Model-driven Standardization of Public Authority Data Interchange

Fabian Büttner[a,*], Ullrich Bartels[b], Lars Hamann[c], Oliver Hofrichter[c], Mirco Kuhlmann[c], Martin Gogolla[c], Lutz Rabe[d], Frank Steimke[d], Yorck Rabenstein[e], Alina Stosiek[e]

[a] *AtlanMod, École des Mines de Nantes - INRIA, Nantes, France*
[b] *Independent Consulting Architect, Germany*
[c] *Database Systems Group, University of Bremen, Germany*
[d] *Koordinierungsstelle für IT-Standards, Die Senatorin für Finanzen, Bremen, Germany*
[e] *]init[AG, Berlin, Germany*

Abstract

In the past decade, several electronic data exchange processes between public authorities have been established by the German public administration. In the context of various legacy systems and numerous suppliers of software for public authorities, it is crucial that these interfaces are open and precisely and uniformly defined, in order to foster free competition and interoperability. A community of such projects and specifications for various public administration domains has arisen from an early adopter project in the domain of data interchange between the 5,400 German municipal citizen registers. A central coordination office provides a framework for these projects that is put into operation by a unified model-driven method, supported by tools and components, involving UML profiles, model validation, and model-to-text transformations into several technical domains. We report how this model-driven approach has already proven to be effective in a number of projects, and how it could contribute to the development of standardized e-government specifications in various ways.

Keywords: e-government, standardization, interoperability, industrial case study, model validation, model transformation

1. Introduction

This article reports on the successful application of model-driven engineering (MDE) in the context of a national e-government strategy [28]. The beginnings of this story date back to 2001, when a model-driven development approach was first employed by an early adopter project of the German federal and state governments named OSCI–XMeld[1]. The aim was to efficiently develop a specification for the standardized, XML-based electronic data exchange between the approx. 5,400 German municipal citizen registers [4]. This specification, which essentially describes a set of secure web services, has been made mandatory by the German Ministry of Interior. It is today implemented by all software vendors providing applications in this domain.

Emanating from the success of this first project, the approach has been generalized and transferred to other projects in the following years. Many of them were also successful, so that the German federal and state governments officially gave this model-driven way of developing XML-based e-government data interchange specifications (so-called *XÖV*[2] specifications) an official recommendation in 2006.

*This research was partially funded by the Nouvelles Équipes Program of the Pays de la Loire Region (France).

[1] In German, 'Meld' abbreviates *Meldewesen*, the domain of citizen registration

[2] In German, 'XÖV' – 'XML in der Öffentlichen Verwaltung' – can be translated to 'XML in the public administration'

To address the interoperability issues that came up naturally with a growing number of XÖV specifications, an office for the coordination of their development was established, and the approach became – in addition to supporting the efficient production of individual specifications – also an instrument to foster their syntactical and semantic interoperability. Thus, the 'standardization of e-government standards' came into focus.

In September 2010, the federal and state government established the permanent IT Planning Council[3], whose IT standards coordination office[4] is now (among other responsibilities) in charge of providing and operating a uniform model-driven development method and corresponding tools for currently 17 XÖV specifications in several public administration domains, with further projects on the horizon.

We want to be clear that the success of e-government projects is first of all determined by non-technological aspects. In particular, the transition from paper-based to electronic data interchange requires a close collaboration of the concerned administrative domains and a semantic alignment of the legal foundations. In this context, a technological approach like the one we present here can only be supportive. Nevertheless, after one decade of applying MDE to e-government standardization in various projects, some of them on a large scale and being in operation (and object of permanent extension) for several years already, we can state that it is an ongoing success story. Our development approach supports the objectives in various ways, for the actors in individual standardization projects (in terms of development cost, quality, and time to market) as well as for the central coordination office (in terms of assessing and supporting semantic alignment and interoperability among the standards). From the MDE perspective, the approach comprises the following elements:

- several UML models and further semantic artifacts, some belonging to the owners of the individual XÖV specifications, some being shared semantic artifacts maintained by the IT coordination office;
- UML profiles (including well-formedness rules) defining domain-specific metadata and governing that the models conform to given interoperability criteria for XÖV;
- a common, configurable open source tool, the *XGenerator*, that automatically validates the models against the profiles and transforms them into various artifacts that make up a data interchange specification (e.g., documentation fragments, XML Schema files, web service description files); and
- a central, web-based repository, the *XRepository*, that holds the various semantic assets (in particular, the models) of all standards.

In this article, we explain how these elements, together, form a system for the development and maintenance of standardized XÖV specifications, and we discuss how this approach has proven effective. The following sections are organized as follows: In Section 2 we explain the context of e-government standardization in more detail, in order to provide the context for our approach. Section 3 explains the model-driven development system for XÖV specifications. Section 4 presents several facts and observations gathered from the projects and it discusses why we consider our approach an MDE success story. Section 5 puts our approach in the context of related work. Section 6 summarizes and concludes the paper and points out future directions.

2. Context of our Work

In order to explain our technical contribution, we first need to introduce the domain of e-government standardization in Germany in slightly more details.

[3]In German: IT-Planungsrat
[4]In German: Koordinierungsstelle für IT-Standards (KoSIT)

2.1. XÖV Specifications for e-government Data Interchange

In order to offer best service to the public, the public administration establishes an increasing number of electronic communication processes. While the administrative processes within individual public authorities (e.g., within one municipal office) are mostly carried out using IT systems already, processes that span multiple authorities and domains are often still based on paper. Therefore, they are time and labor consuming due to media conversions. One of the main reasons for this situation is that the German law sets high requirements for secure data exchange in the field of e-government. It was not before techniques for electronic signatures and secure web services became available, that the latter kind of processes came more and more into the IT focus.

In a heterogeneous landscape of systems, vendors, and operators for the public administration, establishing electronic data interchange between authorities requires open specifications of data interchange that are independent of the individual IT products, in order to ensure free competition and avoid vendor lock-in. For domain-independent aspects such as data transport, authentication, and encryption, the responsible bodies of the public administration can today refer to well-accepted industrial standards in order to define the required service infrastructure (typically web services). However, the semantic (application) layer is in general specific to the various legal and organizational aspects of the administrative domain. Thus, we find several so-called semantic specifications that are developed, maintained, and owned by different bodies of the public administration (typically federal ministries).

XÖV aims to standardize the development of such specifications. There are currently 17 XÖV specifications, covering a wide range of domains, for example the communication between municipal citizen registers and the federal tax authority, transmission of data between law courts, and transmission of data to the federal office of statistics. The specifications do not address the direct communication with citizens, but focus on data interchange between IT systems of public authorities. In general, they directly map legal liabilities and duties into the technical domain, and in several cases, the adherence to the specification has been made mandatory by the responsible body of the public administration (e.g., a federal ministry). Notice that, in a complementing sense, it has to be ensured by the specification developers that the processes described in the specification do not exceed the legal basis (which would be a privacy breach).

It is important to mention that the public authorities typically do not implement these specifications themselves. In general, systems and products are provided by the market. Thus, specifications must be open and precise enough to be implemented by various software vendors. In practice, the vendors are often integrated into the specification development process in order to ensure that the specification can be eventually implemented without unforeseen problems.

