The Secret Life of OCL Constraints

Oliver Hofrichter
University of Bremen

Lars Hamann
University of Bremen

Martin Gogolla
University of Bremen

Frank Steimke
KoSIT, Die Senatorin für Finanzen, Bremen

ABSTRACT

This paper reports on the use of OCL constraints in a German e-government project and focuses on the identification of diverse manifestations of invariants. Beyond invariants' formal content three other manifestations are identified: (a) feedback by a tool based on the processed invariants, (b) the invariant's textual explanation as a basis for a modeler who uses the invariants and (c) implicit assumptions for a model transformation resulting from the invariants.

Categories and Subject Descriptors

D.3.3 [**Programming Languages**]: Language Constructs and Features—*constraints*

General Terms

Design, Standardization, Languages

Keywords

Manifestation of invariants, E-government, Usability

1. INTRODUCTION

Model-driven engineering (MDE) plays a central role in the German e-government project "XÖV" [1], which stands for "XML in the public administration" (German: XML in der öffentlichen Verwaltung). This project is concerned with the standardization of the electronic inter-authority data exchange in the German public administration. In this context, standardization is of high importance because the public authority's IT systems are implemented by a great number of different vendors. At the moment, electronic data interchange of 21 areas of the German public administration are specified this way. A message interchange specification is represented by XML schema files and a DocBook documentation. These components are automatically generated from an abstract model formulated in the Unified Modeling Language (UML) by an MDE tool called XGenerator.

OCL'12 September 30 2012, Innsbruck, Austria

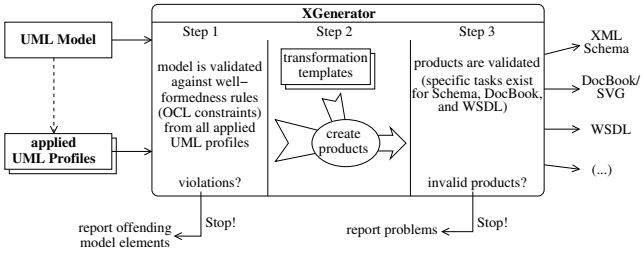

Figure 1: The validation and transformation process

The model-driven approach makes sure that specifications are designed fast, iteratively, efficiently and in a way that guarantees the consistency between the UML model and the generated results at any moment. The XGenerator tool allows for model validation and model transformation. It is based on Eclipse and the UML-based Specification Environment (USE) and was developed in a cooperation between the coordination office and partners from academia and industry.

The validation and transformation process for standard development is summarised in Figure 1. OCL is successfully deployed in different places: First, OCL well-formedness rules are processed by the MDE tool to ensure that the UML profiles, applied in the project, are used in a correct way and that valid results arise from the generation process. Further, OCL is used as a query language inside templates formulated in a template language and processed by the MDE tool to perform model-to-text-transformations. The XGenerator processes a model of the data exchange, developed by domain experts from authorities with support of modelers. In order to be able to align the model to the domain and to add information that is necessary for the transformation into several artifacts, UML profiles are applied to the model. One of these profiles is the so called "XÖV" profile. It includes 28 stereotypes. Altogether the profile is supplemented with 119 well-formedness rules. Among these there are 88 mandatory, 9 are optional and 22 are "recommendation". Furthermore, the following categories of invariants based on best practices are distinguished: (a) rules that regard the conformance of the model's structure, (b) rules regarding the XML schema design's conformity and (c) rules regarding the correct application of documentation stereotypes. Due to its size (119 rules are applied) and its practice-orientation this project is well-suited for the investigation of the multifacetedness of OCL invariants.

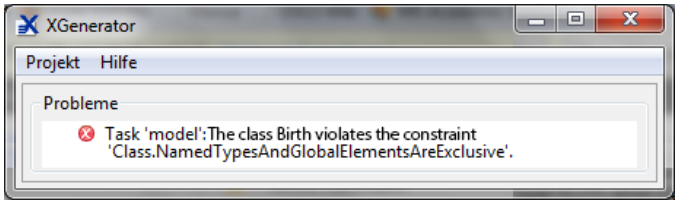

Figure 2: Invariant's manifestation for the user of the MDE tool XGenerator

2. INVARIANTS' MANIFESTATIONS

Three other manifestations of invariants that exceed their formal content were identified in the context of the regarded project. These are described in the following.

