

Directions for further development of OCL 2.0

OMG Document number: ad/2003-05-01

By Anneke Kleppe and Jos Warmer

In the OMG technical meeting of March 2003, the ADTF has recommended for adoption the UML 2.0 Infrastructure proposal, OMG document ad03-01-01, and the OCL 2.0 proposal, OMG document ad2003-01-07. These two proposals need to be properly aligned. The expectation is that one or more FTF's will be established during the upcoming OMG technical meeting in June 2003 in Paris. This document contains directions along which the OCL metamodel and the UML Infrastructure metamodel can be integrated to form one single language. The OCL has originally been conceived as part of the UML language, and has never been intended to be used separately. A tight connection between the two metamodels is necessary for all modelers that want to use either UML, OCL, MOF, or a combination of these.

Within the OMG the OCL is heavily used for specifying constraints on metamodels in the various OMG standards. OCL is also used extensively in many of the proposals for the MOF 2.0 QVT RfP, as the solution for a query language, and as part of a transformation definition language. In all these cases, the OCL is used at the metamodel level, i.e. at the MOF level. Formally this is incorrect, because OCL isn't part of the MOF. Within the UML/MOF/OCL 2.0 framework, the UML and the MOF share a common core. The MOF 2.0 proposal reuses the UML 2.0 Infrastructure. By integrating the OCL into the reused part of the UML 2.0 infrastructure, the OCL metamodel is integrated with the MOF 2.0 as well. The definition of OCL at the MOF level is automatically achieved.

Some aspects of the coupling of the OCL metamodel with the UML Superstructure metamodel are addressed in this document as well. These discussions are preliminary, because the Superstructure has not yet been approved by the ADTF. Further work will be necessary.

1.1 INTEGRATION OF OCL AND UML METAMODELS

OCL has always relied on the type system defined in the UML, therefore the OCL 2.0 submission has been written in anticipation of the approval of the UML Infrastructure and Superstructure submissions. In practise this means that the connections between the OCL metamodel and the UML metamodels have been identified early on. These connections have been minimized and clarified in the subsequent versions of the OCL submission. Because of this, it is rather easy to integrate both metamodels into one, thus creating a combined UML/OCL language.

1.1.1 OCL Core and SuperStructure Aspects

During a long period of time it was not clear which concepts would be incorporated in the UML Infrastructure and which would be in the Superstructure. Therefore it was not possible to adapt the OCL submission to support the division between these two. The first issue that needs to be settled is the distinction between the elements of the OCL metamodel that can only be linked to superstructure concepts and the ones that are more fundamental and can be linked to the infrastructure.

We propose to divide the OCL *AbstractSyntax* package into an *OCL-core* and an *OCL-super* package, as has been done in the accompanying Rose model. The *OCL-core* can be part of the *InfrastructureLibrary::Core* package, the *OCL-super* of the *UML* package that is defined in the Superstructure submission. The *OCL-super* package contains the following elements from the OCL *AbstractSyntax* package:

- *OclMessageArg*, and its associations
- *OclMessageExp*, and its associations
- *OclMessageType*, and its associations
- *OclState*, and its associations
- *UnspecifiedValueExp*, and its associations
- the association between *NavigationCallExp* and *OclExpression* with rolename *qualifiers*

The *OCL-core* package contains all other elements from the OCL *AbstractSyntax* package.

1.1.2 Coupling OCL-Core to UML-Basic or UML-Constructs Package

The next issue is whether the *OCL-core* package should be connected to the *Basic* or *Constructs* subpackage of the UML *InfrastructureLibrary::Core* package. In our opinion both options are feasible. However, there are two facts that support the choice for the *Constructs* package. First, the MOF 2.0 Core submission (ad/2003-04-07) uses the *Basic* package to define EMOF, "which is the subset of MOF that closely corresponds to the facilities found in OOPLs and XML". Second, OCL iterators (loop operations) are much more advanced and powerful than the looping facilities found in most OOPLs. Mapping OCL iterators to a loop construct in an OOPL is not a straightforward task. Therefore, we propose to connect the *OCL-core* package to the *Constructs* package, as shown in figure 1-1.

1.2 CHANGES IN OCL ABSTRACT SYNTAX METAMODEL

The OCL metamodel as proposed in ad2003-01-07 needs to be adjusted to fit within the UML Infrastructure. The adjustments are minor and easy to understand.

