QVT 1.2 Revision Task Force

Ballot 3

"Preview 1"

30 January 2014

QVT 1.2 RTF Formal Issues Resolution Vote - Ballot No. 3

Poll start date: Wednesday, 5 February 2014 (01:00 AM EDT - 06:00 GMT)

Poll closing date: Wednesday , 19 February 2014 (07:00 PM EDT - 24:00 GMT)

What is being voted on: Proposed issue resolutions for the set of issues listed in the tables on the following pages. The proposer is listed for each resolution, the full text of the issues and corresponding resolutions can be found in the section following the issue tables.

- Only officially registered RTF members in good standing are allowed to vote
- Voters who do not vote in two successive ballots lose their good standing and will be removed from the RTF membership; they can only be reinstated by a TC vote
- Quorum for a vote is half the registered RTF members
- Simple majority (of non-abstaining votes) decides the vote
- Votes should be sent via email to the chair of the RTF(ed@willink.me.uk) as well as to the qvt-rtf@omg.org list.
- Members can submit their vote anytime between the poll start date and the poll closing date.
- During the polling interval, members are allowed to change their votes. The most recent vote will be assumed to supersede all previous votes cast by a member.
- The possible ways to vote are:
  - Yes
  - No
  - Abstain (Note: “Abstain” does not influence the voting result but *does* count towards quorum)
- Votes can be cast either for individual issues or for the entire block (but not a mix of both)

Block Vote

<Company name> votes {Yes | No | Abstain} for the entire block of proposed issue resolutions identified below and specified in the appendix to this document.

Individual Issues Vote (NB: ONLY if you choose not to use the Block Vote option above!) <Company name> votes as follows on the proposed issue resolutions specified in and specified in the appendix to this document. Then vote for one issue resolution per line, like this:

12345 {Yes | No | Abstain}

No Vote

A brief explanation for each No vote should be provided.

QVT 1.2 RTF Membership

Representative	Organisation	Status
Manfred Koethe	88solutions	
Pete Rivett	Adaptive	
Bernd Wenzel	Fachhochschule Vorarlberg	
Michael Wagner	Fraunhofer FOKUS	
Jishnu Mukerji	Hewlett-Packard	
Didier Vojtisek	INRIA	
Xavier Blanc	Laboratoire Informatique de Paris 6	
Nicolas Rouquette	NASA	
Andrius Strazdauskas	No Magic, Inc.	
Edward Willink	Nomos Software	(CHAIR)
Victor Sanchez	Open Canarias, SL	
Laurent Rioux	THALES	

Revision Details

Table of Contents

Disposition: Resolved

Issue 11602: Section: 7.13

Source:

Vienna University of Technology (Johann Oberleitner, joe(at)infosys.tuwien.ac.at)

Summary:

I have built a transformation engine based on QVT relations. The QVT relations example in Annex A contains a 'function' PrimitiveTypeToSqlType at the end of the example in textual syntax. This example is the only relations example in the whole spec. Nowhere is the semantics of 'function' defined nor contains the grammar of the concrete syntax a function keyword. However, 'query' is defined. Is 'function' another name for 'query'?

Resolution:

QVT Base has a Function.queryExpression.

QVT 1.1 added the mapping from queryCS to Function.

Not 'nowhere' but we can do better.

Revised Text:

In 7.11.1.5 Function change

A function may be specified

to

Since a function is side effect free, it is often called a query. A function may be specified

Disposition: **Resolved**

Issue 11826: Section 7.11.2.3: Empty CollectionTemplateExp is useful

Source:

Nomos Software (Dr. Edward Willink, ed(at)willink.me.uk)

Summary:

7.11.2.3 and Figure 7.6 both impose a lowerBound of 1 on CollectionTemplateExp.member and CollectionTemplateExp.rest.

However, the concrete syntax permits an empty match. This empty match is exploited in the UnsharedWhenVarsToMgVars relation in Section 10.

Suggest:

Change CollectionTemplateExp.member to [0..*]

Change CollectionTemplateExp.rest to [0..1]

Add a sentence to clarify the empty match semantics.

Resolution:

Yes.

Revised Text:

In 7.11.2.3 CollectionTemplateExp change

`member : OclExpression [1..*] {composes}`

The expressions that the elements of the collection must have matches for. A special variable _ may be used to indicate that any arbitrary element may be matched and ignored.

`rest : Variable [1]`

The variable that the rest of the collection (i.e., excluding elements matched by member expressions) must match. A special variable _ may be used to indicate that any arbitrary collection may be matched and ignored.

to

`member : OclExpression [*] {composes}`

The expressions that the elements of the collection must have matches for. A special variable _ may be used to indicate that any arbitrary element may be matched and ignored. The expression may be omitted to restrict a match to an empty collection.

`rest : Variable [0..1]`

The variable that the rest of the collection (i.e., excluding elements matched by member expressions) must match. A special variable _ may be used to indicate that any arbitrary collection may be matched and ignored. The variable may be omitted to restrict a match to a collection with no elements other than those matched by the member expressions.

Disposition: **Resolved**

Issue 12518: errors and anomalies in QVT_1.0.mdl in the 07-07-08 ZIP

Source:

Nomos Software (Dr. Edward Willink, ed(at)willink.me.uk)

Summary:

Use of automated tooling to support comparison of the models developed initially as part of the Eclipse GMT/UMLX project and being transferred to the Eclipse QVT Declarative/QVT Operational Mappings Projects reveals the following errors and anomalies in QVT_1.0.mdl in the 07-07-08 ZIP.

Since the diagrams are printed from QVT_1.0.mdl all the QVT problems also occur in 08-04-03.

