MDA Employed in a Joint eGovernment Strategy: An Experience Report

Fabian Büttner[1], Mirco Kuhlmann[1], Martin Gogolla[1], Jens Dietrich[2], Frank Steimke[2], Andre Pankratz[3], Alina Stosiek[3], and Alexander Salomon[4]

[1] University of Bremen, Computer Science Department, Database Systems Group
{green,mk,gogolla}@tzi.de
[2] OSCI Leitstelle, Senator für Finanzen, Bremen
{jens.dietrich,frank.steimke}@finanzen.bremen.de
[3]]init[Aktiengesellschaft für digitale Kommunikation, Berlin
{andre.pankratz,alina.stosiek}@init.de
[4] Bundesministerium des Innern, Referat IT2 (KBSt), Berlin
alexander.salomon@bmi.bund.de

Abstract. DOL (Deutschland OnLine, Germany online) is a joint eGovernment strategy in Germany. Its objectives include the development of standards for electronic data exchange between public authorities. To achieve this goal, DOL projects are employing UML models, profiles, and tools: A cross-project conceptual model and a profile for common core components are used to harmonize the different domain specific models, and a further DOL specific UML profile has been developed to attach data transmission details to the domain specific models. The automated creation of data exchange specifications from these platform-specific models is realised by an MDA tool developed in DOL: The XGenerator allows the validation of UML models against applied profiles by checking OCL constraints, and the XGenerator supports a flexible, MDA based generation of XML Schema definitions, DocBook based documentation, and WSDL descriptions. This paper reports on how UML profiles and formal MDA techniques have been assembled into a reusable architecture for the development of data exchange specifications.

1 Overview

DOL (Deutschland OnLine, Germany online) is a joint eGovernment strategy by the federal government, federal-state governments, and municipalities in Germany [DOL08]. Its objectives include the provision of a standardisation infrastructure to support the development and deployment of semantic standards for electronic data exchange between and with public authorities, so-called XOEV standards (XML in der OEffentlichen Verwaltung, XML in public authorities). Several concrete ongoing XOEV projects (including civil status registration, data exchange for immigration offices, data exchange in the judiciary domain, and vehicle registration) are currently realised as DOL projects on top of this infrastructure. The pioneer project XMELD [XME08] (XML im MELDewesen, XML for municipal citizens registration) has been successfully finished in time and budget thanks to extensive use of model driven techniques. Several similar projects have been started under the roof of DOL as well.

The model-driven standardisation infrastructure of DOL employs UML models, profiles, and tools to provide a unifying production chain for XOEV standards: A core components model is being developed which collects basic information entities that are common in several areas of the public administration (e.g., entities like Name or Address), enabling interoperable processes across administrative domains. This reusable conceptual model is based on the concepts of the UN/CEFACT Core Components specification [UN03], aligning the standardisation approach of DOL with standardisation efforts in other European countries, such as France [SYN08].

Automated profile validation is used to ensure correct usage of core components in the domain models of XOEV projects. The MDA tool XGenerator has been developed in DOL to perform this profile validation in addition to the transformation into XOEV standards: These XOEV standards (currently consisting of documentation, XML Schema files, and – in some projects – WSDL files) are generated from domain models in an MDA-based way. A DOL specific UML profile, the XOEV UML profile, has been developed to attach platform-specific information to the platform-independent domain models. This additional information is used by the tool XGenerator to produce the various parts of an XOEV specification from an annotated UML model in a configurable way. This means that the XGenerator is both a profile validator and a flexible model transformation tool. OCL is a central ingredient in this production chain. It is employed to describe the well-formedness rules in both profiles and as a model query language within the transformation language of the XGenerator.

The model-driven architecture of DOL, including the XOEV profile and our tool XGenerator, has been developed in cooperation between government, public authorities, industry, and academia. The DOL project shows a successful transfer of MDA technology into real world projects. This paper reports on the interplay between the core components model, domain models, both UML profiles (the Core Components profile and the XOEV profile), and our tool XGenerator within the DOL architecture. The XGenerator has been released under the GNU public license on [SF08] and we encourage employment in other projects.

