Model and Notation (BPMN), advanced editors have been handcrafted, for others dedicated editor

* This research work is supervised by Manuel Wimmer and categorized as *end of initial stage and beginning of maturity stage*. More details are depicted in Section 7.

support is missing. XML has been primarily designed as machine-processible with immutable concrete syntax. More specifically, users of XML-based languages are bound to tree-based syntax that is described as verbose and complex in terms of human comprehension [3]. One of the main consequences of such syntax is the limited capability to improve upon human-comprehension and therefore maintainability. Conquering this limitation requires breaking out of inflexible XML syntax by providing an approach to construct a fully-customizable concrete syntax, which is also referred to as visual syntax or visual notation [23].

Bridging Modelware and Grammarware [26] for markup languages and modeling languages will enable the reuse and ease the maintenance of a significant amount of data encoded in the XML legacy format. Moreover, bridging Grammarware and Modelware for markup languages and modeling languages will lower the entrance barrier for developers to capture models containing domain-specific information and their management in an automated manner through the use of Model-Driven Engineering (MDE) tools [16].

State-of-the-art Model-Driven Language Engineering (MDLE) [17] frameworks, such as Xtext [8], allow the development of Domain-Specific Modeling Languages (DSMLs) [32] and the customization of their textual syntax. However, manually re-creating existing XML-based languages is a complex, error-prone, and time consuming task requiring complex language-engineering skills [10, 20, 22]. Additionally, DSMLs that ought to replace markup languages leave open backward-compatibility with usually comprehensive XML applications.

Evolving from machine-oriented languages to human-oriented languages enables lowering barriers imposed by machine-oriented languages on characteristics such as human perception, cognition, and usability. Likewise, human-oriented languages tend to be less appropriate for executability and interoperability. This suggests the need for automated transformations of languages between different technical spaces [19] as well as the automated transformation of language instances, i.e., XML instances and DSML models. In other words, the bridge between Grammarware and Modelware for markup languages and modeling languages still needs to be established. Individual challenges tackled in the proposed dissertation include but are not limited to the following:

***Modernization Barriers between Markup Languages and DSMLs.*** To pursue the modernization of markup languages, i.e., the evolution of markup languages to languages supporting state-of-the-art technologies, such as, advanced editing capabilities, we need to identify barriers of existing MDLE frameworks as well as develop solutions to automate the modernization procedure. In more detail, we need to find a technique that bridges XSDs with DSMLs in terms of a combined approach that allows to produce instances on both sides. Further, the technique is intended to be applicable to any XSD-based language.

***Generic Modernization of Markup Languages.*** Furthermore, we need to formalize solutions for these barriers in a generic framework that is able to modernize software languages and, in particular, XSD-based languages intended to address individual limitations imposed by machine orientation and human orientation. Whereas this includes appropriate language transformations, it also

requires the transformation of instances defined in such languages. Moreover, state-of-the-art MDE frameworks and tools have to be extended to support individual solutions as well as extended editor features, such as syntax coloring, content assistance, validation, and quick fixes, intending to increase comprehensibility and therefore maintainability.

***Language Comprehensibility Analysis.*** To establish a common ground to evaluate a software language's orientation towards machines and humans, high-level metrics that describe the objectives of both machine-oriented software languages and languages that are human-oriented have to be identified and formalized such as executability and time to understand, create, and manipulate language instances. Furthermore, such metrics have to be measurable in terms of case studies, user studies, and/or other experiments. Therefore, to conduct studies focusing on language comprehensibility and therefore maintainability of individual instances we need to identify and construct or reuse appropriate methods to analyze language comprehensibility metrics.

## 2 Related Work

With respect to our approach of modernizing markup languages, in particular XSD-based languages, with modeling languages, such as Xtext-based languages, there exists a set of related approaches which cover certain aspects regarding the transition between involved technical spaces: (*i*) bridges between XMLware and Modelware and (*ii*) bridges between Modelware and Grammarware. To the best of our knowledge, there exists only one approach [9] to bridge XMLware and Grammarware directly which focuses on XSD and Xtext. But of course, there are other efforts in different contexts for bridging XML schemas and BNF-like languages, e.g., in the context of grammar hunting [35].