The model from section 3.2, the metamodel for OCL types, shown in figure 1-2, should be changed as shown in figure 1-3. The metaclasses that are shifted to the *OCL-super* package have been removed, and the metaclasses that used to come from the *UML1.4::Core* package have been replaced by their counterparts from the *Constructs* package.

The model from section 3.3.1, the basic metamodel for expressions, shown in figure 1-4, should be changed according to figure 1-5. In it the *OclExpression* metaclass no longer inherits from *UML1.4::Core::ModelElement*, but from *InfrastructureLibrary::Constructs::ValueSpecification*. The latter already has an association with *Type*, therefore the *UML1.4::Core::Classifier* metaclass has been replaced by *InfrastructureLibrary::Constructs::Type*. The most radical changes need to be made to the model from section 3.3.2, shown in figure 1-6, which contains the metamodel for calling model properties. Because the metaclasses *Attribute* and *AssociationEnd* are not part of the *Constructs* package, both are replaced by *StructuralFeature*. Furthermore, the

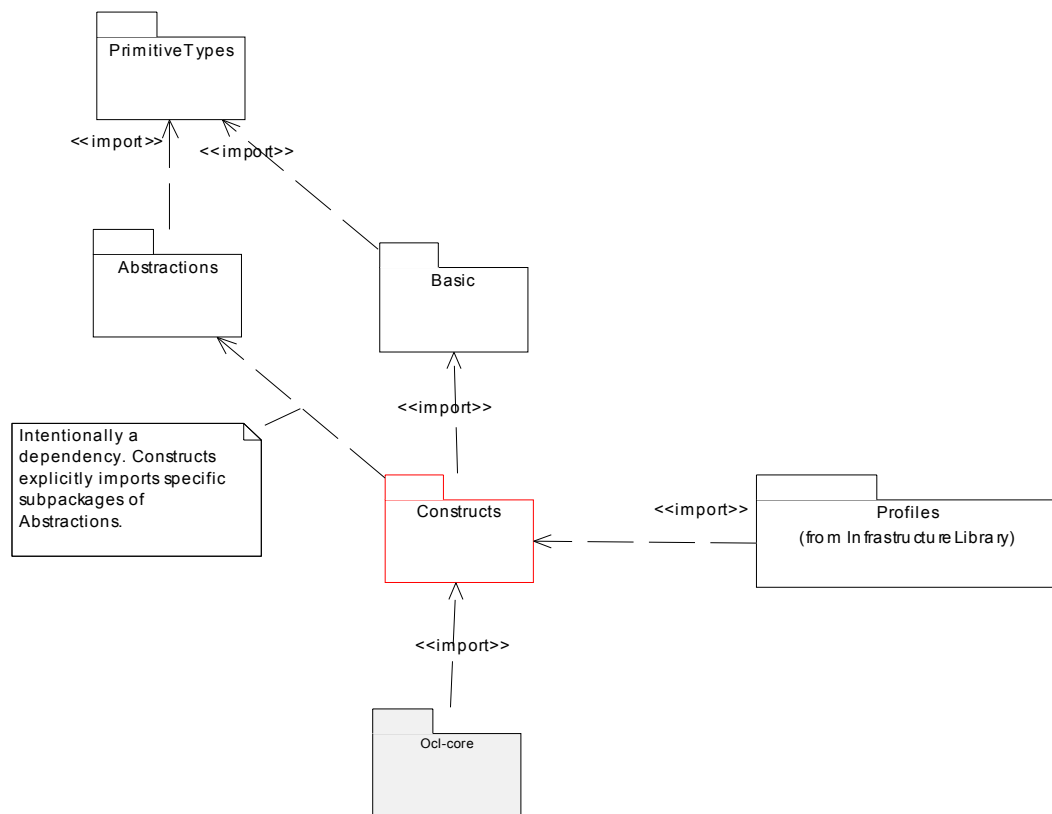


Figure 1-1 Relations of OCL-core and other subpackages of the InfrastructureLibrary package

UML1.4::Core::Operation metaclass has been replaced by its counterpart in the *Constructs* package. The result is shown in figure 1-7.

Without showing the diagrams it will be clear that in the metamodel for literal expressions, the association from the metaclass *EnumLiteralExp* to *UML1.4::Core::EnumLiteral* will need to refer to the metaclass *EnumerationLiteral* from *Constructs*.