From a semantic point of view, an XÖV specification comprises process models, messages, semantic data types, and code lists, as shown in the first two columns in Table 1. When an XÖV specification is finally released, these artifacts are packaged into several deliverables:

1. a normative PDF document describing the scope of application, the data interchange processes and their messages, and further technical details, such as the mapping onto the secure transport architecture of the respective domain;
2. a UML model, containing the process models and the information structures of the messages
3. representations of the messages in XML Schema and description files for the different services in the Web Service Description Language (WSDL); and
4. XML representations of all code lists in the Genericode [39] format.

2.2. Interoperability and Standardization

With a growing number of XÖV specifications, increased coordination and standardization are required in order to achieve synergies and interoperable implementations. This is particularly important because IT systems of one authority often have to implement multiple specifications, as they have to exchange information with communication partners from different domains.

The central coordination office for IT standards has been put in charge of this. It consults the different owners of XÖV specifications (typically federal ministries) in the development, recommends best practices,

Semantic artifact	*OSCI–XMeld* example	XÖV interoperability rules
Process models. They describe the data interchange process between two or more authorities (modeled as UML activity diagrams: who sends which message to whom, under which condition).	'Registration in a new municipality' (e.g., after a relocation) triggers a message interchange between the old municipality, the new municipality, and the federal tax authority.	Technical processes and messages, such as 'received ill-formed message' and 'return to sender' should be kept outside of individual specifications. Wherever possible, these aspects should be defined in the secure transport layer referenced by the XÖV specification (typically industrial standards for secure web services). All XML Schema files must follow the naming and design rules for XÖV.
Messages. The messages in these models are specified by UML classes, corresponding XML schemas, and additional textual documentation.	A 'conflict' message is sent from the federal tax authority to two or more municipal citizen registers when the tax authority suspects that the same citizen is registered with principal residence in two municipalities.	
Semantic Data Types. Recurring building blocks of messages are modeled as explicit entities with well-defined semantics. They are represented as UML classes and, where applicable, corresponding types in XML Schema. Information entities can also be shared among specifications, which makes those entities specifications of their own.	The 'address of residence' is a common information entity that is used in many places not only in the *OSCI–XMeld* specification, but also in two others, the civil status data interchange (*XPersonenstand*) and the interchange between foreign offices (*XAusländer*).	The same concept should not be modeled twice (in different specifications). Ideally, similar information entities should be generalized. There are two levels of interoperability for information entities: semantic alignment following the Core Components methodology [10], and, stronger, real reuse of XML Schema data types down to the XML Schema level.
Code Lists. Enumerations of value domains (as part of messages) are very common in all data interchange specifications. They can be a specific part of a specification, but they can also be stand-alone (allowing to be changed independently).	The 'list of all religious communities for which the government collects their taxes' is an example for a code list that is used by the *OSCI–XMeld* specification, but that is kept and maintained separately.	The same concept should not be modeled twice. Shared and independent code lists should be referenced and distributed consistently.

Table 1: Semantic building blocks of XÖV specifications

and monitors the specification landscape. It also aligns the national efforts with the activities in other countries, in particular the European Union.

For XÖV projects, the coordination maintains a framework [34] covering both organizational and technical facets of the development of XÖV specifications and providing guidelines for new projects. The organizational side addresses aspects such as ownership and maintenance, which are not in the focus of this article. Before a specification can be released as an XÖV specification, it has to be certified by the coordination office, which verifies whether the specification meets all organizational, semantic, and technical requirements.

Apart from organizational aspects, the framework provides several rules of different requirement levels (*must*, *should*, *recommendation* [6]) that aim at interoperability among the standards on the semantic and technical side, as sketched in the third column of Table 1. The first level *must* describes mandatory requirements. The other levels are both non-mandatory. Violations of *should* rules must be explicitly justified when applying for XÖV conformance certification, whereas deviations from *recommendation* do not have to be justified.

On the semantic level, the objective is to share common data types between the specifications. In XÖV, the strongest level of alignment is the actual reuse of XML Schema data types, but less strict forms are possible, too. The latter is supported by a set of so-called core components (non-technical descriptions of data types), which have been developed by a series of 'data conferences' of the XÖV projects from 2006 to 2009, following the UN/CEFACT Core Components methodology [10]. The XÖV framework complements

frameworks such as SAGA[5], which lists recommended industrial standards for e-government interoperability in Germany [46].

A mandatory requirement for all projects is that the specification must be represented by one or more well-structured UML models, and that all XML Schema files must be generated from these models using a fixed model-to-Schema transformation (which is part of the MDE approach to be described in the following section). The models are the main artifact when certifying the conformance of XÖV specifications.

3. MDE for XÖV

The coordination office maintains a model-driven system for the development of XÖV specifications. It supports the effective development of XÖV specifications and fosters the interoperability between them, as described in the previous section. Its application is mandatory in order to be certified.

The first class citizens in this system are the specification models, defining processes and semantic components, e.g., data types. All semantic components can be shared as separate sub-models and be reused by other specifications, and a central 'market place' for those objects is provided by the *XRepository* [54].

To each model, one or more profiles are applied to capture additional meta information. The standard profile provides annotations to describe the reuse of shared semantic components, to provide structured documentation (e.g., to explicitly link processes and messages to their legal foundations), and to describe the technical representation of the models as XML schemata and web service descriptors.

The standard profile comprises extensive well-formedness rules that ensure its correct application. This way, it enforces several XÖV conformance rules already on the modeling level. In addition to the standard profile, the projects are free to add individual profiles according to their needs, for example, to introduce more stringent well-formedness rules respecting project-specific modeling conventions, or to include further technical and non-technical metadata in the model.

A configurable, open source MDE tool, the *XGenerator* [9], is provided for all projects. It performs two major tasks: First, it automatically validates the conformance of the models to the well-formedness rules of all applied profiles (giving detailed feedback about violations and offending model elements). Second, it generates the various required technical representations from a model, such as XML schemata, web service descriptors, and documentation fragments (in DocBook format). Figure 1 illustrates the MDE-related development activities.