***XMLware and Modelware.*** There exist several approaches to either apply transformations from Modelware to XMLware [30, 6, 21, 29, 28] or transformations from XMLware to Modelware [5, 7, 31]. In most of these approaches, Modelware is represented by Unified Modeling Language (UML) [27] models (centered around UML class diagrams) and XMLware either by XSDs or Document Type Definitions (DTDs). We propose an approach that differs from these approaches in several ways. For example, while Wimmer et al. [30] employ DTDs to generate Meta Object Facility (MOF)-based metamodels, we use XSDs to generate Ecore-based metamodels. Furthermore, our approach does not stop after having created the abstract syntax of the language defined, for instance, with a metamodel but also produces a textual concrete syntax using Xtext.

***Modelware and Grammarware.*** There are two different kinds of approaches that aim to bridge Grammarware and Modelware by switching between grammars and metamodels. On one hand, grammar-based approaches [34, 18, 1] generate metamodels out of existing grammar definitions. On the other hand, metamodel-based approaches generate grammars out of existing metamodels [13, 35, 24] or link metamodels with grammars [15]. Especially, EMFText [12] seems to be an interesting alternative to Xtext used in our approach, as there is also the possibility to automatically derive several configurable concrete textual syntaxes

for one metamodel. However, the syntax configuration in EMFText is limited to predefined options and cannot be extended. Furthermore, Cánovas et al. [14] present the Gra2Mol transformation language in which concrete syntax metamodel instances are transformed to abstract syntax metamodel instances.

***XMLware and Grammarware.*** Eysenholdt et al. [9] present a report on the migration of a large modeling environment from XML/UML to Xtext/GMF. In their legacy modeling environment, they identified that XML is inefficient due to verbose syntax and lack of tool support, and that the loading of UML modules and models is very inefficient. Therefore, they performed a modernization of their modeling environment by starting from XML schemas from which Ecore metamodels are created and then manually modified before creating concrete syntaxes through hand-crafted customization. In contrast, the goal of our work is to perform necessary metamodel adaptions in an automated fashion. Therefore, we enable the automated modernization of any XSD-based language as well as avoiding repeated manual adaptations caused by changed XSD specifications.

***Language Comprehensibility.*** Aranda et al. [2] present a framework to evaluate the comprehensibility of (graphical) model representations based on theoretical frameworks in cognitive science. They list several challenges imposed by empirical evaluation of comprehensibility. For example, information equivalence of two different representations cannot be guaranteed even if the underlying conceptual content of different human readers is equal. Qualitative data and a human's inherent ability to operationalize such information is described as notoriously difficult. However, it is possible to construct comprehensibility variables to capture affected comprehensibility—like time required to understand the representation—and affecting comprehensibility—such as previous expertise with the domain being modeled. Eventually, by employing such measures, we hypothesize that human-comprehensibility of concrete syntax can be measured.

## 3 Proposed Solution

The increasing success of models in software development and model transformations in MDE during software development activities, such as automated forward engineering, highlight reasons to benefit from the same infrastructure to automate other tasks. Hence, we propose a model-driven solution that automates bridging XMLware, Modelware, and Grammarware. Our goal is to provide a framework that automatically modernizes XSD-based languages to metamodel-based languages that are supported by rich language workbenches, flexible syntax, and model-based techniques such as code generation, transformation, and validation. We achieve this by (*i*) chaining together tools and transformations (into what is from now on referred to as *Default Transformation Chain*), (*ii*) introducing new transformations that overcome existing gaps between XMLware, Modelware, and Grammarware such as mixed content and wildcards, data types and restrictions, and identifiers and identifier references, as well as (*iii*) introducing a *Concrete Syntax Configuration Language* enabling the flexible definition of textual concrete syntaxes.