Naturally, all well-formedness rules will need to be checked according to these changes. Likewise, the OCL concrete syntax, semantics, and standard library will need to be checked. We foresee only minor adjustments in those chapters. One aspect that needs adjustment is clear: the Infrastructure submission defines the semantics of associations using Bags. This needs to be taken into account in the semantics of *AssociationEndCall* in the OCL proposal. We do not foresee any technical problem, because the OCL semantics already support the notion of bags. The main task will be to make sure this is consistently changed,

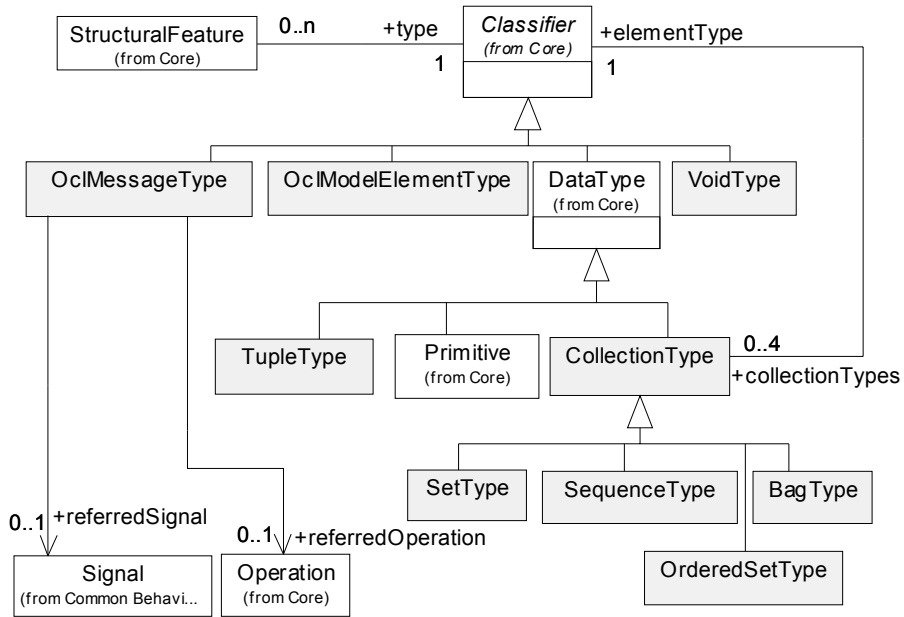


Figure 1-2 Old version of the abstract syntax kernel metamodel for OCL Types

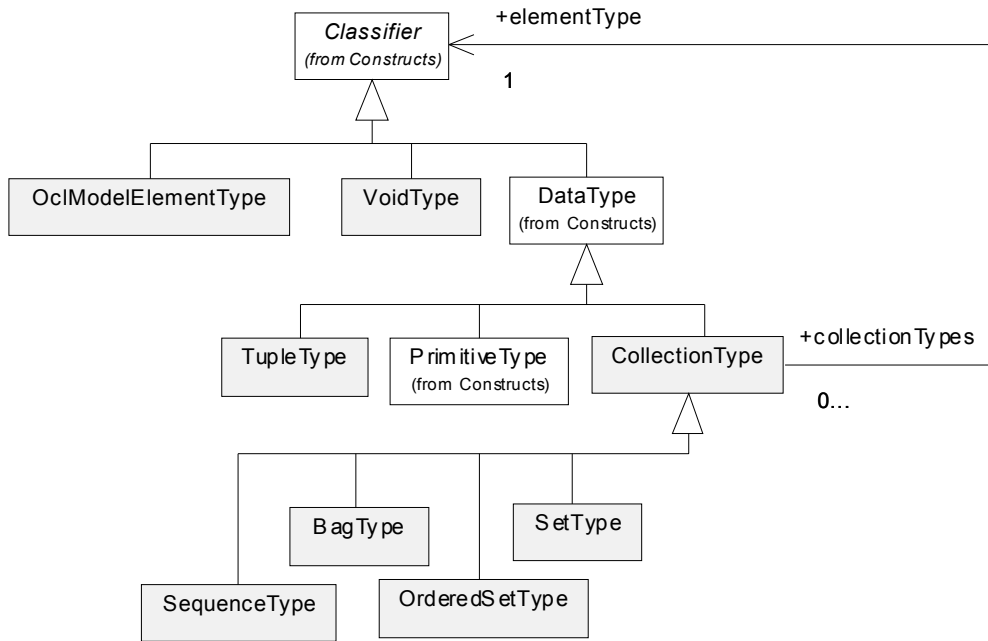


Figure 1-3 New version of the abstract syntax metamodel of the OCL-core package for OCL Types