Textual errors in 08-04-03 cannot be analyzed automatically. There are so many that a thorough proof read is required combined with a statement that the diagrams only are normative

Resolution:

The QVT 1.0 diagrams and models were originally from the models in the QVT_1.0.mdl file. The diagrams rely on proprietary tooling. Unfortunately some independent evolution occurred and so there were many inconsistencies.

Consistent Ecore/EMOF files from Eclipse were endorsed as the QVT 1.1 non-normative files.

For QVT 1.2 the primary non-normative files are UML models derived from the QVT 1.1 Ecore files. The diagrams are redrawn from the UML using the Open Source Papyrus tool.

Revised Text:

Replace Fig xxx by …

<Diagrams to follow>

Disposition: **Resolved**

Issue 12522: Errors and anomalies in QVT 1.0 07-07-08 ZIP qvtbase.ecore.

Source:

Nomos Software (Dr. Edward Willink, ed(at)willink.me.uk)

Summary:

Use of automated tooling to support comparison of the models developed initially as part of the Eclipse GMT/UMLX project and being transferred to the Eclipse QVT Declarative/QVT Operational Mappings Projects reveals the following errors and anomalies in qvt_metamodel.emof.xml in the 07-07-08 ZIP.

Note that these errors and anomalies are not the same as those separately reported for the QVT_1.0.mdl from which the EMOF was notionally auto-generated.

An Ecore file resolving these anomalies is attached.

'nsPrefix' for 'QVTBase' should be 'qvtb' rather than 'qvtbase'

'nsURI' for 'QVTBase' should be ' http://schema.omg.org/spec/QVT/1.0/qvtbase.xml' rather than 'http://www.schema.omg.org/spec/QVT/1.0/qvtbase'

'name' for 'QVTBase' should be 'QVTBase' rather than 'qvtbase'

'abstract' for 'Domain' should be 'true' rather than 'false'

'abstract' for 'Rule' should be 'true' rather than 'false'

'lowerBound' for 'Domain.typedModel' should be '1' rather than '0'

'lowerBound' for 'Rule.transformation' should be '0' rather than '1'

'ordered' for 'Pattern.bindsTo' should be 'false' rather than 'true'

'ordered' for 'Pattern.predicate' should be 'false' rather than 'true'

'ordered' for 'Rule.domain' should be 'false' rather than 'true'

'ordered' for 'Transformation.modelParameter' should be 'false' rather than 'true'

'ordered' for 'Transformation.ownedTag' should be 'false' rather than 'true'

'ordered' for 'Transformation.rule' should be 'false' rather than 'true'

'ordered' for 'TypedModel.dependsOn' should be 'false' rather than 'true'

'ordered' for 'TypedModel.usedPackage' should be 'false' rather than 'true'

Unnavigable 'opposite' of 'Rule.overrides' should be modelled

Unnavigable 'opposite' of 'Transformation.extends' should be modelled

Unnavigable 'opposite' of 'TypedModel.dependsOn' should be modelled

Resolution:

These changes mostly affect non-normative files which were corrected when QVT 1.1 issued revised files based on Eclipse QVT contributions. However a few changes remain to be resolved in the main text.

No: Domain.typedModel lowerbound of 0 is correct; primitive domains have no TypedModel

Rule.transformation lowerbound change is Issue 11825.

Revised Text:

In Fig 7.4 change

- Rule.transformation multiplicity from 1 to 0..1.
- Transformation.modelParameter to not ordered.
- Transformation.overriden to overridden.

In 7.11.1.1 Transformation change

```
transformation: Transformation[1]
```

to

```
transformation: Transformation[0..1]
```

In 7.11.1.4 Rule change

```
transformation: Transformation[1]
```

to

```
transformation: Transformation[0..1]
```

In the non-normative files change

- Transformation.modelParameter to not ordered.

Disposition: **Resolved**

Issue 12523: Errors and anomalies in QVT 1.0 07-07-08 ZIP qvttemplate.ecore.

Source:

Nomos Software (Dr. Edward Willink, ed(at)willink.me.uk)

Summary:

Use of automated tooling to support comparison of the models developed initially as part of the Eclipse GMT/UMLX project and being transferred to the Eclipse QVT Declarative/QVT Operational Mappings Projects reveals the following errors and anomalies in qvt_metamodel.emof.xml in the 07-07-08 ZIP.

Note that these errors and anomalies are not the same as those separately reported for the QVT_1.0.mdl from which the EMOF was notionally auto-generated.

An Ecore file resolving these anomalies is attached.

'nsURI' for 'QVTTemplate' should be ' http://schema.omg.org/spec/QVT/1.0/qvttemplate.xml' rather than ' http://www.schema.omg.org/spec/QVT/1.0/qvtrelation'

'nsPrefix' for 'QVTTemplate' should be 'qvtt' rather than 'qvttemplate'

'name' for 'QVTTemplate' should be 'QVTTemplate' rather than 'qvttemplate'

'eType' for 'CollectionTemplateExp.rest' should be 'Variable' rather than 'OclExpression'

'CollectionTemplateExp.kind' should be undefined

'lowerBound' for 'CollectionTemplateExp.member' should be '0' rather than '1'

'lowerBound' for 'CollectionTemplateExp.rest' should be '0' rather than '1'

'lowerBound' for 'CollectionTemplateExp.referredCollectionType' should be '1' rather than '0'

'ordered' for 'CollectionTemplateExp.member' should be 'false' rather than 'true'

'ordered' for 'ObjectTemplateExp.part' should be 'false' rather than 'true'

Unnavigable 'opposite' of 'CollectionTemplateExp.member' should be modelled

Unnavigable 'opposite' of 'CollectionTemplateExp.rest' should be modelled

Unnavigable 'opposite' of 'PropertyTemplateItem.referredProperty' should be modelled

Unnavigable 'opposite' of 'PropertyTemplateItem.value' should be modelled

Unnavigable 'opposite' of 'TemplateExp.where' should be modelled

Resolution:

These changes mostly affect non-normative files which were corrected when QVT 1.1 issued revised files based on Eclipse QVT contributions. However a few changes remain to be resolved in the main text.