The remaining paper is structured as follows: Section 2 starts with a short bird's eye view on the DOL project, explaining its overall objectives and introducing its standardisation infrastructure. Section 3 explains the relationship between the employed UML models and profiles. Section 4 presents the tool XGenerator. In Sect. 5, we conclude with our thoughts regarding the employment of models and tools in DOL and present future directions.

2 A Bird's Eye View on DOL

The goal of DOL is to focus on the improvement of eGovernment processes with high importance to all levels of administration. The overall project structure of DOL is depicted in Fig. 1. Several XOEV projects focus on the improvement of concrete eGovernment processes (municipal citizens registration, civil status registration, data exchange for immigration offices, data exchange in the judiciary domain, and vehicle registration). Two projects provide enabling infrastructure for eGovernment processes wrt. network infrastructure and standardisation infrastructure. Apart from the running XOEV projects in Fig. 1 there are several further projects underway to improve other domains.

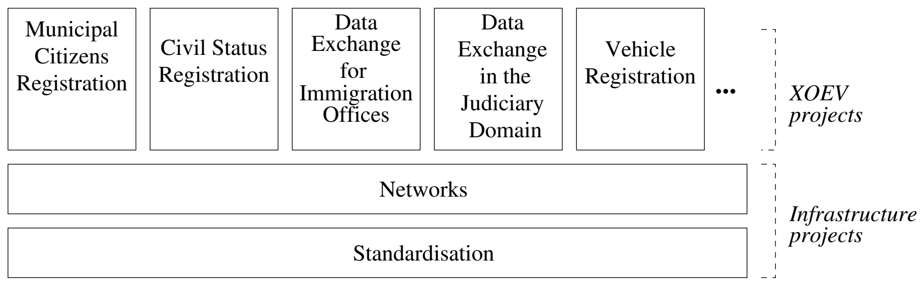

Fig. 1. Overall Project Structure of DOL

Most eGovernment processes cover different levels of administration and therefore require electronic processes between public authorities. Accordingly the goal of the standardisation infrastructure project within DOL is to provide an infrastructure that supports efficient development and deployment of semantic standards for electronic data exchange between administrations (Government-to-Government) and between administrations and their customers like companies (Government-to-Business). These semantic standards are called XOEV standards and are developed by XOEV projects.

The XOEV projects are considered as customers of the standardisation infrastructure project. Therefore the DOL project provides support for these projects in developing successful XOEV standards. This support includes the provision of common rules resp. methodologies (like UML profiles), technical infrastructure resp. tools (like a standards repository and the MDA tool XGenerator, and reusable conceptual models, so-called core components).

An important task of XOEV projects is to specify the electronic processes between communication partners describing which data has to (resp. is allowed to) be transmitted and when. In almost all projects, these decisions have to be made in accordance with existing laws. All this information is collected in UML models. The resulting conceptual data transmission models have to be finally translated into precise specifications (published by the respective authorities). These specifications, containing XML schemas, documentation, and WSDL files, are required by tool vendors later on to implement interoperable solutions for the public administration.

The development of such a specification is time and resource consuming. Therefore, in addition to concrete technological recommendations (e.g., for using XML on top of the secure OSCI Transport [OSC02] protocol), the initiative also aims to standardise the definition process of electronic data exchange specifications.

To achieve this goal, DOL provides a model-driven project architecture. This architecture contains several reusable modeling components as explained in the following section.

3 Production Chain for XOEV specifications — PIMs, PSMs, and Generated Artifacts

A conceptual (i.e., platform independent) model of the data that has to be transmitted is an essential requirement in each eGovernment project. This is especially true for XOEV projects which require precise specification of data and processes. The DOL participants early recognized that several concepts are common across all public authority domains. In accordance with the UN/CEFACT [UN03], these are information *core components* in DOL. Examples of core components are the name of a natural person, and the address. These core concepts can be reused in several concrete projects by derivation.

Within DOL, a so-called "data conference" works on a regularly basis to identify core components using the respective legal bases of all DOL projects. The data conference generalises and maintains these core components and collects them as a UML model. Core components from this model can be reused as information entities in concrete projects. The UN/CEFACT Core Components Profile (CC profile) [UN06] is employed to ensure correct derivation.