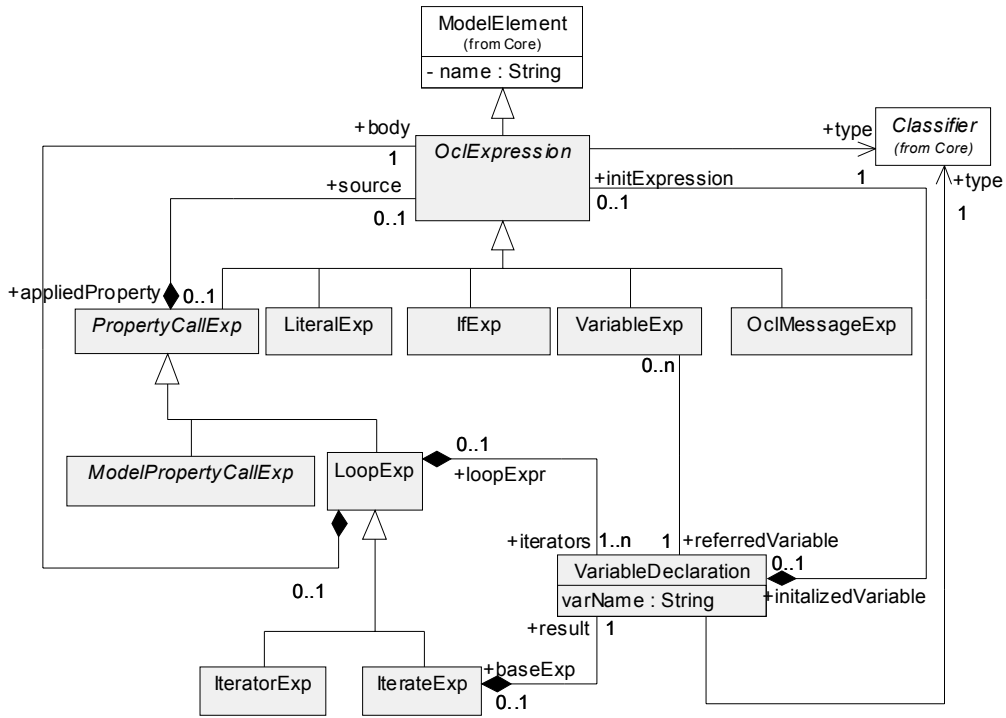


Figure 1-4 Old version of the basic structure of the abstract syntax kernel metamodel for Expressions