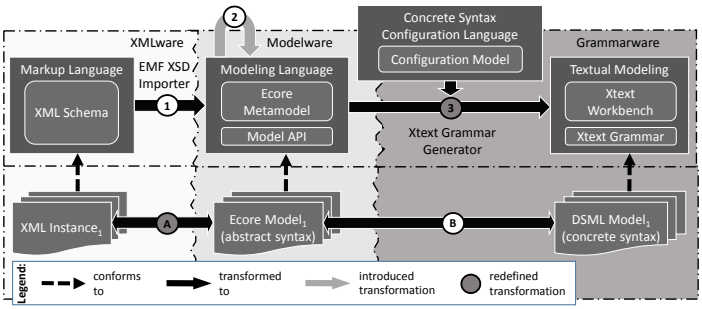


**Fig. 1.** Overview of the XMLText framework

Figure 1 depicts a conceptual overview of our XML to Xtext (XMLText) framework [25]. Like in the *Default Transformation Chain*, the first step is to transform a given *XML Schema* to an *Ecore Metamodel* by employing the *EMF XSD Importer* ①. To overcome limitations of this transformation, we complement it with novel transformations that adapt the generated *Ecore Metamodel* ②. For example, in order to tackle the issue of not supported feature maps occurring for mixed content and wildcards, we adapt the *Ecore Metamodel* by replacing feature maps with generic concrete constructs. Next, the adapted metamodel is used as input for generating the *Xtext Grammar* with the *Xtext Grammar Generator* provided by the Xtext framework ③. Also this step has to be extended with novel transformations that overcome limitations of the existing grammar generator, e.g., to enable storing actual values for attributes by importing, and referencing a library of data types. Moreover, we enable the automated customization of the target DSML's textual concrete syntax by providing a *Configuration Model*, i.e., a customizable template to specify the concrete syntax striving to enhance human-comprehensibility and therefore maintainability. For the adaptions of the *Ecore Metamodel* introduced by the XMLText framework, it is necessary to customize existing transformations (cf. Ⓐ in Figure 1) to act upon them on instance level. Therefore, we customize (*i*) the deserializer that reads *XML Instance*s and creates in-memory *Ecore Model* representations conforming to the adapted *Ecore Metamodel* and (*ii*) the serializer that stores *Ecore Model*s as *XML Instance*s. As a result of keeping the *Xtext Grammar* coupled to the *Ecore Metamodel*, we are able to reuse the existing transformation Ⓑ between instances of the *Xtext Grammar* and the *Ecore Metamodel*.

With the introduction of transformation ② and the adaption of the transformations ③ and Ⓐ, our XMLText framework overcomes limitations of existing bridges between *XMLware* and *Grammarware* and thus allows an improved automated modernization of XML-based languages to metamodel-based and textual DSMLs. Listing 1.2 shows the result of applying the XMLText framework on an exemplary XML-based language instance, i.e., specifying an Apache web server cloud node, used in Listing 1.1.

```
1 <nodeTemplate id="ApacheWebServer" type="ApacheWebServerType" name="
      ↪Apache␣Web␣Server">
2   <properties id="ApacheWebServerProperties">
```

```
    <numCpus>1</numCpus>
    <memory>1024</memory>
  </properties>
</nodeTemplate>
```

**Listing 1.1.** Exemplary XML-based language instance

```
TNodeTemplate ApacheWebServer {
  name: "Apache Web Server"
  type: ApacheWebServerType
  Properties ApacheWebServerProperties {
      NumCpus: "1"
      Memory: "1024"
  }
}
```

**Listing 1.2.** Exemplary DSML model

## 4 Preliminary Work

Preliminary work in MDE and specifically in terms of interoperability between languages ultimately led to this research topic. In earlier work [4] we realized that many modeling languages targeting equal or similar purposes, e.g., modeling of cloud applications, are built from scratch causing extensive mismatches and difficulties in terms of interoperability among each other [33]. Hence, in [25] we established an initial framework that exploits existing seams between the technical spaces XMLware, Modelware, and Grammarware as well as closes several gaps between them. The resulting approach is able to generate Xtext-based editors from XSDs providing extended editor functionality, customization of textual concrete syntax, and round-trip transformations enabling the exchange of data between the involved technical spaces. The feasibility of the approach has been evaluated by a case study on TOSCA—an XML-based standard for defining Cloud deployments. The results show that the approach enables bridging XMLware with Modelware and Grammarware in several ways going beyond existing approaches by integrating useful and independent parts as well as improvements that allow the automated generation of editors that are at least equivalent to editors manually built for XML-based languages. However, while the results indicate that most of the known gaps have been bridged, some, e.g., XML namespaces, still need to be overcome. Further, XSD-based languages that employ a different subset of XML Schema language constructs may uncover previously unknown gaps that require further investigation.

