Electronic Communications of the EASST
Volume 36 (2010)

Proceedings of the Workshop on OCL and Textual Modelling (OCL 2010)

Navigating Across Non-Navigable Ecore References via OCL

Martin Hanysz, Tobias Hoppe, Axel Uhl, Andreas Seibel, Holger Giese, Philipp Berger and Stephan Hildebrandt

6 pages

Guest Editors: Jordi Cabot, Tony Clark, Manuel Clavel, Martin Gogolla
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Navigating Across Non-Navigable Ecore References via OCL

Martin Hanysz[1], Tobias Hoppe[1], Axel Uhl[2], Andreas Seibel[1], Holger Giese[1], Philipp Berger[1] and Stephan Hildebrandt[1]

[1] holger.giese@hpi.uni-potsdam.de, http://www.hpi.uni-potsdam.de/giese/
Hasso Plattner Institute at the University of Potsdam
Prof.-Dr.-Helmert-Str. 2-3, 14482 Potsdam, Germany

[2] axel.uhl@sap.com
SAP AG, Office of the CTO
Dietmar-Hopp-Allee 16, 69190 Walldorf, Germany

Abstract: The Eclipse Modeling Framework (EMF) and its meta-meta model Ecore support uni-directional and bi-directional references. It is quite common that references are defined uni-directionally only because of saving storage space or separating meta models, which is problematic when implementing Object Constraint Language (OCL) constraints that require navigation against the direction of uni-directional references. This is essential for certain approaches, e.g., incremental evaluation of OCL constraints on models shown by Altenhofen et al. that is used in SAP's Modeling Infrastructure (MOIN). In this paper, we present an approach that overcomes the aforementioned issue by providing navigation across non-navigable Ecore references via OCL. We further discuss different alternative solutions and briefly describe the realization that was outcome of a project in cooperation with the SAP AG.

Keywords: OCL, Ecore, EMF, Navigation

1 Introduction

The Object Constraint Language (OCL) [Obj06b] was originally designed as a formal language to describe invariants and queries on Unified Modeling Language (UML) [Obj04] models. Nowadays, the application of OCL is more extensive, e.g., it is applied in Meta Object Facility (MOF) [Obj06a] compliant tools or the Eclipse Modeling Framework (EMF) [SBPM08] to constrain EMF models that are conform to the meta-meta model Ecore. However, in contrast to Complete MOF (CMOF) [Obj06a, pp. 45], Ecore, which is conceptually equal to Essential MOF (EMOF)[Obj06a, pp. 31], uses the restricted reference concept of EMOF. In CMOF references (associations) always have association ends, which implies that there is always the possibility to address the opposite of a reference, even if the reference is defined as uni-directional.

In Ecore references are always uni-directional. A bi-directional reference is realized by explicitly modeling two references as being the opposite of each other. Additionally, Ecore does not support the concept of association ends. Thus, if defining a reference as uni-directional, the opposite of the Ecore reference cannot be addressed. Defining bi-directional references between elements of different meta models is not supported by EMF, because it would result in cyclic

dependencies between these meta models. This is problematic because legacy EMF meta models do not always provide bi-directional references (cf. UML2 tools[1]) to save storage space and meta models may be separated in several meta models (separation of concerns). In both cases, OCL constraints cannot be expressed against navigation direction. This is essential for certain approaches, e.g., incremental evaluation of OCL constraints on models shown by Altenhofen et al. in [AHK07] that is implemented in SAP's Modeling Infrastructure (MOIN). Furthermore, OCL constraints can be optimized by restructuring navigation, which is restricted when references are uni-directional. Thus, navigating across non-navigable references via OCL is required to overcome this issue in such cases.

In this paper, we present an approach that overcomes this issue with minimal impact to existing applications. It supports navigation across non-navigable references via OCL *within* and *between* EMF meta models without adding cyclic dependencies. The approach was implemented as a part of a project (see [Ber10, Hop10, Han10, Sch10]) in cooperation with the SAP AG[2]. Therefore, we first introduce our requirements and design decisions in Section 2. In Section 3 we give a brief explanation of our approach and finally conclude this paper in Section 4.