Figure 2 shows the dependencies between the participating UML models within the production chain for XOEV specifications. The core components model is situated in the common infrastructure part. When applicable, core components are reused in the domain model of an XOEV project. The remaining parts of the model are project-specific. The domain modeling is performed by the project's domain working group. The way core components can be reused is defined by the CC profile. This ensures that the reused components remain interoperable and simplifies the implementation of cross-project administrative processes. The XGenerator tool is able to support the modeler in the correct reuse of core components by validating the OCL well-formedness rules of the CC profile (described in detail in Sect. 4, as a forward reference for the tool see Fig. 5).

Finally, this conceptual model has to be casted into a technical specification that can be delivered to system vendors. On the technical level, this means that the structures of the conceptual model have to be translated into XML schemas. Further, XML namespaces have to be defined, digital signature information has to be added, and the messages have to be distributed as services (WSDL files). An XOEV specification typically consists of one specification document (PDF) and several accompanying XML Schema and WSDL files. (E.g. for XMELD 892 pages of documentation, 421 complex schema types, and three web service descriptions).

Due to the model-driven approach, large parts of these specification artifacts can be generated from the UML model. A second, DOL-specific UML profile, the XOEV profile, has been built in DOL to incorporate the necessary technical details into the model. The XOEV profile also defines several OCL constraints that prohibit wrong application of stereotypes and enforce modeling standards (e.g., naming rules). The domain model is enriched with stereotypes and properties of the XOEV profile so that a platform specific model (PSM) comes up that can be used to derive the specification parts depicted in the bottom of Fig. 2.

The XGenerator is used to formally check the completeness of the PSM and to validate the adherence with the OCL well-formedness rules of the XOEV profile. Having

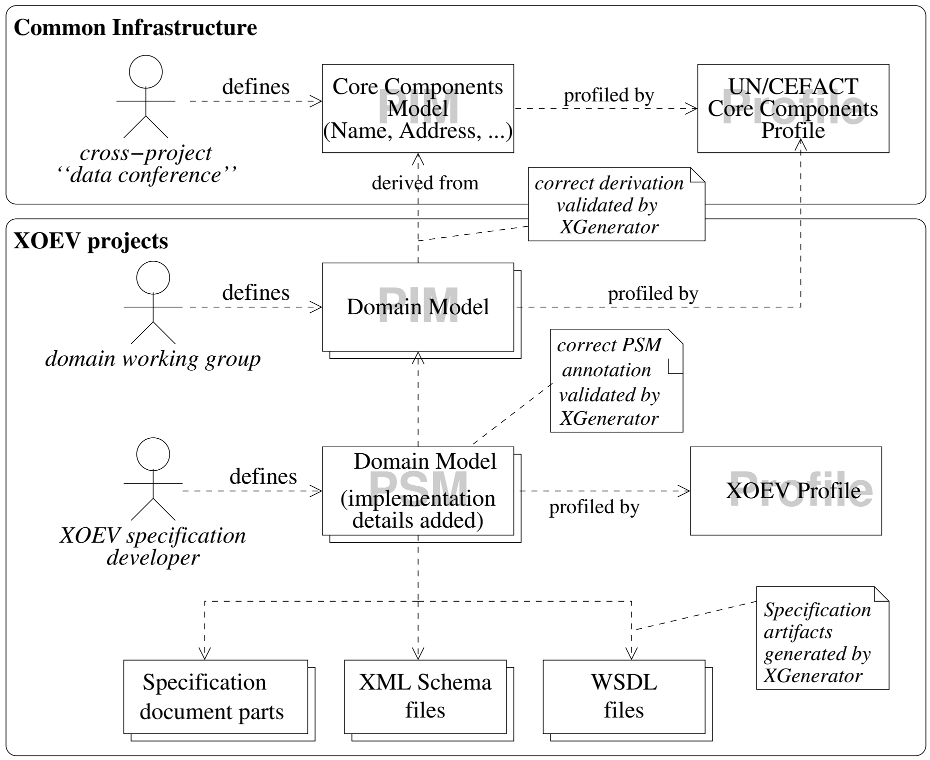

Fig. 2. Production Chain for XOEV Specifications

a well-formed PSM, the XGenerator can automatically generate DocBook (for PDF), XML Schema, and WSDL. The transformation templates used in this step again employ OCL as a model query language, as described in Sect. 4. At the time being the same set of transformation templates is applied in all projects with only minor variations (some templates are intentionally designed as variation points).