CollectionTemplateExp.rest/member lowerBound is Issue 11826.

PropertyTemplateItem.value multiplicity is right in diagram not text.

Revised Text:

In Fig 7.6 change

- Correct part/objContainer positions; should show ObjectTemplateExp.part
- Correct CollectionTemplateExp.referredCollectionType to multiplicity 1.
- Correct CollectionTemplateExp.member to multiplicity *.

- Correct CollectionTemplateExp.rest to multiplicity 0..1.

In 7.11.2.3 CollectionTemplateExp change

```
member : OclExpression [1..*] {composes}
…
rest : Variable [1]
```

to

```
member : OclExpression [*] {composes}
…
rest : Variable [0..1]
```

In 7.11.2.4 PropertyTemplateItem change

```
value: OclExpression [0..1] {composes}
```

to

```
value: OclExpression [1] {composes}
```

Disposition: **Resolved**

Issue 12524: Errors and anomalies in QVT 1.0 07-07-08 ZIP qvtrelation.ecore.

Source:

Nomos Software (Dr. Edward Willink, ed(at)willink.me.uk)

Summary:

Use of automated tooling to support comparison of the models developed initially as part of the Eclipse GMT/UMLX project and being transferred to the Eclipse QVT Declarative/QVT Operational Mappings Projects reveals the following errors and anomalies in qvt_metamodel.emof.xml in the 07-07-08 ZIP.

Note that these errors and anomalies are not the same as those separately reported for the QVT_1.0.mdl from which the EMOF was notionally auto-generated.

An Ecore file resolving these anomalies is attached.

'nsURI' for 'QVTRelation' should be ' http://schema.omg.org/spec/QVT/1.0/qvtrelation.xml' rather than ' http://www.schema.omg.org/spec/QVT/1.0/qvtrelation'

'nsPrefix' for 'QVTRelation' should be 'qvtr' rather than 'qvtrelation'

'name' for 'QVTRelation' should be 'QVTRelation' rather than 'qvtrelation'

'lowerBound' for 'RelationCallExp.argument' should be '2' rather than '0'

'lowerBound' for 'RelationCallExp.referredRelation' should be '1' rather than '0'

'lowerBound' for 'RelationDomain.pattern' should be '1' rather than '0'

'containment' for 'Relation.operationalImpl' should be 'true' rather than 'false'

'transient' for 'RelationImplementation.relation' should be 'true' rather than 'false'

'ordered' for 'Key.part' should be 'false' rather than 'true'

'ordered' for 'Relation.operationalImpl' should be 'false' rather than 'true'

'ordered' for 'Relation.variable' should be 'false' rather than 'true'

'ordered' for 'RelationDomain.defaultAssignment' should be 'false' rather than 'true'

'ordered' for 'RelationalTransformation.ownedKey' should be 'false' rather than 'true'

Unnavigable 'opposite' of 'Relation.when' should be modelled

Unnavigable 'opposite' of 'Relation.where' should be modelled

Unnavigable 'opposite' of 'RelationDomain.defaultAssignment' should be modelled

Unnavigable 'opposite' of 'RelationDomainAssignment.valueExp' should be modelled

Unnavigable 'opposite' of 'RelationDomainAssignment.variable' should be modelled

Resolution:

These changes mostly affect non-normative files which were corrected when QVT 1.1 issued revised files based on Eclipse QVT contributions. However a few changes remain to be resolved in the main text.

Revised Text:

In Fig 7.7

- Remove the partial diagram artifact at the bottom of the diagram
- Correct the RelationImplementation.relation multiplicity to 1

- Correct the RelationCallExp.referredRelation multiplicity to 1
- Correct the RelationCallExp.argument multiplicity to 2..*

In 7.11.3.1 RelationalTransformation change

```
key: Key [*] {composes}
```

to

```
ownedKey: Key [*] {composes}
```

In 7.11.3.2 Relation change

```
/domain: Domain [*] {composes} (from Rule)
```

to

```
/domain: RelationDomain [*] {composes} (from Rule)
```

In the non-normative files change

- RelationImplementation.relation multiplicity to 1

Disposition: **Resolved**

Issue 12525: Errors and anomalies in QVT 1.0 07-07-08 ZIP qvtcore.ecore.

Source:

Nomos Software (Dr. Edward Willink, ed(at)willink.me.uk)

Summary:

Use of automated tooling to support comparison of the models developed initially as part of the Eclipse GMT/UMLX project and being transferred to the Eclipse QVT Declarative/QVT Operational Mappings Projects reveals the following errors and anomalies in qvt_metamodel.emof.xml in the 07-07-08 ZIP.

Note that these errors and anomalies are not the same as those separately reported for the QVT_1.0.mdl from which the EMOF was notionally auto-generated.

An Ecore file resolving these anomalies is attached.