2 Design Considerations

To support the acceptance of our approach by the EMF community, the following requirements have to be satisfied:

1. Avoid invalidation of existing technologies
2. Avoid impact on current usage of technologies
3. Avoid high integration efforts

There are several alternatives to provide navigation across non-navigable references in Ecore. A straightforward approach is to only use bi-directional references in EMF models instead of uni-directional ones. Consequently, partitioning of EMF models would be restricted due to cyclic dependencies and the partition in two meta models as shown in Figure 1 would not be possible. This clearly violates requirement two, because it restricts how EMF models can be partitioned.

Another approach that avoids the dependency cycles a bi-directional reference creates, is to allow the opposite reference to be contained by the original reference. This circumvents the limitations of the aforementioned approach, but yields severals other problems. For example the validation of multiplicity or ordering constraints for the opposite reference becomes a very expensive operation, due to the fact that the list of referenced instances is not stored explicitly. Apart from that, the original architecture assumes that a reference is always contained by an element. Since this assumption does not hold anymore, existing technologies relying on it might be invalidated.

To circumvent those requirement violations, another alternative that avoids to change Ecore has been taken into account. It relies on annotations, which are added to uni-directional references in EMF models, as shown in Figure 1. The example shows a simple situation with two

[1] http://www.eclipse.org/uml2/
[2] http://www.sap.com

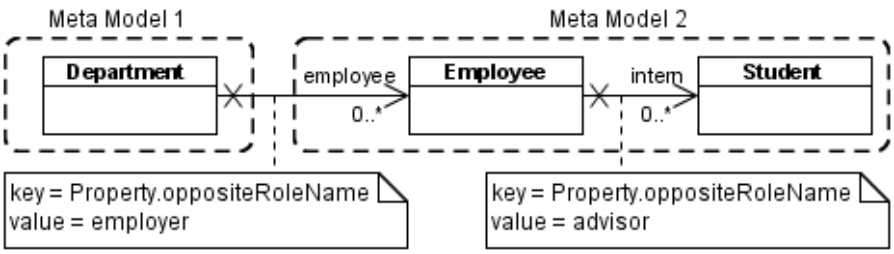

Figure 1: An example that visualizes the problem of uni-directional references

meta models defining two concerns of a company. The annotations contain opposite role names of the references. This solution does not impact the usage of existing technologies, because it can be seamlessly integrated into existing EMF models. It also does not impact requirement three since these annotations can be automatically generated. To navigate across non-navigable references using these annotations, we extend the OCL meta model as well as the OCL tooling of EMF. Our extension of OCL does not invalidate requirement two because it is downward compatible.

3 Contribution

In this section, we explain how we exploit the annotation concept of Ecore and the extensions we have made to the OCL tooling in EMF.

3.1 Adding Annotations to EMF Models

Annotations in Ecore are defined by the EAnnotation element, which can be added to any element. EAnnotation may have a details entry, that maps key to value attributes. In our approach, every uni-directional reference that should be navigated in its opposite direction, represented by an EReference element, is associated with an EAnnotation element holding a details entry that has a key attribute "Property.oppositeRoleName". The value attribute contains the name of the opposite role. EMF models that should benefit from bi-directional navigability can be modified manually or automatically.

3.2 Extending OCL

Our approach does not come without extending EMF's OCL, but we are using the *dot* notation for navigation across references to be compatible to legacy OCL constraints. Thus, there is no need to introduce an extra notation to make the missing navigability explicit. For example, the expression in Figure 2 refers to the annotated opposite role name of the EReference *intern*. This constraint expresses, that each Student must have at least one Employee as his advisor.