Although the process explained above is top down from the technical point of view, it is incremental and iterative in the workflow sense. The core components model has to evolve as more requirements for the standardisation of common information concepts arise in (a growing number of) XOEV projects. Therefore, all projects participate in the data conference and name candidates for core components. When the different perspectives to a common concept have been harmonized in the data conference, new core components are added to the core components repository. Figure 3 depicts the iterative character of this process.

3.1 Short Example

Figure 4 shows an instantiation of the models occurring in the architecture of Fig. 2. The domain model in the lower part is taken from an imaginary project XTenant which

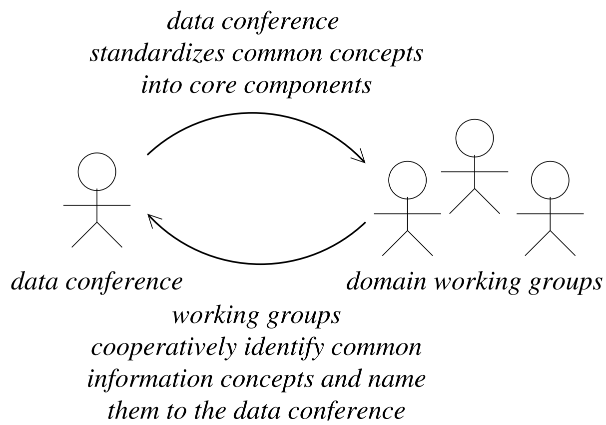

Fig. 3. How Core Components Come Into Existence

is used as a training example in DOL. It illustrates the usage of both UML profiles, the CC profile and the XOEV profile. All German names have been translated to English for easier understanding. In the upper model we see the core components *NameOfNaturalPerson* and *CommonName*, as provided by the data conference. They are stereotyped with «ACC» (aggregate core component). Their attributes are stereotyped with «BCC» (basic core component) and their associations are stereotyped with «ASCC» (associated core component).

In the lower model (domain model of XTenant) we see two domain classes derived from core components using «basedOn». In this model, *NameOfNaturalPerson* and *CommonName* (marked as «ABIE», aggregate business entity) only contain those subsets of their base class properties that are required in the imaginary administrative processes of XTenant. For example, the maiden name is not relevant here. The properties reused in the domain model are marked as «BBIE» (basis business entity) resp. «ASBIE» (associated business entity). An ABIE class must only contain properties that have already been defined (with the same name) in the corresponding core component. Tightening multiplicities of properties, as for *FamilyName* ($0..1 \rightarrow 1$), is allowed when deriving from a core component, as well as replacing a property type with a more specific property type. All these rules are expressed formally as OCL constraints in [UN03].

With the profiling described so far, Fig. 4 shows the conceptual domain model of XTenant, annotated by stereotypes from the CC profile. The remaining stereotypes are part of the XOEV profile. In this figure, they describe how to represent instances of this model in XML using XML Schema. The stereotype «xsdSchema» on the domain model defines that this package will be translated into an XML Schema file with the given name and namespace. All classes with «xsdNamedType» will be translated into named types in XML Schema (as opposed to an anonymous type). The «xsdElement» annotation states that the properties should be expressed as *xs:element* in the schema. The *position* number for each property defines the position in the XML Schema ele-