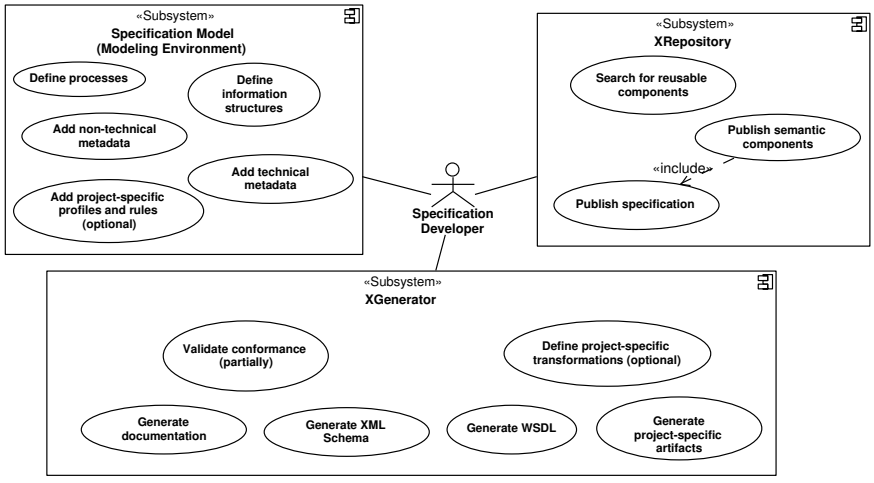

Figure 1: Tasks in the MDE environment

[5]German for: standards and architectures for e-government applications

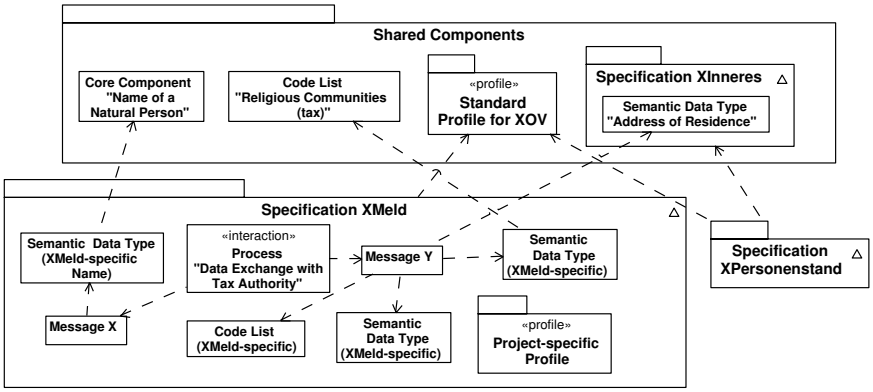

Figure 2: Specification models and shared semantic components

3.1. Specification Model

Figure 2 depicts the elements of the XÖV modeling world using the three aforementioned specifications for the municipal domain, *OSCI–XMeld*, *XPersonenstand*, and *XAusländer*. The figure focuses on the specification model for the XÖV specification, *OSCI–XMeld*, containing most importantly the processes (UML interactions) and information structures (classes modeling the messages that are sent as part of the processes). These information structures comprise semantic data types (classes modeling recurring information entities) and code lists (a general form of enumerations with metadata). In addition to the semantic components that belong exclusively to the specification *OSCI–XMeld*, the *OSCI–XMeld* model also refers to shared semantic components.

3.1.1. Shared Semantic Components

There are four kinds of semantic components that are shared and reused by specifications: (1) semantic data types, (2) core components, (3) code lists, and (4) full specifications that aggregate components. As discussed in Section 3.3, the XRepository is the central point to publish and access these shared semantic components.

Semantic Data Types. Semantic data types can be shared and related between XÖV specifications in two manners:

As the most direct (and most interoperable) kind of reuse, two or more XÖV specifications can 'outsource' common data types, typically packaged as another XÖV specification. On this level of reuse, the specifications share actual UML classes (e.g., the 'Address of Residence'). The technique for reuse is the import mechanism at the XML Schema level.

Core Components. On a weaker level, core components [10, 47] are used to ensure the semantic alignment between data types on a conceptual level. Core components do not have a technical representation per-se, but define a common agreement between all XÖV projects on the semantics of central basic concepts such as 'Name of a Natural Person'. Individual standards can derive their semantic data types from core components, allowing a certain degree of freedom, e.g., to remove elements that are not required and to deviate from the structure where needed for legacy reasons. In the scope of this article, core components are defined by classes that are tagged using stereotypes from a corresponding core components UML profile. When specific data types (i.e., as part of a specification model) are derived from a core component, the correct derivation is ensured using several well-formedness rules. This includes that any deviations from the core component (which is possible) must be documented.

Code Lists. Code lists are a central semantic asset in all data interchange processes, and the agreement on same code lists is essential for interoperable services. Code lists can be integrated into or referenced

from specifications, depending on the volatility of the list. Code lists can be regarded as versioned, indexed tables that can be referenced using a global unique identifier. Inside the model, code lists can be represented as UML enumerations (which is appropriate for small, static code lists), or they can be referenced using only the code list metadata (which is appropriate for large or volatile code lists). Code lists are distributed in the Genericode format [39].

XÖV Specifications. Finally, it is possible to reuse specification models. This enables, in particular, related specifications to share packages of semantic data types that have a unique technical representation. For example, the three XÖV specifications *OSCI–XMeld*, *XPersonenstand*, and *XAusländer* (c.f. Tables 1 and 2) share a common module named XInneres, containing common data types. This way of reuse is stronger than using core components and requires a strong legal and organizational connection between the sharing specifications (in terms of ownership and maintenance, etc.).

3.1.2. UML Profiles

To enrich the conceptual models with domain- and platform-specific information, and to enforce several structural properties of the models, UML profiles are attached to the specification models. In particular, the generation of XML Schema and WSDL files from the model, as well as keeping structured documentation such as legal references requires the annotation of several stereotypes. In this aspect, the core, mandatory profile for XÖV specifications (the *XÖV profile*) makes a compromise between uniformity and expressiveness: While it aims at a uniform generation of schemata and web service descriptions for all specifications, it has to deal with several legacy situations. Thus, it provides at some points ways to specify platform-specific information in the model. However, for non-legacy projects, we generally aim to assume a default translation to XML Schema. In this case, only little platform-specific information has to be provided in the model in order to transform it into the technical representations.

The *XÖV profile* currently comprises 34 stereotypes. Some of them are direct representations of target platform elements like «xsdElement» for XML schema elements, «xsdTitled» for meaningful titles in the documentation or «wsdlService» for WSDL service definitions, while other are more abstract and lead to more complex transformation rules such as «xsdCodeList» and «xsdCode» for the representation of a code list and the data type which represents values of such a list.

The *XÖV profile* subsumes central concepts of the UN/CEFACT methodology [10], which enables the reuse of core components, as described above. In addition to this standard profile, further project-specific profiles can also be attached to the models, e.g., to specify references to the respective legal foundation in a more fine-grained manner than in standard profile, i.e., describing on the level of messages and fields which clause of which legal act governs that data transmission. In the *OSCI–XMeld* model, for example, messages are annotated with references such as "§§ 139b Abs. 7 und 8 AO und 39e Abs. 2 Nr. 1 – 3 EStG", referencing to the specific clauses of the two German fiscal acts that require (and allow) this data transformation. This level of legal references is supported by the standard *XÖV profile*. Via an *OSCI–XMeld*-specific profile, the individual data fields used by the messages are further annotated with references to the federal catalog of data fields for municipal citizen registers [8].

All profiles can be complemented by well-formedness rules specified as invariants using the Object Constraint Language (OCL)[52, 43]. These rules are processed by the XGenerator to validate the models. Our approach makes extensive use of non-trivial well-formedness rules to precisely express XÖV conformance rules as far as possible on the modeling level. For example, the current version of the *XÖV profile* contains 122 well-formedness rules. As a specific extension to OCL, our approach attaches a level (*must*, *should*, *recommendation*) to each well-formedness rule, corresponding to the levels of the XÖV conformance rules. The well-formedness rules are handled differently in the model validation step depending on their level. Of the 122 well-formedness rules of the *XÖV profile*, 88 are *must*, 11 are *should*, and 23 are *recommendation*.