The extensions we have made are in the background and extend OCL's abstract syntax (meta model), as well as its parser, validator, and evaluator.

```
context Student inv:
        self.advisor.size() > 0
```

Figure 2: An example OCL expression using reverse navigation

3.2.1 Abstract Syntax of OCL

Although we did not change the concrete syntax of OCL, we have extended the abstract syntax of OCL to reflect the navigation across non-navigable references. Therefore, we extended the NavigationCallExp with an OppositePropertyCallExp as shown in Figure 3. An OppositePropertyCallExp describes the navigation of a reference between two elements that are not explicitly connected via an EReference pointing from the source element to the target elements, which avoids cyclic dependencies if elements are defined in different meta models. Instead, this association is built by an EReference from the target that defines the opposite navigation direction.

The OppositePropertyCallExp inherits a navigation source property, which is the element from where the navigation is started. It is also related to a referred opposite property, which contains the opposite role name that is defined in the EAnnotation of an EReference pointing to the navigation source element.

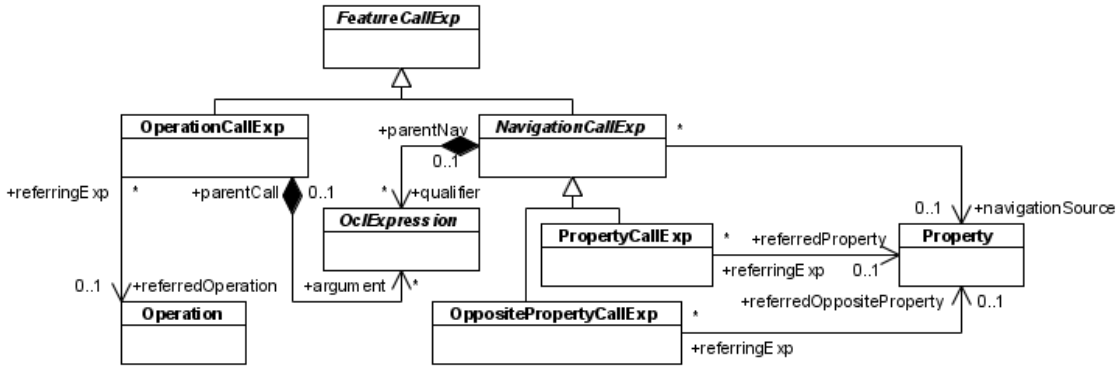

Figure 3: Extended OCL meta model

3.2.2 OCL Parser

OCL constraints are handled by means of their abstract syntax. Thus, OCL constraints defined in the concrete syntax have to be parsed into the abstract syntax first. The parsing of OCL constraints in EMF is performed by an OCL analyzer. We extended the parsing functionality to create OppositePropertyCallExp instances when necessary.

When the OCL analyzer comes across the *dot* notation, it tries to resolve the token next to it to a type name, a variable name, a property name, an opposite property name, an association class name, and finally to an undefined name. The resolving of an opposite property name is

encapsulated in the OppositeEndFinder to allow different implementations of it[3]. The example in Figure 2 contains an opposite property after the first *dot*, since *advisor* is the annotated opposite role name of the EReference *intern*. Hence, an OppositePropertyCallExp is created.

3.2.3 OCL Validator

After parsing the OCL expression into an abstract syntax, it needs to be validated. Throughout the whole OCL tooling of EMF, several visitors are used. To make these visitors aware of opposite properties, the common Visitor interface has been extended with a visitOppositePropertyCallExp() method and the AbstractVisitor has been adapted to properly call this method when coming across an OppositePropertyCallExp. The Visitor used to validate OCL expressions is the ValidationVisitor and has been modified to implement this method to check if the correctness of the OppositePropertyCallExp is given.

3.2.4 OCL Evaluator

The OCL evaluator actually executes the evaluation of the parsed and validated OCL expression. For this task, the EvaluationVisitor is used. It has also been modified to implement the visitOppositePropertyCallExp() method and use a modified, opposite property aware version of the EvaluationEnvironment to perform the navigation.