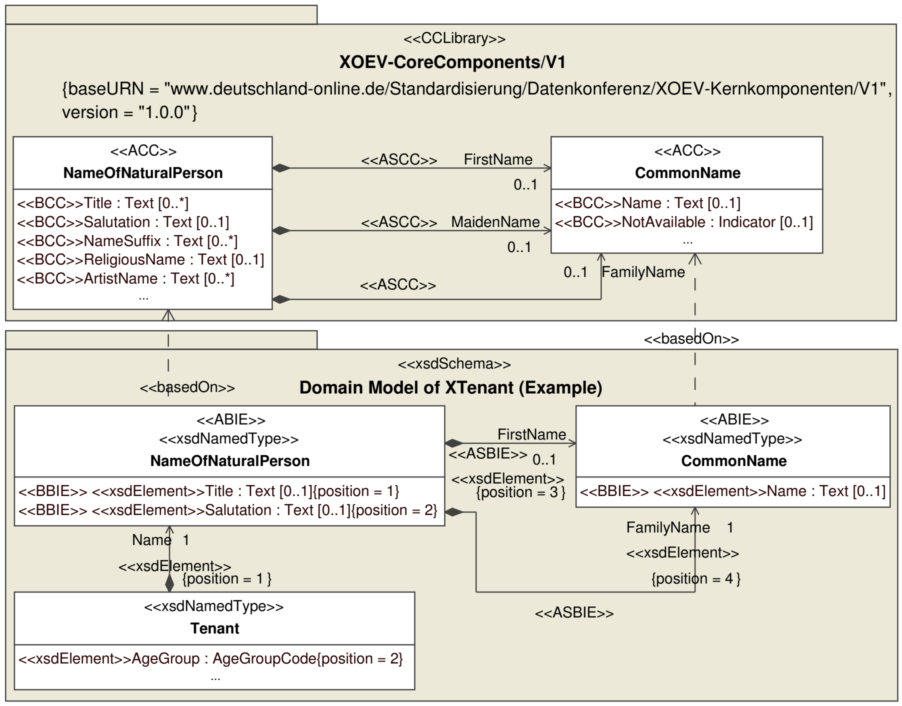

Fig. 4. Example: Platform-specific Domain Model of XTenant with relevant Core Components

ment model. This number is important because we are not aware of any UML tool that allows a consistent ordering of UML attributes and association ends for a class. Among several other well-formedness rules, the XOEV profile ensures that all properties of a class have distinct position numbers. Having such a platform-specific domain model, we can automatically generate the different parts of an XOEV specification using the XGenerator, which is explained in more detail in the following section.

4 The XGenerator Tool

The XGenerator is a central MDA ingredient in DOL. It supports DOL's model-driven development process in two major activities, *model validation* and *model transformation*. The tool is utilised in the domain working groups (to check model validity) as well as in the final generation of the specification products (XML Schema, Documentation, WSDL files).

Figure 5 shows a data flow perspective of the XGenerator. Basically, the tool reads a UML model and produces several output documents; in case of any problems it produces an error report instead. A more detailed description follows:

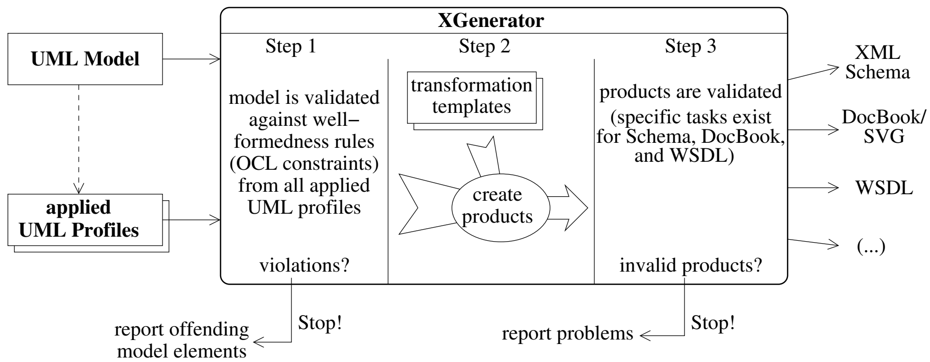

Fig. 5. The XGenerator: Validation and Transformation

Initially, the UML model is read from a corresponding XMI (XML Metadata Interchange) file. This file is exported from a CASE tool. The EMF UML2 XMI importer used here supports a wide range of UML tools. All UML profiles that are applied to this model are read as well (i.e., the CC and XOEV profiles). As the first step the tool then validates the model against its profiles. This means all well-formedness rules from the profiles are checked by evaluating their OCL invariants. If any well-formedness rule is violated, a detailed list of the violated rules and the offending UML model elements is reported to the user and the tool stops.

If the model has been successfully validated, the XGenerator generates its outputs. The tool supports the generation of arbitrary textual output formats. A future version might also support other transformation paradigms, such as model-to-model transformations. The generation process is configured by means of *transformation templates*. The current projects employ transformation templates for XML Schema, DocBook (including SVG for graphics), and WSDL. Transformation templates can be developed and maintained without modification of the tool. Well-formedness rule specification and the transformation language are described in detail in subsections 4.1 and 4.2 below.