3.2. XGenerator

The central MDE application in our approach is an open source application called *XGenerator* [9]. It fulfills three major tasks inside of the overall MDE process:

1. model validation, to check well-formedness rules as defined by the UML profiles,

2. model-to-text transformation, to generate various output formats, and
3. additional post-transformation validation of the generated artifacts.

The XGenerator is able to read profiled UML 2 models based on the Eclipse Modeling Framework (EMF) [48, 18]. After successfully reading an input model the XGenerator executes the aforementioned tasks whereas a proceeding task is only executed if the previous task terminated successfully. Further, the user can invoke a 'validation only' run. The three tasks of the XGenerator, as depicted in Fig. 3, are described in more detail in the next subsections.

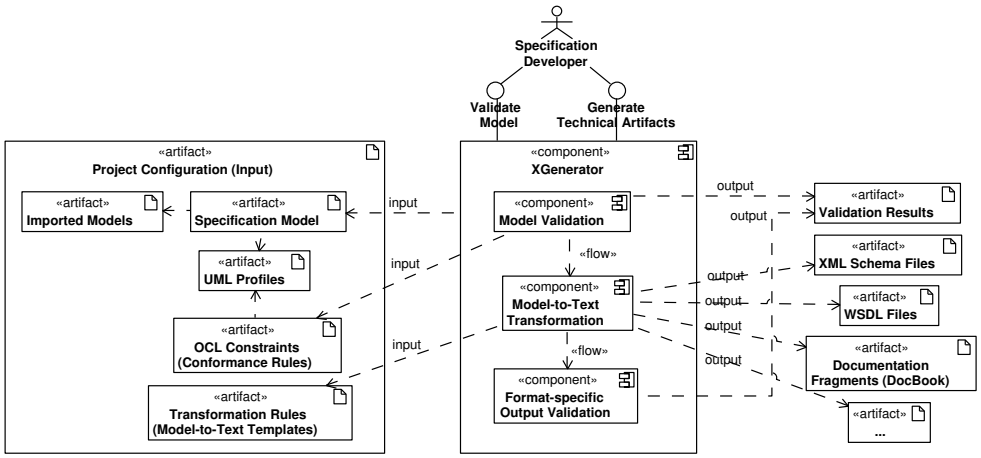

Figure 3: The XGenerator component

3.2.1. Model Validation

The validation is conducted by checking the well-formedness rules that are defined for the profiles (and which are defined in the context of metamodel elements such as *Class* or *Property*). In contrast to the common definition of invariants that a model is only consistent if all given invariants evaluate to true, the XGenerator allows a finer grained categorization of invariants, according to the three requirement levels generally used in XÖV. For the strongest level *must*, all invariants need to be fulfilled to accept the input model as well-formed. If such an invariant fails no valid output will be generated (as the result would be syntactically invalid or not conforming to XÖV). Violations of weaker rules (*should* and *recommendation*) are reported to the user as warnings (for the *should* level) and hints (for the *recommendation* level), but do not abort the overall generation task.

3.2.2. Model Transformation

All output artifacts are text-based (e.g., XML Schema, WSDL, DocBook, SVG), and model-to-text transformations are used to generate them. A transformation language has been specifically developed for the XGenerator. It is an imperative transformation language (c.f. [36]) that is based on templates, similar to *MOF Model to Text* [40]. These templates merge three different languages:

1. a query language for the source model;
2. an imperative control flow language; and
3. the text fragments of the target language.

As a query language for the templates, the Object Constraint Language (OCL) is used. The XGenerator realizes OCL-based queries by integrating the core of the UML/OCL tool USE [22, 7, 49]. To reuse expressions, additional helper query operations can be defined as libraries to simplify the templates. As the imperative language, Apache Velocity [51] is used, which provides the required control structures and the various text manipulation and output directives.

An example of a template is depicted in Fig. 4. It shows a (simplified) fragment of a template that generates a schema definition for a complex type. The template has a parameter n that is of type *Class*. In the figure, each language is presented in a different style: The query language OCL is depicted bold and slanted, the template language is depicted solid underlined, and the target language is shown in a normal font. As one can see, between the target language output of the XML tags `<xs:complexType></xs:complexType>` the model element n is queried for its name if it is enriched with the stereotype «xsdNamedType». Furthermore, the complex type is defined as abstract if the model element identified by n is abstract.

```
<xs:complexType>
#if($ocl.eval("n.extension_xsdNamedType.isDefined()"))
  name='$ocl.eval("n.name")'
#end
#if($ocl.eval("n.isAbstract"))
  abstract='true'
#end
>...</xs:complexType>
```

Figure 4: Illustrative excerpt of a Transformation Template

Most of the XÖV projects make use of the possibility to generate documentation from the model. The documentation generation produces DocBook fragments that can be integrated into a manually written specification. These manually written parts 'glue' together the technical parts with surrounding prose (text not directly related to model elements) to finally get a human readable specification.

To give an impression of the amount of generated artifacts, we provide numbers from the current release of *OSCI–XMeld*: In total ≈ 1200 documentation fragments (DocBook sections), 16 schema files, 21 WSDL files and 54 code lists are generated.

3.2.3. Post-Transformation Validation

The general objective is that the transformation templates generate well-formed output for any well-formed input model (i.e., for any input model that fulfills all *must* level rules). As an additional safety net (in particular for transformation development), the XGenerator performs a specific well-formedness validation for the generated results for several formats. In general, all XML outputs are verified against their schemas. For specific target languages such as XML Schema, more powerful well-formedness checkers have been installed.

This safety net guarantees that even if an invalid input model slips through the well-formedness rules (or if a valid input model is not properly handled by the transformation templates), no ill-formed output artifacts can be generated unnoticed.

3.3. XRepository

The *XRepository* [54] is a central, moderated service for all those involved in the planning and development of electronic data interchange in the German public administration. The open repository provides a structured means to publish, browse, search, share, retrieve, and subscribe to semantic components. It implements the Asset Description Metadata Schema (ADMS) specification [16] in order to enable the interchange with other national and international repositories.

The focus of the repository is on the semantic components related to XÖV projects and specifications. Its content is provided mainly by the community of projects themselves. In order to be certified, the use of the repository is mandatory (although the repository also addresses projects that do not aim for full certification). In particular, the specification (documentation and technical artifacts) and the specification model must be published in there. The moderation supports the goal that already modeled concepts can be found and reused, respectively that deviations from this goal are justified and made transparent.