'nsURI' for 'QVTCore' should be ' http://schema.omg.org/spec/QVT/1.0/qvtcore.xml' rather than ' http://www.schema.omg.org/spec/QVT/1.0/qvtcore'

'nsPrefix' for 'QVTCore' should be 'qvtc' rather than 'qvtcore'

'name' for 'QVTCore' should be 'QVTCore' rather than 'qvtcore'

'eSuperTypes' for 'Assignment' should be 'Element' rather than nothing

'eSuperTypes' for 'EnforcementOperation' should be 'Element' rather than nothing

'abstract' for 'Assignment' should be 'true' rather than 'false'

'containment' for 'EnforcementOperation.operationCallExp' should be 'true' rather than 'false'

'containment' for 'Mapping.local' should be 'true' rather than 'false'

'transient' for 'Mapping.context' should be 'true' rather than 'false'

'Assignment.slotExpression' should be undefined

'PropertyAssignment.slotExpression' should be defined

'CorePattern.variable' should be defined

'Mapping.refinement' should be defined

'Mapping.refinement' should be the 'opposite' of 'Mapping.specification'

'ordered' for 'BottomPattern.assignment' should be 'false' rather than 'true'

'ordered' for 'BottomPattern.enforcementOperation' should be 'false' rather than 'true'

'ordered' for 'BottomPattern.realizedVariable' should be 'false' rather than 'true'

'ordered' for 'Mapping.local' should be 'false' rather than 'true'

'ordered' for 'Mapping.specification' should be 'false' rather than 'true'

Unnavigable 'opposite' of 'Assignment.value' should be modelled

Unnavigable 'opposite' of 'PropertyAssignment.targetProperty' should be modelled

Unnavigable 'opposite' of 'VariableAssignment.targetVariable' should be modelled

Resolution:

These changes mostly affect non-normative files which were corrected when QVT 1.1 issued revised files based on Eclipse QVT contributions. However a few changes remain to be resolved in the main text.

CorePattern.variable is Issue 10938

Assignment.slotExpression is Issue 11108

Revised Text:

In Fig 9.2

- add a CorePattern to Variable composition arc for CorePattern.variable
- Correct the RelationImplementation.relation multiplicity to 1

In Figure 9.3

- move the Assignment end of OclExpression.slotExpression to PropertyAssignment.
- Change the EnforcementOperation.bottomPattern multiplicity to 1.

In 9.17.1 CorePattern add

```
variable: Variable [*] {composes}
```

In 9.17.6 Mapping change

```
domain : Domain [*] {composes} (From QVTBase)
```

to

```
/domain : CoreDomain [*] {composes} (From QVTBase)
```

In 9.17.7 RealizedVariable change

```
bottomPattern : BottomPattern [*] {composes}
```

to

```
bottomPattern : BottomPattern [1]
```

In 9.17.8 Assignment change

```
bottomPattern: BottomPattern
```

to

```
bottomPattern : BottomPattern [1]
```

From 9.17.8 Assignment move

```
slotExpression: OclExpression [1] {composes}
```

An OCL expression identifying the object whose property value is to be assigned.

to 9.17.9 PropertyAssignment

```
slotExpression: OclExpression [1] {composes}
```

An OCL expression identifying the object whose property value is to be assigned.

In the non-normative files change

- EnforcementOperation.bottomPattern multiplicity to 1

Disposition: **Resolved**

Issue 12526: Errors and anomalies in QVT 1.0 07-07-08 ZIP imperativeocl.ecore.

Source:

Nomos Software (Dr. Edward Willink, ed(at)willink.me.uk)

Summary:

Use of automated tooling to support comparison of the models developed initially as part of the Eclipse GMT/UMLX project and being transferred to the Eclipse QVT Declarative/QVT Operational Mappings Projects reveals the following errors and anomalies in qvt_metamodel.emof.xml in the 07-07-08 ZIP.

Note that these errors and anomalies are not the same as those separately reported for the QVT_1.0.mdl from which the EMOF was notionally auto-generated.

An Ecore file resolving these anomalies is attached.

'nsURI' for 'ImperativeOCL' should be 'http://schema.omg.org/spec/QVT/1.0/imperativeocl.xml' rather than 'http://www.schema.omg.org/spec/QVT/1.0/imperativeocl'

'nsPrefix' for 'ImperativeOCL' should be 'impocl' rather than 'imperativeocl'

'name' for 'ImperativeOCL' should be 'ImperativeOCL' rather than 'imperativeocl'

'name' for 'TryExp.exceptClause' should be 'exceptClause' rather than undefined

'containment' for 'UnpackExp.targetVariable' should be 'false' rather than 'true'

'ordered' for 'DictLiteralExp.part' should be 'false' rather than 'true'

'lowerBound' for 'ReturnExp.value' should be '0' rather than '1'

'defaultValueLiteral' for 'AssertExp.severity' should be 'error' rather than undefined

'defaultValueLiteral' for 'VariableInitExp.withResult' should be 'false' rather than undefined

'eSuperTypes' for 'SwitchExp' should be 'ImperativeExpression' rather than 'CallExp','ImperativeExpression'