## 5 Expected Contributions

The general idea to tackle the problem introduced in Section 1 is to address the challenges involved in modernizing software languages using MDE techniques. Hence, the following contributions are proposed:

***Individual Language Modernization Techniques.*** This contribution intends to overcome individual barriers of existing MDLE frameworks when employing them in the formalization of techniques for the purpose of modernizing markup languages with DSMLs. For example, a technique has to be developed to transform XSD constructs that are currently not natively supported by Ecore,

such as data types, restrictions, and wildcards. Moreover, a language is developed that enables the configuration of the generated language in terms of configuration models. Furthermore, for tackling the challenge of transforming instances between XML-based languages and modernized DSML languages, we will develop model transformations that start from the meta language level and enable transformations on the instance level. Next, fitting techniques and extensions thereof are then used to implement the transformation of language instances starting from the meta language level. Finally, the soundness of individual modernization techniques will be validated using appropriate experiments.

***Generic Framework and Initial Case Study.*** This contribution is designated to overcome challenges dictated by the generic modernization for markup languages through (*i*) the implementation of a generic framework facilitating existing MDE frameworks and tools, (*ii*) merging language modernization techniques, and (*iii*) extending a language's comprehensibility by incorporating the concrete syntax configuration language as well as providing advanced editor features, such as syntax coloring, content assisting, validation, and quick fixes, and their implementation in a generic and automated way. Moreover, to show that a markup language can be modernized into a machine-readable and human-comprehensible language, an initial case study, based on an XSD-based language, will be performed evaluating the feasibility of the developed generic framework.

***Comprehensibility and Maintainability Analysis Framework.*** This contribution seeks to overcome the challenges imposed by evaluating language comprehensibility and therefore maintainability. First, a literature review on comprehensibility analysis, such as empirical comprehensibility [2], will be performed. This includes establishing a common ground to analyze a software language's orientation towards machines and humans and the formalization of high-level metrics that describe the objectives of both machine orientation and human orientation. Second, an evaluation mechanism for software languages in terms of formalized metrics is provided by a set of analysis methods. Next, both metrics and analysis methods thereof are implemented in an analysis framework that can be executed in the form of case studies, user studies, or other experiments, like machine analysis of software artifacts, or a combination of such.

## 6 Plan for Evaluation and Validation

The evaluation and validation of the proposed work are threefold. First, our solution to the problem caused by the current state of the art and its inability to transform instances between the technical spaces Grammarware and Modelware for markup language and modeling languages has been and will be evaluated by conducting case studies on the established bridge. In more detail, the initial case study evaluated the feasibility of our software language modernization framework XMLText in modernizing a markup language with a DSML, i.e., a modeling language, as well as validate the conformance of instances to their respective language. Language semantics are evaluated by performing round-trip transformations, i.e., comparing the source instance with the instance resulting from the round-trip transformation on the same source instance in terms of equality, as well as supplying instances to existing interpreters and compar-

ing the resulting behavior. Moreover, the study also included a comparison of the DSML produced by our framework as well as a hand-crafted DSML of the same language in terms of their completeness to their source XSD-based language. Furthermore, we plan to extend our initial case study by conducting a case study on a set of markup languages ensuring that all language concepts occurring in XSD-based markup languages are covered.

Second, we will evaluate the usability of our framework, in particular, the usability for domain engineers that usually do not have language engineering skills. In more detail, this study will involve professionals as well as students, a categorization of their language engineering skills, an evaluation of their experience with modernizing markup languages with modeling languages by using our framework, as well as the modernized language they produced.

Third, our solution to the problem of fixed concrete syntax of XML hindering human-comprehensibility and hence maintainability will be evaluated by conducting a user study. This study will involve engineers without advanced knowledge in both XMLware and Modelware and evaluate several aspects that are considered important in human comprehensibility metrics. For example, time spent to understand the representation and to create and manipulate instances by employing generic XML editors and their modernized counterparts, i.e., DSML model editors will be evaluated.

Relevant conferences for publishing the results include ECMFA, MODELS, SLE, SPLC, ASE, and PLDI. Moreover, relevant journals include TOPLAS, SoSyM, JSS, and IJPOP.

## 7 Current Status

Figure 2 shows the plan for conducting the presented research as well as its current status.