At its core, the repository stores versioned content of several types. Among these content types XÖV specifications and their models (in UML 2 XMI) can be found. The objects can be further described by metadata and cross-references to other semantic components in the repository. In particular, the level

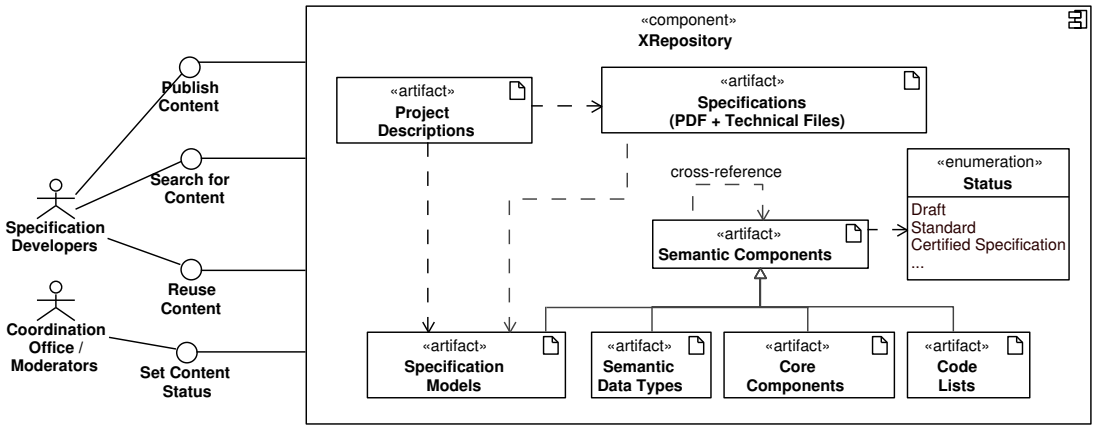

Figure 5: The XRepository component

of quality and standardization can be described for each object, from 'Draft' (the lowest level) to 'XÖV-certified' and 'Standard' (the highest levels for XÖV and non-XÖV specifications – only the operator of the repository can assign this quality level to objects in the repository). Figure 5 depicts the content stored by the repository. The supported content types include: *XÖV project* (a structured portrait of the project, stating its objectives and owners); *XÖV specification* (including the specification document and the technical artifacts); *Code list*; *Core component*; and *Semantic Data Type*. Except for *XÖV project*, all content types are available as UML models. The main interface to the XRepository is a web user interface. In addition, a web service interface is provided to automatically process certain content types (such as code lists).

4. Discussion

After one decade of model-driven specification development in the e-government domain, we state that it is an ongoing success story. It is, however, important to notice that the success of e-government projects like the ones addressed in our context is first of all determined by non-technological aspects. In this context, a technological approach like the one presented here can only be supportive. Thus, we restrict our discussion to the technological contribution. From our point of view we have to consider two questions about our application of MDE:

1. Has it been *effective* – Could we realize the objectives this way?
2. Has it been *better* than alternative approaches?

In short, we provide arguments to answer both questions positively. However, as we did not conduct a controlled experiment to compare the model-driven approach with a conventional one on a realistic scale, we do not provide a quantitative empirical evaluation.

As far as the technical aspects of specification development are concerned, the following points are common objectives for XÖV specification projects:

1. The specification should comprehensively describe the intended data interchange in such a way that its correctness from a business perspective and its conformance to legal foundations and privacy policies can be validated prior to implementation;
2. it must be consistent with respect to the documentation and the technical artifacts;
3. it must conform to the technical XÖV conformance rules;
4. it should follow the XÖV conformance rules on the semantic level (i.e., it should use common components whenever possible);
5. and, of course, it should to be developed and maintained efficiently.

It has to be emphasized that the first two points are particularly important in those cases, where the public administration does not only recommend a specification but also makes it legally binding: In XÖV, the time span between the publication of the specification and going into service is between 9 and 18 months (as the software vendors require some time to implement the new or modified interfaces). Thus, critical errors that are detected after publication will cause significant problems.

4.1. XÖV Specifications Already in Operation

By the time of writing, several XÖV projects have been successfully completed using the presented method and tools. In total, XRepository lists 17 XÖV projects (i.e., projects that aim towards an XÖV conforming specification). Table 2 shows the eight specifications that have already been successfully certified by the coordination office at the time of writing. All specifications were created by independent projects in the responsibility of different bodies of the public administration (for most of them a federal ministry). We include the year when the first official version of each specification has been released to indicate their age and maturity. It has to be emphasized that most of the specifications are subject to constant extension and maintenance, due to a growing number of legal use cases that had to be included. For example, since its first release in 2002, 13 subsequent versions of *OSCI–XMeld* have been released (with 6 or 12 month intervals in between). The situation is similar for other specifications. This fact by itself provides some evidence for the effectiveness of our model-driven way of development.

To give the reader an impression of the operational and structural complexity of the specification, we included three numbers in Table 2: The number of messages counts the top level classes that model one data-interchange step as part of a communication scenario. A typical request-response process requires two message classes, more complex processes require up to ten different classes. The total number of classes and the average number of data fields per message reflect the structural complexity of the specification. If we consider furthermore the high transmission volumes, numbers of communication partners, and the heterogeneous environment of vendors, it should become obvious that these specifications are no MDE toy examples.

4.2. Success Factors

We have identified four aspects in which we consider our model-driven approach to support the development of XÖV specifications significantly better than a conventional, not model-driven approach: (1) early validation; (2) achieving XÖV conformance; (3) consistency among the technical artifacts; and (4) efficiency. We discuss these aspects in the following.

4.2.1. Feasibility of Early Validation

While this is a generic argument for every phased software development process and a common motivation for model-driven software engineering, it very particularly applies to the development of XÖV specifications. Given a heterogeneous environment of several vendors and various mission critical administrative processes, it has been imperative to closely collaborate with experts from the legal, the administrative, and the technical domain (i.e., designated implementers and operators). Translating the legal specifications into practical, operational specifications is by no means a mechanical task, but must be regarded as a large refinement step. To perform this step in the chosen modeling paradigms of processes (as activity diagrams) and messages (as compositional class diagrams) has turned out to be an appropriate level of abstraction, compared to a purely data-centric specification approach.

Many validation tasks can be carried out at this level. Several projects developed extensive test suites of process instantiations (e.g., sequences of messages sent between communicating authorities) at this modeling level. It has to be emphasized that, referring to *OSCI–XMeld* as an example again, very few severe bugs passed the quality assurance phase in the sense that they were not detected before actually implementing the published specification. In particular, the modeling approach supports the stakeholders in validating the conformance to the legal foundations. As we can furthermore rely on the fact that the technical fragments will actually contain exactly the data fields that are defined in the UML models, the XÖV method is in accordance with the transparency principle of the European Interoperability Framework [15].

Specification	First version	Messages	Classes	Average attributes / message	Description
OSCI–XMeld	2002	159	708	88	Data-interchange processes regarding the citizens place of residence between the municipal citizen registers, tax, policy, pension, and other authorities, with several hundred thousand messages being sent every day; the specification is implemented in their products by more than 10 different vendors, and it has been made legally binding
XAusländer	2007	51	289	66	Data-interchange processes between the approx. 600 authorities responsible for foreign citizens; the specification is implemented in their products by more than 12 different vendors and it has been made legally binding
XPersonen-stand	2007	83	327	99	Data-interchange processes regarding changes in the family status (e.g., marriage, birth of children); replaces approx. 10 million paper-based communications per year between the municipal and federal authorities; the specification has been made legally binding
xFall	2010	11	74	37	Defines processes and messages for the adoption of the European directive 2006/123/EC on services in the internal market
XDomea	2007	66	207	70	Standardizes processes and messages for the interchange of files and administrative transactions among authorities
XZUFI	2011	24	139	20	Data-interchange between authority and authority service registries
XStatistik	2010	2	89	144	Unified transmission of statistical data to the central statistics authority
XWaffe	2010	38	131	40	Specifies the processes necessary to build up a national weapon register

Table 2: Specifications currently certified; numbers referring to the latest version

Our observation is that extensively using models prior to realization has been an important success factor in our setting.