Unnavigable 'opposite' of 'AltExp.body' should be modelled

Unnavigable 'opposite' of 'AltExp.condition' should be modelled

Unnavigable 'opposite' of 'AssertExp.assertion' should be modelled

Unnavigable 'opposite' of 'AssignExp.defaultValue' should be modelled

Unnavigable 'opposite' of 'AssignExp.left' should be modelled

Unnavigable 'opposite' of 'AssignExp.value' should be modelled

Unnavigable 'opposite' of 'BlockExp.body' should be modelled

Unnavigable 'opposite' of 'CatchExp.exception' should be modelled

Unnavigable 'opposite' of 'ComputeExp.body' should be modelled

Unnavigable 'opposite' of 'ComputeExp.returnedElement' should be modelled

Unnavigable 'opposite' of 'DictionaryType.keyType' should be modelled

Unnavigable 'opposite' of 'DictLiteralExp.part' should be modelled

Unnavigable 'opposite' of 'DictLiteralPart.key' should be modelled

Unnavigable 'opposite' of 'DictLiteralPart.value' should be modelled

Unnavigable 'opposite' of 'ImperativeIterateExp.target' should be modelled

Unnavigable 'opposite' of 'ImperativeLoopExp.condition' should be modelled

Unnavigable 'opposite' of 'InstantiationExp.argument' should be modelled

Unnavigable 'opposite' of 'LogExp.condition' should be modelled

Unnavigable 'opposite' of 'OrderedTupleLiteralExp.part' should be modelled

Unnavigable 'opposite' of 'OrderedTupleLiteralPart.value' should be modelled

Unnavigable 'opposite' of 'OrderedTupleType.elementType' should be modelled

Unnavigable 'opposite' of 'RaiseExp.exception' should be modelled

Unnavigable 'opposite' of 'SwitchExp.alternativePart' should be modelled

Unnavigable 'opposite' of 'SwitchExp.elsePart' should be modelled

Unnavigable 'opposite' of 'TryExp.exceptClause' should be modelled

Unnavigable 'opposite' of 'TryExp.tryBody' should be modelled

Unnavigable 'opposite' of 'UnlinkExp.item' should be modelled

Unnavigable 'opposite' of 'UnlinkExp.target' should be modelled

Unnavigable 'opposite' of 'VariableInitExp.referredVariable' should be modelled

Unnavigable 'opposite' of 'WhileExp.body' should be modelled

Unnavigable 'opposite' of 'WhileExp.condition' should be modelled

Resolution:

These changes mostly affect non-normative files which were corrected when QVT 1.1 issued revised files based on Eclipse QVT contributions. However a few changes remain to be resolved in the main text.

CatchExp.exceptionVariableName/CatchExp.exception changes are Issue 15997

Remove UnpackExp is Issue 13268

Remove OrderedTupleType is Issue 13268

Remove OrderedTupleLiteralExp is Issue 13268

Remove OrderedTupleLiteralPart is Issue 13268

Add InstantiationExp.initializationOperation is Issue 19021

Revised Text:

In Fig 8.5

- Position * and {ordered} unambiguously for BlockExp.body
- Add Operation
- Add unidirectional reference from InstantiationExp to Operation
  forward role initializationOperation [0..1]
  reverse role instantiationExp [*]

In Fig 8.6

- remove spurious artifact at RHS
- add CatchExp.exceptionVariableName
- change CatchExp.exception multiplicity to +
- change ReturnExp.value multiplicity to 0..1
- remove UnpackExp and its associations

In Fig 8.7

- remove OrderedTupleType and its associations
- remove OrderedTupleLiteralExp and its associations
- remove OrderedTupleLiteralPart and its associations

In 8.2.2.7 ImperativeIterateExp change

```
target : Variable [0..1]
```

to

```
target : Variable [0..1] {composes}
```

In 8.2.2.8 SwitchExp change

```
elsePart : OclExpresion {composes} [0..1]
```

to

```
elsePart : OclExpression {composes} [0..1]
```

In 8.2.2.14 CatchExp and the QVT 1.1 models add

```
exceptionVariableName : String [0..1]
```

In 8.2.2.23 InstantiationExp add

```
initializationOperation : Operation [0..1]
```

The initialization operation that uses the arguments to initialize the object after creation. The initialization operation may be omitted when implicit initialization occurs with no arguments.

In 8.2.2.24 Typedef change

```
condition: OclExpression [1]{composes}
```

to

```
condition: OclExpression [0..1]{composes}
```

In 8.2.2.29 DictLiteralExp change

```
part : DictLiteralPart [*] {composes,ordered}
```

to

```
part : DictLiteralPart [*] {composes}
```

In the non-normative models change

- add CatchExp.exceptionVariableName
- change CatchExp.exception multiplicity to +
- add InstantiationExp.initializationOperation
- remove UnpackExp
- remove OrderedTupleType
- remove OrderedTupleLiteralExp
- remove OrderedTupleLiteralPart

Disposition: **Resolved**

Issue 12527: Errors and anomalies in QVT 1.0 07-07-08 ZIP qvtoperational.ecore.

Source:

Nomos Software (Dr. Edward Willink, ed(at)willink.me.uk)

Summary:

Use of automated tooling to support comparison of the models developed initially as part of the Eclipse GMT/UMLX project and being transferred to the Eclipse QVT Declarative/QVT Operational Mappings Projects reveals the following errors and anomalies in qvt_metamodel.emof.xml in the 07-07-08 ZIP.

Note that these errors and anomalies are not the same as those separately reported for the QVT_1.0.mdl from which the EMOF was notionally auto-generated.

An Ecore file resolving these anomalies is attached.

'nsURI' for 'QVTOperational' should be 'http://schema.omg.org/spec/QVT/1.0/qvtoperational.xml' rather than 'http://www.schema.omg.org/spec/QVT/1.0/qvtoperational'

'nsPrefix' for 'QVTOperational' should be 'qvto' rather than 'qvtoperational'

'name' for 'QVTOperational' should be 'QVTOperational' rather than 'qvtoperational'

'name' for 'MappingParameter.referredDomain' should be 'referredDomain' rather than 'refinedDomain'

'eType' for 'Module.entry' should be 'EntryOperation' rather than 'Operation'

'eSuperTypes' for 'ImperativeCallExp' should be 'OperationCallExp','ImperativeExpression' rather than 'OperationCallExp'

'eSuperTypes' for 'MappingOperation' should be 'ImperativeOperation' rather than 'ImperativeOperation','Operation','NamedElement'

'eSuperTypes' for 'ModelType' should be 'Class' rather than 'Class','URIExtent'