Navigating an OppositePropertyCallExp results in the set of instances, that reference one of the expression's source instances via an EReference that is annotated with the given opposite role name. Therefore, the evaluation of the expression in Figure 2 leads from *self* to all instances of Employee that reference that one instance of Student. Our approach uses EMF's reflection mechanism for resolving the requires instances to avoid cyclic dependencies between meta models. To find all instances pointing to an EObject via an EReference, EMF provides a utility called ECrossReferenceAdapter. The cross-referencer iterates through the content tree rooted at the initially passed EObject, Resource, or ResourceSet and checks for each retrieved reference of all objects if it points to the wanted object or not. In case of a positive match, the object is added to the result [SBPM08, pp. 523].

Because the cross-referencer is a simple implementation, we may get scalability issues when dealing with large models. Thus, we have implemented a modular interface to integrate alternative technologies for resolving OppositePropertyCallExps, e.g, Model Query 2 (MQ2)[4]. MQ2 provides an efficient interpreter for resolving queries, which comes with an index structure. In this alternative, MQ2 queries are formulated and interpreted to return all objects referencing the source via an EReference with a specific EAnnotation.

4 Conclusions & Future Work

Throughout this paper, we presented a solution that overcomes a restriction of Ecore. This restriction prohibits navigability through EMF models via OCL constraints, because uni-directional

[3] currently Model Query 2 is used to find the properly annotated references
[4] http://www.eclipse.org/modeling/emf/?project=query2

references cannot be navigated in reverse. This is problematic, when working with EMF models that provide many uni-directional references (cf. UML2 tools). Our approach can be seamlessly integrated into existing technologies. It uses the EAnnotation concepts of Ecore. These annotations are used by an extension of EMF's OCL that we have implemented.

As future work we plan to evaluate the scalability of our approach by using EMF's cross-referencer, MQ2, and other indexing technologies like EMF Index[5]. Furthermore, multiplicities cannot be expressed via the annotations currently.

Bibliography

[AHK07] M. Altenhofen, T. Hettel, S. Kusterer. OCL Support in an Industrial Environment. In *Proc. of the Workshops and Symposia at MoDELS 2006, Genoa, Italy*. LNCS, pp. 169–178. Springer, 2007.

[Ber10] P. Berger. Central Event Manager for the Eclipse Modeling Framework. 2010. http://www.hpi.uni-potsdam.de/giese/gforge/bibadmin/uploads/pdf/Ber10_Philipp Berger_Bachelorarbeit.pdf.

[Han10] M. Hanysz. Instance-Based Context Calculation of OCL Expressions. 2010. http://www.hpi.uni-potsdam.de/giese/gforge/bibadmin/uploads/pdf/Han10_Martin Hanysz_Bachelorarbeit.pdf.

[Hop10] T. Hoppe. Synthesis of Event Filter determining the reevaluation of affected OCL expressions. 2010. http://www.hpi.uni-potsdam.de/giese/gforge/bibadmin/uploads/pdf/Hop10_TobiasHoppe_Bachelorarbeit .pdf.

[Obj04] Object Management Group. UML 2.0 Superstructure Specification. October 2004. Document: ptc/04-10-02 (convenience document).

[Obj06a] Object Management Group. Meta Object Facility. 2.0 edition, January 2006.

[Obj06b] Object Management Group. Object Constraint Language. 2.0 edition, May 2006.

[SBPM08] D. Steinberg, F. Budinsky, M. Paternostro, E. Merks. *Eclipse Modeling Framework 2.0*. Addison Wesley, 2 edition, December 2008.

[Sch10] T. Schröter. Effiziente Navigation bei der OCL-Auswertung. 2010. http://www.hpi.uni-potsdam.de/giese/gforge/bibadmin/uploads/pdf/Schroeter10_Thea Schroeter_Bachelorarbeit.pdf.

[5] www.eclipse.org/proposals/emf-index/