4.2.2. Ensuring and Verifying XÖV Conformance

From the perspective of interoperability, it is important that conformance to the XÖV framework can be actually verified in practice. This is important both for the coordination office (which has to be able to perform the verification with reasonable effort for every new version of each specification) and for the projects themselves.

Many of the technical XÖV conformance rules are realized as OCL constraints that can be verified automatically with no further effort using the XGenerator. Currently, most of the rules that can be verified automatically regard the conformance of the structure of the model, the conformity of the schema design (naming and design rules), the correct application of structured documentation, and the syntactically correct usage of shared objects. Several rules do not even need to be actively checked (by well-formedness rules), but they are implicitly enforced by the transformation to XML Schema, making it impossible to deviate from the rules.

Other, semantic, conformance aspects cannot be checked automatically. This regards, for example, whether information entities were 're-invented' without need, because a recommended standard already

exists for them. Nevertheless, the uniformly structured model and the cross-referencing to shared semantic components in the models provide a practical, integrated representation of the specification to check these rules.

4.2.3. Consistency among the Specification Artifacts

The XGenerator and its transformation rules guarantee consistency between the model, the corresponding fragments of the specification document, the XML schemata, and the web service descriptions. Notice that we assume the correctness of the transformation rules *de facto*, as the results have been implemented hundreds of times already, and a transformation test suite has been developed that can automatically check the transformation results for regression on the base of canonically comparing the XML files of XÖV specifications.

In a manual development approach, the consistency between the various artifacts must be maintained manually, which implies the risk of undetected inconsistencies and significantly increases the effort required for changes to the specification.

In a non-MDE solution, consistency between schema and web service descriptions could be managed technically (e.g., using automated replication), but the consistency between the documentation and the technical artifacts would be less trivial as soon as more is intended than just having graphical depictions of the Schema inside the documentation. In the long term, the risk is evidently that updates in the one document are not accordingly propagated to the others. Explicit change propagation approaches [25, 3], in particular cross-languages ones [45, 2], might be appropriate to tame this situation – which, however, does not arise in the first place in our approach.

4.2.4. Project Initialization, Time to Market, and Efficiency

XML Schema and WSDL are both complex languages with various design styles, patterns, and anti-patterns. The XÖV standard aims towards a robust, uniform design of specification on the technical level (Schema and WSDL) that follows accepted design rules. Thus, the XÖV conformance rules itself relieve the project from various design decisions and efforts (and from the necessity to have 'XML gurus' in the project team). Nevertheless, even the XÖV rules have to be read, understood, applied, and their application must be verified.

In our approach, the conformance to these rules is ensured both by model validation (checking that the platform-specific choices are used consistently) and by model transformation (that, for example, always generate XÖV-conforming schemas from well-formed input models). The conformance to the design rules is achieved or at least validated mostly automatically, and common traps related to XML-Schema can be avoided. Furthermore, a standard structure for a specification model is ensured, providing a guard rail for new projects to follow.

In summary, this enables new projects to quickly concentrate on their actual matters, relieves the development team from focusing too much on technical aspects, and eventually leads to shorter time to market. This fact has been confirmed several times by responsible developers, e.g., at the regular XÖV user conferences [33, 20].

4.3. Areas of Improvement

Although our approach is in operation for a large number of projects already (and, thus, only careful and controlled changes can be made to the approach), there are two aspects which we want to improve in the future:

First, to our knowledge, the software vendors that implement XÖV specifications base their work on the PDF documentation and on the released technical files. We hope to convince vendors to also take advantage of the UML models directly, e.g., to generate program code.

Second, the XGenerator currently addresses only few aspects of the process models (i.e., of the UML interactions) in its model validation and model transformation tasks. This is not a general limitation of the tool, which can process the whole UML 2 metamodel, but simply an aspect which so far is not regarded in much detail in the conformance rules themselves. In the future version we plan to extend the treatment of

process models both in the conformance rules and in their representation in the model-driven method. In particular, the consistency between the messages and the process models will be checked more thoroughly.

5. Related Work

From the perspective of e-government standardization, there are several national approaches that can be compared to XÖV. As for XÖV, their technological aspects must be seen as part of the e-government strategy and the landscape, and an in-depth discussion goes beyond the scope of this article. We focus on the technological MDE perspective in this section and refer the reader to the surveys of Charalabidis et al. [11] and Guijarro [21, 24] and to the NIFO factsheets of the European Commission [38] for a general comparison of several e-government interoperability frameworks, and to the survey of [44] for a more specific review of the role of modeling in such, typically national, frameworks. In general terms, the concepts of shared semantic components, interoperability rules, and central repositories is common to several of them.

Regarding the XRepository, the various relationships (e.g., composes, transforms to) between the semantic components, models, and specification can be viewed as a specific kind of megamodel [19, 31] that has explicit notions of versioning and maturity levels for the individual semantic components.

5.1. Related Approaches to XML Schema Modeling

There are a number of approaches that generate XML Schema from (profiled) UML models. Profiles such as the one described by Bernauer [5] allow to fully specify all details of an XML schema in UML. On the opposite end, approaches such as XUML [35], UML-to-GML [27, 26, 23], and the work of Domínguez et al. [12] provide no or only very limited choices for the schema generation (and thus achieve a high level of abstraction). Domínguez provides a survey covering several UML-to-Schema approaches [13]. In order to support legacy specifications, the UML profile for XÖV specifications had to provide several choice points for XML schema (but far less than, e.g., Bernauer). It recommends, however, to use the default settings (i.e., not to apply the respective stereotypes and properties). Similar arguments hold for the generation of WSDL files, e.g., when compared to UML profiles such as the one of Vara et al. [50].

It is worth mentioning that there is a request for proposals of the OMG for a UML profile for the US National Information Exchange Model (NIEM) to transform the logical UML model to NIEM conformant XML Schema [42].

5.2. Related Model Transformation Approaches

Following the model transformation taxonomy defined in [36] our approach uses exogenous, vertical transformations to generate the various outputs. We directly use model-to-text transformations. Alternatively, we could have generated platform-specific models (e.g., for XML Schema) using model-to-model (M2M) transformations like QVT [41], ATL [30], the Epsilon Transformation Language [32], and Kermeta [29]. In our view, both directions are viable ways for our setting. The M2T approach is more lightweight in the sense that it does not require to have explicit metamodels for the target languages. The potential drawbacks are that no further transformation steps can be applied on the results (something that has not been relevant for us) and that no model-based validation can be applied on the results (i.e., by checking the well-formedness rules of the output metamodels). We compensate the latter restriction by putting explicit validation tools for XML files in general and XML Schema in particular.