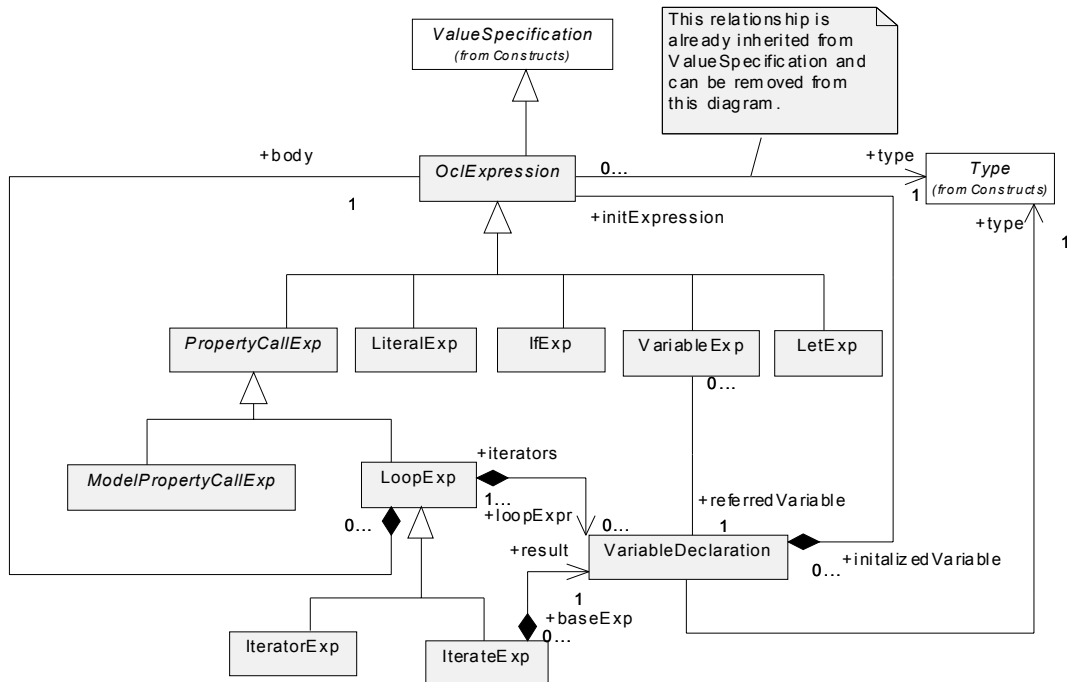


Figure 1-5 New version of the basic structure of the abstract syntax kernel metamodel of the OCL-core package for Expressions

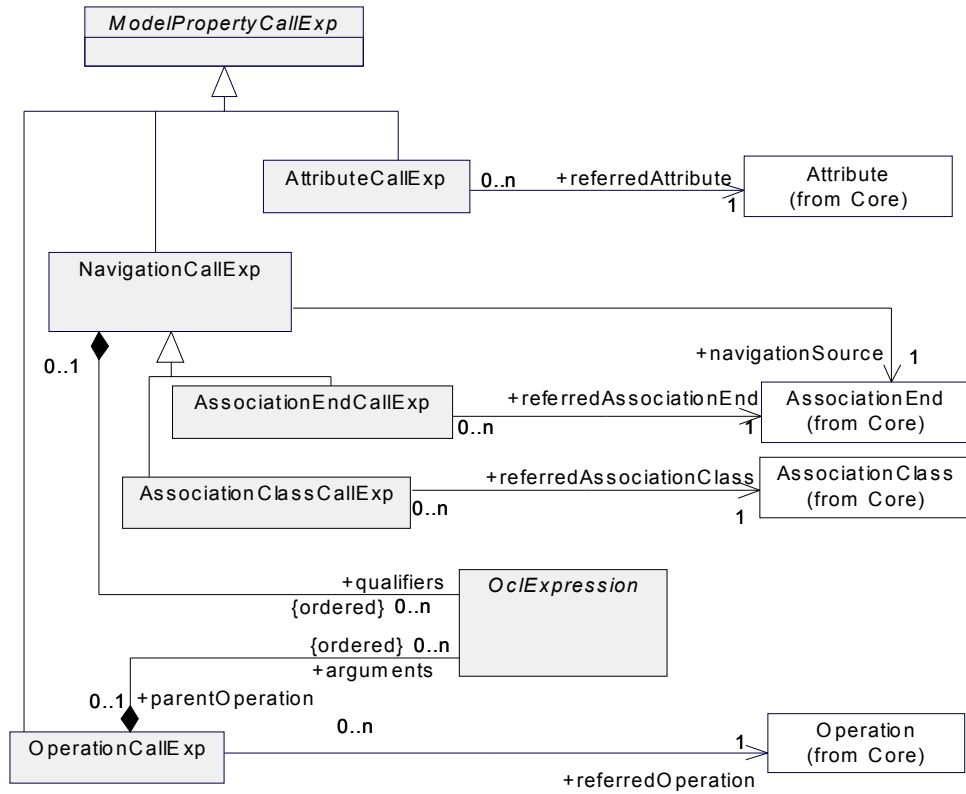


Figure 1-6 Old version of the basic structure of the abstract syntax for ModelPropertyCallExp