For the aforementioned output formats, the XGenerator further supports a final format-specific validation of the results. This is a built-in addition to the tool. Ideally, a valid UML model never produces invalid output documents. However, to detect bugs in the development of transformation templates, Step 3 in Fig. 5 has proven to be a quality assurance measure to catch errors that result from unexpected model constellations. Often, such errors led to further OCL well-formedness rules in the XOEV profile to prevent such constellations in future developments.

4.1 Well-formedness rule validation

UML profiles define well-formedness rules to control their correct application (e.g., correct usage of stereotypes). The XGenerator supports the automatic validation of well-formedness rules when they are specified as OCL invariants. To validate a UML profile, these OCL invariants have to be specified in the context of a UML meta-class or a stereotype.

OCL invariants can be provided as an external XML file to the XGenerator. A (constructed) example well-formedness rule looks as follows:

```
<invariant context='Class' name='propsUniqueInClass'>
  <body>
    self.ownedAttribute->forAll(p1,p2 |
      p1.extensionMyStereo.prop = p2.extensionMyStereo.prop
         implies (p1=p2) )
  </body>
  <documentation>
    Prop values must be unique in each class!
  </documentation>
</invariant>
```

This constraint could be used to validate the model from Fig. 6 (the rest of this figure is discussed in the next subsection). If we would change the prop value of *someAttr* to 1, the XGenerator would report "The class OneClass violates the well-formedness rules propsUniqueInClass - Prop values must be unique in each class!" Notice that the context of *propsUniqueInClass* is *Class*. We could have alternatively used the meta-class *Property* as the context (and rewrite the OCL body) to gain more fine-grained errors. We made such context changes for well-formedness rules several times in both profiles. The changes to the CC profile have been reported back to the UN/CEFACT and are likely to be incorporated into the next version the CC profile.

The XGenerator internally uses the UML Specification Environment (USE) [GBR07] as its OCL evaluation and validation engine.

4.2 Transformation templates

The XGenerator currently provides model-to-text transformations (a comparison to other approaches follows in the next subsection). The transformation templates are operational descriptions that, given a UML model as input, produce one or more text files as output. Similar to template processors such as JSP or Apache Velocity (which is actually used internally by the XGenerator), the transformation templates consist of fragments of the result document which are surrounded by template language directives. Compared with conventional programming languages, the output data is contained directly whereas transformation and template logic is escaped (indicated by '#' or '$' in Fig. 6).

The XGenerator uses only a few processing directives: conditional output, iteration (foreach), sub-template evaluation, and evaluation of OCL expressions.

The templates use OCL as an expression language to query and navigate through the UML model. Applied profile stereotypes and their properties are available as specified by the OMG.

Figure 6 provides a small example to illustrate the XGenerator transformation templates. The example template generates a simple model summary (classes and attributes). On the left-hand side there is a UML model (which consists of only two classes). The model uses a single stereotype «MyStereo» from a UML profile (constructed for this example, not shown in this figure). «MyStereo» introduces a single stereotype property *prop* that can be set wherever this stereotype is applied. (In UML 2, stereotype properties replace UML 1.x tagged values that could be applied independently of stereotypes.) On top of the figure, we see a transformation template which is parametrised with a single package-typed parameter *p*. Let us assume that this template is invoked with our complete UML model (such a root template has to be specified in the XGenerator configuration).