Regarding the M2T language, our solution is proprietary, although it follows the same pattern as the OMG standard MOF Model to Text [40], MOFScript[37], and Xpand [53]. Being based on the Ecore metametamodel and OCL queries, we expect that it could be replaced by, e.g., the open source Acceleo plugin for Eclipse [1]. However, no suitable and mature implementations have been available when we designed the XGenerator tool.

6. Conclusion

In this article we have reported how the model-driven paradigm has been successfully applied as a method for developing standardized XML-based e-government data interchange specifications (XÖV specifications). Unlike most MDE projects, we focus on generating interfaces (typically for web services) that have to be implemented by software vendors. We do not generate actual software. Emanated from an early adopter project, the data exchange between the 5,400 municipal citizen registers in Germany, the approach is now part of an official, coordinated strategy, named XÖV, for a community of currently 17 e-government projects in a challenging heterogeneous context of multiple vendors and domains.

We have presented how profiled models, tools for automated model validation and transformation, and a central repository compose a system for the development of XÖV specifications. Most of the 'basic MDE ingredients' that we employ are standard, such as UML profiles and OCL well-formedness rules. The model validation and transformation engine used has been developed specifically for our needs but could be replaced by off-the-shelf solutions in other projects.

While the overall success of the XÖV e-government projects must be first credited to non-technical factors (such as semantic alignment of the legal foundations), we have discussed how our model-driven approach has nevertheless contributed to the success. Its main advantages are that it facilitates the (early) validation of conformance to the standard for XÖV specification, that is fosters the semantic alignment of co-existing standards, and that it supports a short time to market by providing abstraction from and automated generation of technical specification assets.

In summary, our report provides support for the statement that the model-driven approach is well-suited for the development of standardized interface specifications in a heterogeneous environment.

We see three fields were we like to further improve our approach in the future: First, we hope to convince the software vendors to take more advantage directly of the specification model, to perform model-driven tasks on the software development side, too. Second, we plan to extend the amount of behavioral model validation, in order to check more aspects of the consistency of the specification model. Finally, we expect that we will extend our model-driven approach in the context of further aligning the XÖV framework with the ongoing e-government framework initiatives on the European level and beyond, such as the European Interoperability Framework 2.0 ([14, 15]) and the e-government Core Vocabularies ([17]).

References

[1] Acceleo, 2012. Acceleo homepage. `http://www.eclipse.org/acceleo/`, accessed: 12. Oct. 2012.

[2] Anderson, K. M., Taylor, R. N., Jr., E. J. W., 2000. Chimera: hypermedia for heterogeneous software development enviroments. ACM Trans. Inf. Syst. 18 (3), 211–245.

[3] Arnold, R., Bohner, S., 1996. Software Change Impact Analysis. IEEE Computer Society.

[4] Bartels, U., Steimke, F., 2004. How to Modernize the People Registration Process: Experiences in the Leading e-Government Project in Germany. In: Traunmüller, R. (Ed.), Electronic Government: Third International Conference, EGOV 2004, Zaragoza, Spain, Proceedings. Vol. 3183 of LNCS. pp. 246–249.

[5] Bernauer, M., Kappel, G., Kramler, G., 2004. Representing XML Schema in UML - A Comparison of Approaches. In: Koch, N., Fraternali, P., Wirsing, M. (Eds.), Web Engineering - 4th International Conference, ICWE 2004, Proceedings. Vol. 3140 of LNCS. Springer, pp. 440–444.

[6] Bradner, S., 1997. Key words for use in RFCs to Indicate Requirement Levels. The Internet Engineering Task Force (IETF), RFC 2119.

[7] Büttner, F., Gogolla, M., 2011. Modular Embedding of the Object Constraint Language into a Programming Language. In: Simao, A., Morgan, C. (Eds.), Proc. 14th Brazilian Symposium on Formal Methods (SBMF'2011). Springer, Berlin, LNCS 7021, pp. 124–139.

[8] Bundesvereinigung der Kommunalen Spitzenverbände (Ed.), 2012. Datensatz für das Meldewesen. Einheitlicher Bundes-/Länderteil (DS Meld). Stand: November 2012. Deutscher Gemeindeverlag, "German for: data fields for municipal citizen registers, uniform part for the federal and state levels".

[9] Bundesverwaltungsamt - Bundesstelle für Informationstechnik, 2012. XGenerator 2.2.1. Homepage, `https://joinup.ec.europa.eu/software/xgenerator/home`.

[10] CCTS, 2009. Core Component Technical Specification (CCTS) Version 3.0. United Nations Centre for Trade Facilitation and Electronic Business (UN/CEFACT).

[11] Charalabidis, Y., Lampathaki, F., Kavalaki, A., Askounis, D., 2010. A review of electronic government interoperability frameworks: patterns and challenges. International Journal of Electronic Governance 3 (2), 189–221.