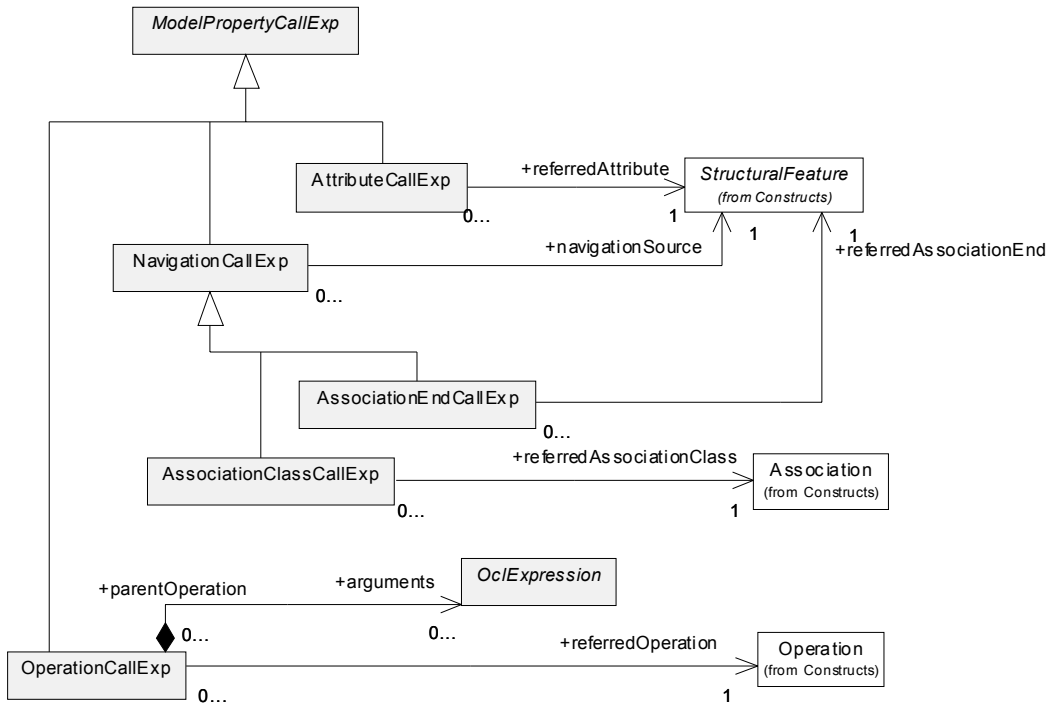


Figure 1-7 New version of the basic structure of the abstract syntax metamodel of the OCL-core package for ModelPropertyCallExp

1.3 USE OF OCL

Chapter 7 in the OCL submission deals with the position of OCL expressions within a model. As expressed in the submission this aspect of OCL depends highly on the UML Infrastructure. In section 5.1 "the meaning (semantics) of an OCL expression ... (is) ... defined as the value yielded by its evaluation in a given environment". And in section 7.1 the submission states that "in principle, everywhere in the UML specification where the term *expression* is used, an OCL expression can be used". This notion is expressed by the inheritance relation between *OclExpression* and *ValueSpecification*. Thus, truly every point in a model where a value specification is called for, an OCL expression can be used.

There are a number of placements of OCL expressions within a model that have a very specific meaning. Three of these have already been defined in the UML Infrastructure Library: preconditions, postconditions, and body-conditions, as shown in figure 1-8. The other forms of use that have been identified in chapter 7 of the OCL submission can be defined likewise, as shown in figure 1-9. The metaclass *ExpressionInOcl* is no longer needed and can be removed. Note that the additional associations with metaclass *Constraint* are the only changes that needs to be made in the UML Infrastructure in order to incorporate all aspects of the *OCL-core* package.

We advise to take special care in defining the semantics of the package merge with regard to constraints. This could prove to be an issue that needs substantial effort.