'eSuperTypes' for 'ResolveExp' should be 'CallExp','ImperativeExpression' rather than 'CallExp'

'upperBound' for 'MappingOperation.when' should be '1' rather than '-1'

'upperBound' for 'MappingOperation.where' should be '1' rather than '-1'

'lowerBound' for 'ModelType.metamodel' should be '1' rather than '0'

'defaultValueLiteral' for 'ImperativeCallExp.isVirtual' should be 'true' rather than undefined

'defaultValueLiteral' for 'ModelType.conformanceKind' should be 'effective' rather than undefined

'ordered' for 'Module.ownedVariable' should be 'false' rather than 'true'

'ordered' for 'OperationBody.variable' should be 'false' rather than 'true'

Unnavigable 'opposite' of 'ContextualProperty.initExpression' should be modelled

Unnavigable 'opposite' of 'ContextualProperty.overridden' should be modelled

Unnavigable 'opposite' of 'ImperativeOperation.overridden' should be modelled

Unnavigable 'opposite' of 'MappingBody.endSection' should be modelled

Unnavigable 'opposite' of 'MappingBody.initSection' should be modelled

Unnavigable 'opposite' of 'MappingOperation.disjunct' should be modelled

Unnavigable 'opposite' of 'MappingOperation.inherited' should be modelled

Unnavigable 'opposite' of 'MappingOperation.merged' should be modelled

Unnavigable 'opposite' of 'MappingOperation.refinedRelation' should be modelled

Unnavigable 'opposite' of 'MappingOperation.when' should be modelled

Unnavigable 'opposite' of 'MappingOperation.where' should be modelled

Unnavigable 'opposite' of 'MappingParameter.referredDomain' should be modelled

Unnavigable 'opposite' of 'ModelType.additionalCondition' should be modelled

Unnavigable 'opposite' of 'ModuleImport.importedModule' should be modelled

Unnavigable 'opposite' of 'Module.entry' should be modelled

Unnavigable 'opposite' of 'Module.ownedTag' should be modelled

Unnavigable 'opposite' of 'Module.ownedVariable' should be modelled

Unnavigable 'opposite' of 'ObjectExp.referredObject' should be modelled

Unnavigable 'opposite' of 'OperationBody.content' should be modelled

Unnavigable 'opposite' of 'OperationBody.variable' should be modelled

Unnavigable 'opposite' of 'OperationalTransformation.intermediateClass' should be modelled

Unnavigable 'opposite' of 'OperationalTransformation.intermediateProperty' should be modelled

Unnavigable 'opposite' of 'OperationalTransformation.modelParameter' should be modelled

Unnavigable 'opposite' of 'OperationalTransformation.refined' should be modelled

Unnavigable 'opposite' of 'OperationalTransformation.relation' should be modelled

Unnavigable 'opposite' of 'ResolveInExp.inMapping' should be modelled

Resolution:

These changes mostly affect non-normative files which were corrected when QVT 1.1 issued revised files based on Eclipse QVT contributions. However a few changes remain to be resolved in the main text.

ContextualProperty .initExpression is Issue 13270

ObjectExp.instantiatedClass is Issue 13276

ObjectExp. initializationOperation is Issue 19021

Revised Text:

In Fig 8.1

- Replace Operation by EntryOperation as target of Module.entry
- Change ModelType.metamodel multiplicity to 1..*

In Fig 8.2

- Disentangle MappingBody.endSection/OperationBody.content and {ordered}
- Add {ordered} to OperationBody.content

Move from 8.2.1.1 OperationalTransformation to 8.2.1.3 Module

```
ownedVariable : Variable [0..*] {composes}
```

The list of variables owned by this module.

In 8.2.1.14 ContextualProperty change

```
overridden: Property [1]
```

to

```
overridden: Property [0..1]
```

In 8.2.1.14 ContextualProperty add

initExpression: OclExpression [0..1] {composes}

In 8.2.1.15 MappingOperation change

```
isBlackbox: Boolean
```

to

```
/isBlackbox: Boolean (from ImperativeOperation)
```

In 8.2.1.21 MappingCallExp Superclasses change

```
OperationCallExp
```

to

```
ImperativeCallExp
```

In 8.2.1.24 ObjectExp change

```
/instantiatedClass: Class [0..1](from InstanciationExp)
```

to

```
/instantiatedClass: Class [1](from InstantiationExp)
```

In 8.2.1.24 ObjectExp add

```
body: Constructor[1] { composes }
```

The constructor to execute.

In 8.2.1.24 ObjectExp add

```
/initializationOperation : Constructor [0..1] (from InstantiationExp)
```

The constructor that uses the arguments to initialize the object after creation. The constructor may be omitted when implicit construction occurs with no arguments.

In the non-normative files change

- ImperativeCallExp.referredObject multiplicity to 1

Disposition: **Resolved**

Issue 13103: element creation and element attachment/detachment to/from an extent

Source:

Open Canarias, SL (Mr. E. Victor Sanchez, vsanchez(at)opencanarias.com)

Summary:

Suggestion: In the Operational Mappings language, element creation and element attachment/detachment to/from an extent should be seen and treated as two different and independent activities. Once an element is created via an ObjectExp, this element is usually attached to another element immediately afterwards, which becomes its container, so the ObjectExp semantics of assigning it to an explicit or inferred extent becomes an unnecessary overhead. And also there are other times when we need to assign to an extent some model elements that may have been created at an unrelated time. They could even exist prior to the transformation execution. A case where this is relevant is with the result of a 'Element::clone()' or 'Element::deepclone()' operation execution. Is it expected that these model elements must belong by default to the same model as the original? How to clone parts of an extent with direction kind == 'in'? How to make them become part of the Set of root elements of another, different extent?