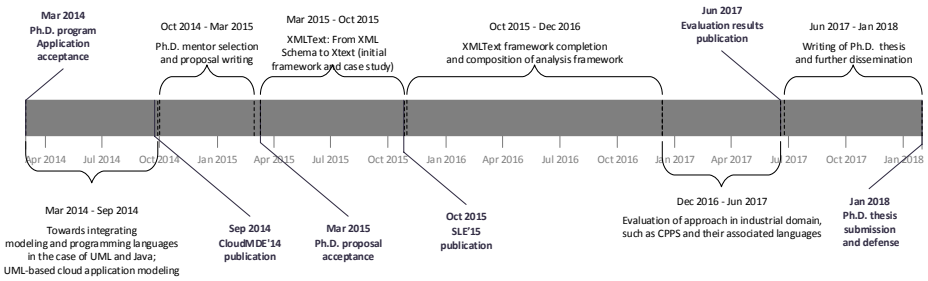


**Fig. 2.** Project phases and events time-line

We started by working on UML-based cloud application modeling [4]. The results have been published in September 2014. During the next phase a mentor has been selected as well as the proposal submitted and accepted. Next, we started working on our software language modernization framework XMLText [25] which resulted in an initial framework and case study based on the cloud topology and orchestration language TOSCA (cf. Section 4) published in

October 2015 at the International Conference on Software Language Engineering (SLE). Currently, we are working on completing the XMLText framework by bridging not yet resolved gaps, develop the concrete syntax configuration language, as well as establishing evaluation metrics and analysis methods required in the following phase. In more detail, the following phase will involve an evaluation of the overall approach in the industrial domain Cyber-Physical Production Systems (CPPS) and its associated languages in terms of a case study as well as a user study (cf. Section 6). The intention of selecting the domain of CPPS is caused by the fact that our faculty runs a laboratory[1] working in this domain and therefore provides us access to an extensive range of industrial partners and a real-world evaluation environment. The latter phase will be concluded with a publication of evaluation results in mid-2017, followed by the composition of a doctoral thesis as well as further dissemination of the established approach and framework.

## References


1. Alanen, M., Porres, I.: A Relation Between Context-Free Grammars and Meta Object Facility Metamodels. Tech. rep., Turku Centre for Computer Science (2003)
2. Aranda, J., Ernst, N., Horkoff, J., Easterbrook, S.: A framework for empirical evaluation of model comprehensibility. In: International Workshop on Modeling in Software Engineering (MISE). pp. 7–7 (2007)
3. Badros, G.J.: JavaML: A Markup Language for Java Source Code. Computer Networks 33(1), 159–177 (2000)
4. Bergmayr, A., Troya, J., Neubauer, P., Wimmer, M., Kappel, G.: UML-based cloud application modeling with libraries, profiles, and templates. In: Proceedings of the 2nd International Workshop on Model-Driven Engineering on and for the Cloud (CloudMDE). pp. 56–65 (2014)
5. Bird, L., Goodchild, A., Halpin, T.: Object Role Modelling and XML-Schema. In: Proc. of ER, pp. 309–322. Springer (2000)
6. Booch, G., Christerson, M., Fuchs, M., Koistinen, J.: UML for XML schema mapping specification. Rational White Paper (1999)
7. Conrad, R., Scheffner, D., Freytag, J.C.: XML Conceptual Modeling using UML. In: Proc. of ER, pp. 558–571. Springer (2000)
8. Eysholdt, M., Behrens, H.: Xtext: Implement your Language Faster than the Quick and Dirty Way. In: Companion Proc. of OOPSLA. pp. 307–309. ACM (2010)
9. Eysholdt, M., et al.: Migrating a Large Modeling Environment from XML/UML to Xtext/GMF. In: Companion Proc. of OOPSLA. pp. 97–104. ACM (2010)
10. Fowler, M.: Domain-specific languages. Pearson Education (2010)
11. Harold, E.R., Means, W.S., Udemadu, K.: XML in a Nutshell, vol. 8. O'reilly Sebastopol, CA (2004)
12. Heidenreich, F., Johannes, J., Karol, S., Seifert, M., Wende, C.: Model-Based Language Engineering with EMFText. In: GTTSE. pp. 322–345 (2011)
13. Heidenreich, F., Johannes, J., Karol, S., et al.: Derivation and Refinement of Textual Syntax for Models. In: Proc. of ECMDA-FA. pp. 114–129 (2009)
14. Izquierdo, J.L.C., Molina, J.G.: Extracting models from source code in software modernization. Software and System Modeling 13(2), 713–734 (2014)