2.1 Feedback about invariants' violation

A range of XML schema files is the data exchange specifications' main component in the project context. These schema files restrict the structure of the XML based messages that are going to be exchanged between different public authorities. In order to create a schema for a message, a modeler has to develop a UML model of the message. In doing so it can happen that the modeler applies the profiles in an incorrect way. However, for the generation of valid schema files the correct application of the profiles is unavoidable. For this purpose the modeler has to gain an information about the violation. Furthermore, this information has to be precise with the result that the particular element that is responsible for the violation can be easily detected by the modeler. Such a feedback mechanism is offered by the XGenerator. An example is depicted in Figure 2. After the XGenerator has validated a model against OCL well-formedness rules, the modeler obtains an information about the violating element and the violated constraint. In the example given in Figure 2 the class `Birth` violates the constraint `NamedTypesAndGlobalElements`. By means of the feedback it is not necessary for the modeler to dive into all details of the invariant's formal specification in order to develop a model with correct application of the profiles.

2.2 Implicit assumptions for model transformation

Another manifestation of invariants has emerged in the context. As depicted in Figure 1, a model is validated and transformed to text subsequently based on templates. With this in mind invariants can influence the model transformation by facilitating the template development. Assertions associated with elements by invariants do not have to be handled in a special way in the templates anymore because the template developer can take them for granted. This way template development is simplified and accelerated and the resulting templates are easier to understand. The invariant `SingleInheritance` and its impact on the templates can serve as an application example. It ensures that certain classes that possess particular qualifications in the project context have at most one superclass. As a result the template developer is not forced to use additional boolean expressions to select the requested element while accessing all parents of a `Classifier` in the templates.

It is very important that assumptions like this are documented in a more abstract form. Otherwise the template developer would have to dive into the invariants' technical details or even worse these assumptions would stay in hiding.

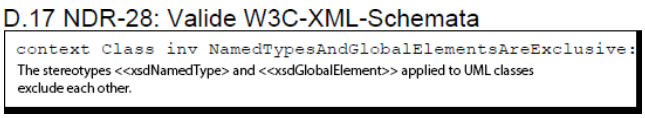

Figure 3: Invariant's verbal manifestation

2.3 Manifestation in verbal form

The third manifestation identified in the project context is an invariant's representation in verbal form. While the invariant's formal manifestation is a proper representation for processing by the XGenerator, modelers using the invariants are in need of a more abstract description of the invariants. Modelers, who specify the messages interchanged between register offices, are not necessarily equipped with knowledge like first-order predicate logic and the available language elements of OCL. However, they have to take into consideration the invariants while modeling. In order to prevent that they have to become acquainted with all the technical details of invariants' formal content a document (called "XÖV" manual) with verbal descriptions of the invariants was developed. An excerpt from the manual explaining the invariant `NamedTypesAndGlobalElementsAreExclusive` is depicted in Figure 3. In this way the modeler can concentrate on the model's contents and has the possibility to have recourse to the manual in case of doubt. Due to its enormous importance for the modelers the manual has to be up to date at any time. Since the verbal manifestation as well as the feedback messages are based on a common source the consistency between these manifestations is warranted.

3. CONCLUSION

In the present work we have investigated the various manifestations of invariants used in an e-government project. Three further manifestations were identified beyond invariants' common formal content. These manifestations may not seem surprising. However, invariants' developers as well as modelers, who use these invariants, have to be fully aware of the existence of these manifestations. Especially, because bringing the manifestations to mind can facilitate tasks like model and template development. Additional Related Work can be found in [2].

4. REFERENCES

[1] F. Büttner, M. Kuhlmann, M. Gogolla, J. Dietrich, F. Steimke, A. Pankratz, A. Stosiek, and A. Salomon. MDA Employed in a Joint eGovernment Strategy: An Experience Report. In T. Bailey, editor, *Proc. 3rd ECMDA Workshop "From Code Centric To Model Centric Software Engineering" (2008)*, `http://www.esi.es/modelplex/c2m/2008/docum/2-buettner_mda_employed_in_egovernment.pdf`, 2008. European Software Institute.

[2] O. Hofrichter, L. Hamann, M. Gogolla, and F. Steimke. The Secret Life of OCL Constraints. `http://www.db.informatik.uni-bremen.de/publications/intern/SecretLife.pdf`, 2012.