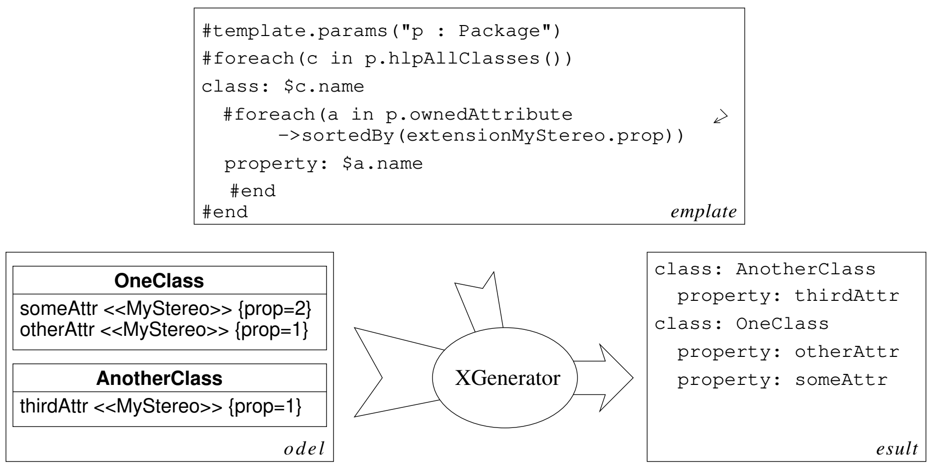

Fig. 6. Transformation Template Example

The template evaluation will produce the output shown on the right-hand side. The evaluation consists of two nested foreach-loops, the outer iterates over all classes, the inner iterates over the attributes of each class. The outer loop uses an additional operation *hlpAllClasses()* in the meta-class Package in its range expression. The XGenerator supports the definition of such additional OCL operations for all meta-classes. Let us assume that *hlpAllClasses()* returns all classes contained in this package in alphabetical ordering. Additional operations allow us to factor out common methods of transformation templates. In contrast, the inner loop uses an in-line OCL expression to sort a set of attributes by their *prop* values.

Notice that three different languages are used in the template depicted in Fig. 6: (1) the template control language consisting of operational constructs (they all start with a hash sign), (2) embedded OCL as a side-effect free expression language, and (3) fragments in the actual output language (e.g., XML Schema or DocBook).

At the time of writing, the XOEV transformation templates consist of approximately 1200 lines of code. These include only simple OCL expressions. Complex logic is factored out into additional operations of approx 230 lines of OCL.

4.3 Comparison to other MDA tools

During the last years, several tools and approaches have been developed that support the model-driven paradigm. Model transformation is one important area in this paradigm, model validation is another. As one of many classifications, one might roughly categorise model transformation approaches into model-to-model and model-to-product (often referred to as model-to-code or model-to-text). In model-to-model transformations, the transformation result is a model (which belongs to a meta-model, either the same as the that of source model or a different one.) In model-to-product transformations, the result is typically not a model in the MOF sense. Instead, various output targets are generated. Typical examples are program code and configuration files.

The OMG's Query, Views, Transformations [OMG06] standard and tools such as ATL [ABJK06], YATL [Pat04], and MOFLON [AKRS06] fall into the first category. Together with the Mof2Text proposal [OMG04] and tools like MofScript [OMG05] and AndroMDA [AMD08], the XGenerator falls into the second category. A general overview of commercial and open source MDA systems (also including UMT, MTL, GMT or BOTL) can be found in [Wan05].

The XGenerator transformation templates most resemble MofScript. Conceptually, their language constructs could be compared with QVT's ImperativeOCL (which also is an operational transformation language, although it is used for model-to-model transformations). XGenerator's targets are less code centric than those of (e.g.) Eclipse JET and it puts a strong focus on validation and profile conformance checking.

5 Evaluation

The model-driven architecture of DOL that we illustrated in this paper consists of several elements: (1) one core components model, (2) several domain models, (3) the CC profile for the correct usage of core components, (4) the XOEV profile for platform-specific tailoring of domain models, (5) a tool for automatic profile validation (the XGenerator), and (6) a tool for the automatic generation of XOEV specifications from platform-specific domain models (again, the XGenerator).

This architecture provides us three major benefits in DOL: (A) The separation of conceptual modeling from technical details (e.g., XML Schema representation) allows a tighter integration of non-technical domain experts into the modeling process. (B) The strong, formally checked reuse of cross-project core components in domain models has greatly improved the conceptual interoperability between XOEV projects. This is enabled through automated validation of OCL well-formedness rules. At the time being, several core components have been identified, unifying important concepts such as *Address*, or *Name*. The inconsistency of these concepts has turned out to be a real barrier for the implementation of cross-domain processes in the past. (C) The integrated generation of the various parts of a XOEV specification (documentation, XML Schema, WSDL) has proven to be a time-saving and, in particular, consistent way to deal with the inherent redundancy between these parts. Further, all artifacts are uniform and consistent in themselves and w.r.t each other. Compared with manual creation of PDF documents, XML Schema, and WSDL files, this provides a considerable reduction of errors and a delivery of specifications having all parts up-to-date.