[1] A list of currently ongoing Christian Doppler laboratories at the TU Wien can be found online at `http://goo.gl/HA95Rp`

15. Jouault, F., Bézivin, J., Kurtev, I.: TCS: a DSL for the specification of textual concrete syntaxes in model engineering. In: Proc. of GPCE. pp. 249–254. ACM (2006)
16. Kolovos, D.S., Rose, L.M., Williams, J.R., Matragkas, N.D., Paige, R.F.: A Lightweight Approach for Managing XML Documents with MDE Languages. In: Proceedings of Modelling Foundations and Applications - 8th European Conference (ECMFA). pp. 118–132 (2012)
17. Kühne, T.: What is a model? In: Language Engineering for Model-Driven Software Development (2004)
18. Kunert, A.: Semi-Automatic Generation of Metamodels and Models from Grammars and Programs. Electronic Notes in Theoretical Computer Science 211, 111–119 (2008)
19. Kurtev, I., Aksit, M., Bézivin, J.: Technical Spaces: An Initial Appraisal. In: Proc. of CoopIS (2002)
20. Luoma, J., Kelly, S., Tolvanen, J.P.: Defining Domain-Specific Modeling Languages: Collected Experiences. In: Proc. DSM Workshop (2004)
21. Mani, M., Lee, D., Muntz, R.R.: Semantic Data Modeling using XML Schemas. In: Conceptual Modeling (ER), pp. 149–163. Springer (2001)
22. Mernik, M., Heering, J., Sloane, A.M.: When and how to develop domain-specific languages. ACM computing surveys (CSUR) 37(4), 316–344 (2005)
23. Moody, D.L.: The physics of notations: toward a scientific basis for constructing visual notations in software engineering. IEEE Transactions on Software Engineering 35(6), 756–779 (2009)
24. Muller, P.A., Hassenforder, M.: HUTN as a bridge between modelware and grammarware-an experience report. In: WISME Workshop (2005)
25. Neubauer, P., Bergmayr, A., Mayerhofer, T., Troya, J., Wimmer, M.: XMLText: From XML Schema to Xtext. In: Proceedings of the International Conference on Software Language Engineering. pp. 71–76. ACM, New York, NY, USA (2015)
26. Paige, R.F., Kolovos, D.S., Polack, F.A.C.: Metamodelling for grammarware researchers. In: Proc. of SLE. pp. 64–82 (2012)
27. Rumbaugh, J., Jacobson, I., Booch, G.: Unified Modeling Language Reference Manual. Pearson Higher Education (2004)
28. Salim, F.D., Price, R., Krishnaswamy, S., Indrawan, M.: UML documentation support for XML schema. In: Australian Conference on Software Engineering. pp. 211–220. IEEE (2004)
29. dos Santos Mello, R., Heuser, C.A.: A Rule-Based Conversion of a DTD to a Conceptual Schema. In: Proc. of ER, pp. 133–148. Springer (2001)
30. Schauerhuber, A., Wimmer, M., Kapsammer, E., Schwinger, W., Retschitzegger, W.: Bridging WebML to model-driven engineering: from document type definitions to meta object facility. IET Software 1(3), 81–97 (2007)
31. Skogan, D.: UML as a schema language for XML based data interchanging. In: Proc. of UML (1999)
32. Tolvanen, J., Kelly, S.: Defining domain-specific modeling languages to automate product derivation: Collected experiences. In: Proc. of SPLC. pp. 198–209 (2005)
33. Vallecillo, A., et al.: MDWEnet: A practical approach to achieving interoperability of model-driven web engineering methods. In: Proceedings of the 3rd International Workshop on Model-Driven Web Engineering (MDWE) (2007)
34. Wimmer, M., Kramler, G.: Bridging grammarware and modelware. In: Proc. of Satellite Events at MoDELS. pp. 159–168. Springer (2006)
35. Zaytsev, V.: Grammar Zoo: A corpus of experimental grammarware. Sci. Comput. Program. 98, 28–51 (2015)