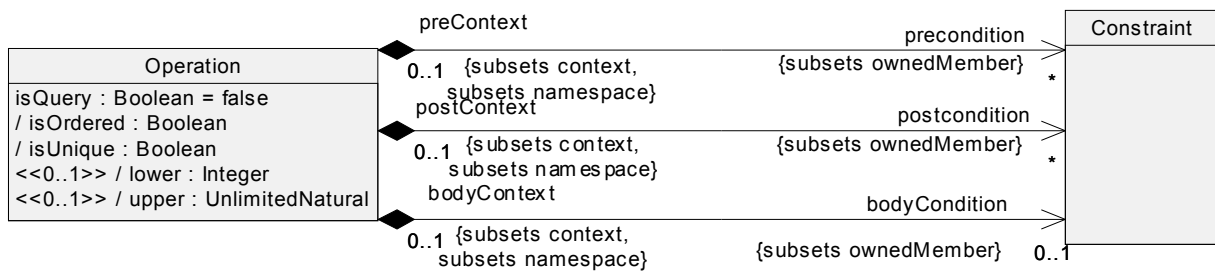


Figure 1-8 Precondition, postcondition, and bodycondition in Constructs package

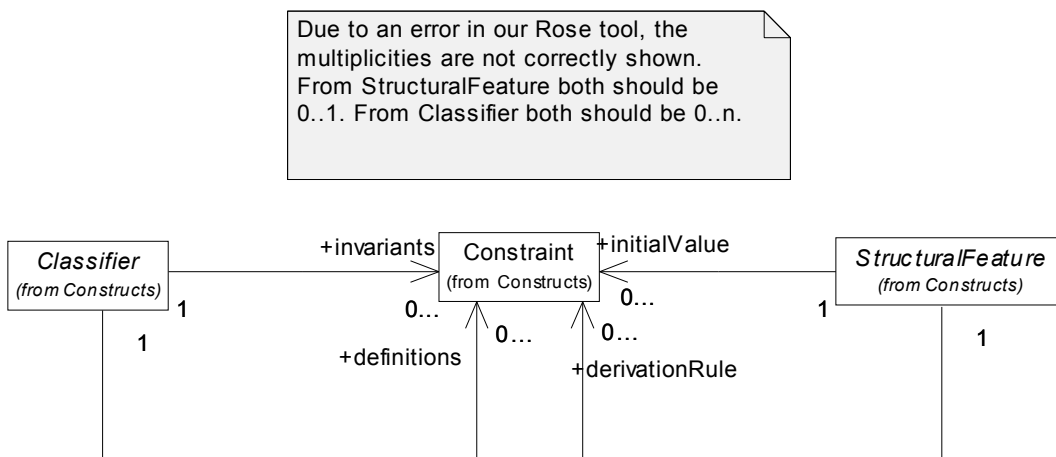


Figure 1-9 Additions to the UML metamodel to support other uses of OCL expression

1.4 REMAINING ISSUES

1.4.1 Integration of OCL and UML Superstructure

In the integration of OCL with the UML Superstructure three issues need to be resolved:

- OclMessage
- Qualifiers
- Query behavior with an OCL expression as specification

A preliminary assessment is that the metaclass *OclMessageExp* should be associated with *CallOperationAction* and *SendSignalAction*, instead of with *UML1.4::CommonBehavior::CallAction* and *UML1.4::CommonBehavior::SendAction* respectively. The other necessary changes are obvious.

1.4.2 A summary of Remaining issues

With the changes described above the integration of the UML infrastructure and *OCL-core* packages is straightforward. The list below describes a number of tasks that remain to be done.

- Decide whether *OCL-core* should be coupled with the Basic or the Constructs package. Or, at least, provide a more elaborate argumentation for the choice for the Constructs package.
- Make the distinction of OCL into an *OCL-core* and an *OCL-super* package clear and adjust the OCL submission accordingly.
- References to the *UML1.4::Classifier* have been changed into references to the *UML 2.0 InfrastructureLibrary::Core::Constructs::Classifier*. Decide whether this choice is correct or whether the *UML 2.0 InfrastructureLibrary::Core::Constructs::Type* metaclass should be used instead.
- Adjust the well-formedness rules in OCL abstract syntax.
- Check the additional operations defined on UML types, as in section 3.3.8 of the OCL submission.
- Adjust the mapping rules given in the OCL concrete syntax.
- Adjust the well-formedness rules in the OCL semantics.
- Check the definition of the OCL standard library.
- Rewrite the OCL submission chapter 7 "The Use of OCL Expressions in UML Models"
- Give complete mapping of concrete syntax from section 7.4 of the OCL submission to abstract syntax as in chapter 4 of the OCL submission.
- Adjust the *OCL-super* package to integrate with the UML 2.0 Superstructure, once the superstructure submission is approved.
- Define the semantics of the package merge in UML 2.0 with regard to constraints.

1.5 CONCLUSION

The OCL submission has been written with the future integration with the UML 2.0 in mind. As shown, it seems relatively easy to perform the integration along the lines described in this document. Undoubtedly a lot of smaller details will need to be taken care of, but we foresee no major issues.