[12] Domínguez, E., Lloret, J., Pérez, B., Rodríguez, Á., Rubio, A. L., Zapata, M. A., 2011. Evolution of xml schemas and documents from stereotyped uml class models: A traceable approach. Information & Software Technology 53 (1), 34–50.
[13] Domínguez, E., Lloret, J., Pérez, B., Rodríguez, u., Rubio, n., Zapata, M., 2007. A Survey of UML Models to XML Schemas Transformations. In: Benatallah, B., Casati, F., Georgakopoulos, D., Bartolini, C., Sadiq, W., Godart, C. (Eds.), Web Information Systems Engineering - WISE 2007. Vol. 4831 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg, pp. 184–195.
[14] EC, 2010. Annex I to the Commission communication on interoperability – European Interoperability Strategy (EIS), COM(2010) 744 final. The European Commission, Interoperability solutions for European public administrations (ISA) program.
[15] EC, 2010. Annex II to the Commission communication on interoperability - European Interoperability Framework (EIF). The European Commission, Interoperability solutions for European public administrations (ISA) program.
[16] EC, 2012. ADMS – Assert Description Metadata Schema Specification 1.00. The European Commission, Interoperability solutions for European public administrations (ISA) program.
[17] EC, 2012. eGovernment Core Vocabularies v.2. The European Commission, Interoperability solutions for European public administrations (ISA) program.
[18] EMF, 2012. Eclipse Modeling Framework Project (EMF). Homepage, http://www.eclipse.org/modeling/emf/.
[19] Favre, J.-M., Nguyen, T., 2005. Towards a megamodel to model software evolution through transformations. Electr. Notes Theor. Comput. Sci. 127 (3), 59–74.
[20] Franz, S., 2011. XKfz: Erfahrungen eines XÖV-Novizen. In: [33], slides.
[21] Gil-García, J. R., Pardo, T. A., 2005. E-government success factors: Mapping practical tools to theoretical foundations. Government Information Quarterly 22 (2), 187 – 216.
[22] Gogolla, M., Büttner, F., Richters, M., 2007. USE: A UML-Based Specification Environment for Validating UML and OCL. Science of Computer Programming 69, 27–34.
[23] Grønmo, R., Solheim, I., Skogan, D., 2002. Experiences of UML-to-GML Encoding. In: 5th AGILE Conference on Geographic Information Science: Palma (Balearic Islands, Spain). pp. 1–13, available online, http://itcnt05.itc.nl/agile_old/Conference/mallorca2002/proceedings/dia25/Session_2/s2_Gronmo.pdf.
[24] Guijarro, L., 2007. Interoperability frameworks and enterprise architectures in e-government initiatives in Europe and the United States. Government Information Quarterly 24 (1), 89 – 101.
URL http://www.sciencedirect.com/science/article/pii/S0740624X06000864
[25] Hassan, A. E., Holt, R. C., 2004. Predicting change propagation in software systems. In: 20th International Conference on Software Maintenance (ICSM 2004), 11-17 September 2004, Chicago, IL, USA. IEEE Computer Society, pp. 284–293.
[26] Interactive Instruments, 2012. ShapeChange. Homepage, http://www.interactive-instruments.de/ugas/, accessed: 12. Oct. 2012.
[27] ISO, 2007. Iso 19136:2007 geography markup language (gml), annex e: Uml-to-gml application schema encoding rules.
[28] IT-Planungsrat, 2010. National e-government strategy. Published online. http://www.it-planungsrat.de.
[29] Jézéquel, J.-M., Barais, O., Fleurey, F., 2009. Model driven language engineering with kermeta. In: Fernandes, J. M., Lämmel, R., Visser, J., Saraiva, J. (Eds.), Generative and Transformational Techniques in Software Engineering III - International Summer School, GTTSE 2009, Braga, Portugal, July 6-11, 2009. Revised Papers. Vol. 6491 of Lecture Notes in Computer Science. Springer, pp. 201–221.
[30] Jouault, F., Allilaire, F., Bézivin, J., Kurtev, I., 2008. ATL: A model transformation tool. Sci. Comput. Program. 72 (1-2).
[31] Jouault, F., Vanhooff, B., Bruneliere, H., Doux, G., Berbers, Y., Bézivin, J., 2010. Inter-dsl coordination support by combining megamodeling and model weaving. In: Shin, S. Y., Ossowski, S., Schumacher, M., Palakal, M. J., Hung, C.-C. (Eds.), Proceedings of the 2010 ACM Symposium on Applied Computing (SAC), Sierre, Switzerland, March 22-26, 2010. pp. 2011–2018.
[32] Kolovos, D. S., Paige, R. F., Polack, F., 2008. The epsilon transformation language. In: Vallecillo, A., Gray, J., Pierantonio, A. (Eds.), Theory and Practice of Model Transformations, First International Conference, ICMT 2008, Zürich, Switzerland, July 1-2, 2008, Proceedings. Vol. 5063 of Lecture Notes in Computer Science. Springer, pp. 46–60.
[33] KOSIT, 2011. Vorträge der 4. XÖV-Anwenderkonferenz (presentations of the fourth XÖV user conference).
[34] KOSIT, 2012. Handbuch zur Entwicklung XÖV-konformer IT-Standards, Version 1.1. Koordinierungsstelle für IT-Standards (KoSIT), Die Senatorin für Finanzen Bremen, published online. http://www.xoev.de/sixcms/media.php/13/XÖV-HandbuchV1_1.pdf.
[35] Liu, H., Lu, Y., Yang, Q., 2006. XML conceptual modeling with XUML. In: Osterweil, L. J., Rombach, H. D., Soffa, M. L. (Eds.), 28th International Conference on Software Engineering (ICSE 2006). ACM, pp. 973–976.
[36] Mens, T., Gorp, P. V., 2006. A taxonomy of model transformation. Electronic Notes in Theoretical Computer Science 152 (0), 125 – 142, proceedings of the International Workshop on Graph and Model Transformation (GraMoT 2005).
URL http://www.sciencedirect.com/science/article/pii/S1571066106001435
[37] MOFScript, 2012. MOFScript homepage. http://eclipse.org/gmt/mofscript/, accessed: 12. Oct. 2012.
[38] NIFO, 2012. National Interoperability Framework Observatory (NIFO) Factsheets. Available online, http://joinup.ec.europa.eu/elibrary/factsheet/national-interoperability-framework-observatory-nifo-factsheets.
[39] OASIS, 2007. Code List Representation (Genericode) Specification, Version 1.0. Available online, http://docs.oasis-open.org/codelist/cs-genericode-1.0/doc/oasis-code-list-representationgenericode. pdf.
[40] Object Management Group (OMG), 2008. MOF Model to Text Transformation Language, v1.0. Available online, http://www.omg.org/spec/MOFM2T/1.0/.
[41] Object Management Group (OMG), 2011. Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification,

Version 1.1.
URL http://www.omg.org/spec/QVT/1.1/
[42] Object Management Group (OMG), 2011. UML Profile for NIEM (NIEM-UML), Request For Proposal. OMG Document gov/2011-06-04. Available online, http://www.omg.org/cgi-bin/doc?gov/2011-06-04.
[43] OMG, 2012. Object Constraint Language Specification, version 2.3.1 (Document formal/2012-01-01).
[44] Peristeras, V., Tarabanis, K. A., Goudos, S. K., 2009. Model-driven egovernment interoperability: A review of the state of the art. Computer Standards & Interfaces 31 (4), 613–628.
[45] Pfeiffer, R.-H., Wasowski, A., 2012. Cross-language support mechanisms significantly aid software development. In: France, R. B., Kazmeier, J., Breu, R., Atkinson, C. (Eds.), Model Driven Engineering Languages and Systems - 15th International Conference, MODELS 2012, Innsbruck, Austria, September 30-October 5, 2012. Proceedings. Vol. 7590 of Lecture Notes in Computer Science. Springer, pp. 168–184.
[46] SAGA, 2011. SAGA-Modul Technische Spezifikationen, version 5.0.0. Die Beauftragte der Bundesregierung für Informationstechnik (BfIT), Bundesministerium des Innern.
[47] Stantchev, V., Schoenherr, M., Dietrich, J., 2009. Layered Government and E-Citizenship: Objectives and Technical Challenges in the EU. In: Internet and Web Applications and Services, 2009. ICIW '09. Fourth International Conference on. pp. 614 –619.
[48] Steinberg, D., Budinsky, F., Paternostro, M., Merks, E., 2008. EMF: Eclipse Modeling Framework, 2nd Edition. Addison-Wesley Longman, Amsterdam.
[49] USE, 2012. A UML-based Specification Environment. Homepage, http://sourceforge.net/projects/useocl/.
[50] Vara, J., de Castro, V., Marcos, E., 2005. WSDL automatic generation from UML models in a MDA framework. In: Next Generation Web Services Practices, 2005. NWeSP 2005. International Conference on. p. 6 pp.
[51] Velocity, 2012. The Apache Velocity Project. Homepage, http://velocity.apache.org/.
[52] Warmer, J., Kleppe, A., 2003. The Object Constraint Language: Getting Your Models Ready for MDA. Object Technology Series. Addison-Wesley, Reading/MA.
[53] Xpand, 2012. Xpand homepage. http://wiki.eclipse.org/Xpand, accessed: 12. Oct. 2012.
[54] XRepository, 2012. XRepository. Homepage, http://www.xrepository.de.