Central to our approach is the use of OCL: (1) as the language for modeling the profiles (in the step for validating the stereotype application) and (2) as the language for accessing the input model (in the step for transforming the input model into various output models). Because OCL is a high-level modeling language and not a programming language, concepts for both steps can be formulated (and can be later improved easily) on a conceptual and not on a programming level. OCL well-formedness rules can guarantee the correctness of the input model independent from any performed transformation, and OCL navigation expressions allow easy and powerful access to the input model. It is symptomatically that, while there have been almost no changes to the XGenerator tool itself in the past, several improvements to the UML profiles and transformation templates have been made (and will be made in the future) through the XOEV projects.

What further improvements could be and have to be done? The large bundle of technologies still intimidates new projects. This is especially true because the support for profiled modeling is still rather uncomfortable in all UML tools we know. Therefore, a couple of plug-ins for the UML tool MagicDraw are currently being finalized that provide support for easier management of common profile related modeling tasks, e.g., «xsdElement»/position handling. Convincing more people, particularly from the domain and administrative levels, that formal, model-driven techniques can bring substantial improvements for system development will continue to be a demanding but also important task in the future.

References

[ABJK06] F. Allilaire, J. Bézivin, F. Jouault, and I. Kurtev. Atl - eclipse support for model transformation. In *Proceedings of the Eclipse Technology eXchange workshop (eTX) at the ECOOP 2006 Conference, Nantes, France*, 2006.

[AKRS06] Carsten Amelunxen, Alexander Königs, Tobias Rötschke, and Andy Schürr. MOFLON: A standard-compliant metamodeling framework with graph transformations. In Arend Rensink and Jos Warmer, editors, *ECMDA-FA*, volume 4066 of *Lecture Notes in Computer Science*, pages 361–375. Springer, 2006.

[AMD08] The AndroMDA tool. website, 2008. http://galaxy.andromda.de.

[DOL08] Deutschland Online. website, 2008. http://www.deutschland-online.de.

[GBR07] Martin Gogolla, Fabian Büttner, and Mark Richters. USE: A UML-Based Specification Environment for Validating UML and OCL. *Science of Computer Programming*, 2007.

[OMG04] MOF model to text transformation language RFP. OMG document ad/04-04-07, www.omg.org, 2004.

[OMG05] MOFScript – Second Revised submission for MOF Model to Text Transformation Language RFP. OMG document ad/05-11-03, www.omg.org, 2005.

[OMG06] MOF QVT final adopted specification. OMG document ad/05-11-01, www.omg.org, 2006.

[OSC02] OSCI Transport 1.2 final specification. Technical report, OSCI Leitstelle, 2002. http://www1.osci.de/sixcms/detail.php?gsid=bremen02.c.1205.de.

[Pat04] Octavian Patrascoiu. YATL:Yet Another Transformation Language. In *Proceedings of the 1st European MDA Workshop, MDA-IA*, pages 83–90. University of Twente, the Nederlands, January 2004.

[SF08] The XGenerator project in the SourceForge OpenSource project repository. website, 2008. http://www.sourceforge.net/projects/xgenerator2.

[SYN08] Website of the Frech eGovernment initiative “Synergies - Les ressources de l’administration électroniquie”, in French, 2008. http://www.synergies-publiques.fr/.

[UN03] UN/CEFACT - Core Component Technical Specification Version 2.01. Technical report, United Nations Centre for Trade Facilation and Electronic Business, 2003. http://www.untmg.org/dmdocuments/CCTS_v201_2003_11_15.pdf. Also published as ISO standard 15000-5 (Electronic Business Extensible Markup Language (ebXML), Part 5).

[UN06] UN/CEFACT UML Profile for Core Components Version 1.0 (BCSS). Technical report, United Nations Centre for Trade Facilation and Electronic Business – Techniques and Methodologies Group, Oct 2006. Available online at http://www.untmg.org.

[Wan05] W. Wang. *Evaluation of UML Model Transformation Tools*. Technical University of Vienna, Business Informatics Group, Master Thesis, 2005.

[XME08] Website of the municipal citizens registration project XMELD, in German, 2008. http://www1.osci.de/sixcms/detail.php?gsid=bremen02.c.1168.de.