Resolution:

This seems to be a misunderstanding. ObjectExp does not require a created object to be attached to an inferred extent.

This functionality seems to be present already.

Creation without attachment requires a null or null-valued extent during creation.

Attachment after creation can occur through an ObjectExp update to the required extent.

Re-attachment presumably occurs through an ObjectExp update to the new extent.

Detachment then naturally occurs through an ObjectExp update to a null-valued extent.

Revised Text:

In 8.2.1.24 ObjectExp add after the first paragraph

Object creation , initialisation and residence are separate activities.

Object creation occurs when the *referredObject* has a null value; it is skipped if the *referredObject* variable references an existing object.

Object initialization always occurs, but may be trivial if the *body* is empty.

Object residence is left unchanged when the *extent* is omitted, so object creation will normally result in an object without any residence; the residence will be established as soon as the created object is put at the target end of some composition relationship. An explicit object residence may be established by specifying the model parameter for the required extent as the *extent*. Specifying an *extent* with a null value ensures that the created object has no residence; this may remove the residence of a pre-existing object.

Disposition: **Resolved**

Issue 13182: QVTo Standard Library: Some operation's signatures seem to be erroneous.

Source:

Open Canarias, SL (Mr. Adolfo Sanchez-Barbudo Herrera, adolfosbh(at)opencanarias.com)

Summary:

** QVTo Standard Library: Some operation's signatures seem to be erroneous. **

- the name in the signature of the operation allSubojectsOfType (8.3.4.7) has a typo. Rename correctly
- - the returned value in the signature of the operation raisedException (8.3.6.4) should better be Exception.
- the asList (8.3.8.5) operation's signature is not correct for the types OrderedSet(T), Sequence(T), Bag(T). They shouldn't have any parameter.
- -P.S: Why is this operation included in section 8.3.8 Operations List. I would recomend the creation of new sections for each collection type, instead.
- - OCLStdlib already defines toLower and toUpper operations. Since these operations may be considered as side-effects operations. I should clarify one of the possible situations: 1. toLower and toUpper are not intended to be side-effect operations. Remove them from the section. 2. toLower and toUpper are always intended to be side-effect operations, so that OCL Operations are discarded. This must be clarified. 3. both (side-effect and free side-effect) operations, are available in QVTtransformations. In this case I would change the name of QVTo std lib operations to distinguish.
- - In section (8.3.9) lastToUpper must be renamed as lastToLower.
- -P.S: Why all the QVTo Std Lib operations have a subsection number, excepting String's operations?

Resolution:

allSubojectsOfType – see Issue 13989 resolution.

raisedException – Yes. But the Status operatuions are misleadingly under Transformation. Add a missing section heading.

asList(T) – Yes, supersedes Issue 19146 resolution.

PS – see Issue 19146 resolution.

toLower – Yes. Make it clear that all String operations are immutable. Redirect OCL 2.4 synonyms to OCL 2.4 deprecating the synonyms. WE can also fix some bad typos, Boolean rather than Integer/Real returns and references to the non-existent Float type.

lastToUpper – this seems to be a major bloat. Either name or description could change. Since the name exists with an obvious functionality correct the description.

PS Yes

Revised Text:

After 8.3.6.3 Transformation wait insert

<<new 8.3.7 number>> Status

The following operations may be used to inrterrogate the Status objects that synchronizes the end of a transformation.

In 8.3.6.4 Transformation raisedException change

Status::raisedException () : Class

to

Status::raisedException () : Exception

In the Issue 19146 replacement text, replacethe erroneous trailing (T) parameter by an empty () parameter list in the following signatures:

```
List(T)::add(T) : Void
List(T)::asList(T)
List(T)::clone(T)
List(T)::deepclone(T)
Collection(T)::asList(T)
Collection(T)::clone(T)
Collection(T)::deepclone(T)
Bag(T)::asList(T)
Bag(T)::clone(T)
Bag(T)::deepclone(T)
OrderedSet(T)::asList(T)
OrderedSet(T)::clone(T)
OrderedSet(T)::deepclone(T)
Sequence(T)::asList(T)
Sequence(T)::clone(T)
Sequence(T)::deepclone(T)
```

(Set needs no change.)

In the body of 8.3.9 replace

String::size () : Integer

The size operation returns the length of the sequence of characters represented by the object at hand.

Synonym operation: **length**()

by

String::length () : Integer

The length operation returns the length of the sequence of characters represented by the object at hand.

This is a synonym of the OCL String::size() operation. It is therefore deprecated.

In the body of 8.3.9 replace

String::toLower () : String

Converts all of the characters in this string to lowercase characters.

String::toUpper () : String

Converts all of the characters in this string to uppercase characters.

by

String::toLower () : String

Converts all of the characters in this string to lowercase characters.

This is a synonym of the OCL String::toLowerCase() operation. It is therefore deprecated.

String::toUpper () : String

Converts all of the characters in this string to uppercase characters.

This is a synonym of the OCL String::toUpperCase() operation. It is therefore deprecated.

In the body of 8.3.9 replace

String::lastToUpper () : String

Converts the last character in the string to a lowercase character.

by

String::lastToUpper () : String

Converts the last character in the string to an uppercase character.

In the body of 8.3.9 replace

String::matchFloat (i:Integer) : Boolean

Returns true if the string represents a float.

by

String::matchFloat (i:Integer) : Boolean

Returns true if the string represents a real.

This is a synonym of the matchReal() operation. It is therefore deprecated.

String::matchReal (i:Integer) : Boolean

Returns true if the string represents a real.

In the body of 8.3.9 replace

String::asInteger() : Boolean

Returns a Integer value if the string can be interpreted as as integer. Null otherwise.

String::asFloat() : Boolean

Returns a Float value if the string can be interpreted as as float. Null otherwise.

by

String::asInteger() : Integer

Returns a Integer value if the string can be interpreted as as integer. Null otherwise.

String::asFloat() : Real

Returns a Real value if the string can be interpreted as as real. Null otherwise.

This is a synonym of the asReal() operation. It is therefore deprecated.

String::asReal() : Real

Returns a Real value if the string can be interpreted as as float. Null otherwise.

In 8.3.9 give each operation a sub-sub-sub-section number as for other types.

Disposition: **Resolved**

Issue 13183: QVTo Standard Library. Clarification of the side-effect operations is needed.

Source:

Open Canarias, SL (Mr. Adolfo Sanchez-Barbudo Herrera, adolfosbh(at)opencanarias.com)

Summary:

** QVTo Standard Library. Clarification of the side-effect operations is needed. I would explicity clarify which operations may modify the object source on which the operations are called. All the stdlib operations must clarify: 1. if the operation acts on the own object or on a copy of the object. 2. which object(s) is(are) exactly returned (the own object, a (collection of) new object(s), a (collection of) referenced object(s)) For example: String::trim operation clearly says that creates a new object copy of itself which is modified and returned. However, String::firstToUpper operation may have several interpretations.

Resolution:

Issue 19146 resolution makes the behaviour of List clear.

Dict clarified below and a typo

String clarified below.

Revised Text:

In 8.3.7.1 Dictionary get change

The null value is returned is not present.

to

The null value is returned if k is not present. Modifying the returned value modifies the value stored in the dictionary.

In 8.3.7.4 Dictionary put change

Assigns a value to a key.

to

Modifies the dictionary by assigning a value to a key. Modifying the value modifies the value stored in the dictionary.

In 8.3.7.5 Dictionary clear change

Removes all values in the dictionary

to

Modifies the dictionary by removing all values.

In 8.3.7.7 Dictionary values change

Returns the list of values in a list. The order is arbitrary.

to

Returns a new list of the values in the dictionary. The order is arbitrary. Modifying the returned list does not modify the contents of the dictionary. Modifying the values in the list modifies the values stored in the dictionary.

In 8.3.7.8 Dictionary keys change

Returns the list of keys in a list. The order is arbitrary.

to

Returns a new list of keys to the dictionary. Modifying the returned list does not modify the contents of the dictionary. Modifying the values in the list may modify the keys to the dictionary contents giving

unpredictable behaviour. If mutable types such as List or Dictionary are used as Dictionary keys applications should take care to create clones where appropriate.

In 8.3.9 String change

All string operations defined in OCL are available.

to

All string operations defined in OCL 2.4 are available. An OCL String is immutable and so there are no operations that modify strings.

Disposition: **Resolved**

Issue 13267: Page 73, Section 8.2.1.10 OperationalTransformation.

Source:

Open Canarias, SL (Mr. Adolfo Sanchez-Barbudo Herrera, adolfosbh(at)opencanarias.com)

Summary:

Problem's text: "overriden: Operation [0..1]"

discussion: the overriden operation should be an ImperativeOperation. This is correctly showd in the diagram.

suggestion: Replace "Operation" by "ImperativeOperation".

Resolution:

Yes.

Already correct in the QVT 1.1 models.

Revised Text:

In 8.2.1.10 ImperativeOperation change

overridden: Operation [0..1]

to

overridden: ImperativeOperation [0..1]

Disposition: **Resolved**

Issue 13269: Page 75: Section 8.2.1.13 Constructor.

Source:

Open Canarias, SL (Mr. Adolfo Sanchez-Barbudo Herrera, adolfosbh(at)opencanarias.com)

Summary:

Problem's text: "/body: BlockExp [0..1] {composes} (from ImperativeOperation)

The expression serving to populate the object using the given parameters. This expression should necessarily be an ObjectExp instance.

discussion: This is not coherent with the ImperativeOperation definition. Body is an OperationBody, specifically a ConstructorBody.

Suggestion: Replace the text above by the following:

"/body: OperationBody [0..1] {composes} (from ImperativeOperation)

The operation body of the constructor. It should necessarily be a ConstructorBody instance.

Resolution:

Yes. But we can go all the way by modeling the ConstructorBody constraint.

Revised Text:

In 8.2.1.13 Constructor change

`/body: BlockExp [0..1]{composes}(from ImperativeOperation)`
The expression serving to populate the object using the given parameters. This expression should necessarily be an `ObjectExp` instance.

to

`/body: ConstructorBody [0..1]{composes}(from ImperativeOperation)`
The imperative implementation for this constructor.

Disposition: **Resolved**

Issue 13270: Page 75: Section 8.2.1.14 ContextualProperty.

Source:

Open Canarias, SL (Mr. Adolfo Sanchez-Barbudo Herrera, adolfosbh(at)opencanarias.com)

Summary:

missed text: there is a missed text related to the initExpression association.

discussion: If we have a look to the diagram in figure 8.2, we may realize that a description of the initExpression association is missed.

suggestion: include the following text in the association's description:

initExpression: OclExpression [0..1] {composes}

An optional OCL Expression to initialize the contextual property.

Resolution:

Yes.

Already correct in the QVT 1.1 models.

Revised Text:

In 8.2.1.14 ContextualProperty add

```
initExpression: OclExpression [0..1] {composes}
```

An optional OCL Expression to initialize the contextual property.

Disposition: **Resolved**

Issue 13276: Page 87: Section 8.2.1.24 ObjectExp.

Source:

Open Canarias, SL (Mr. Adolfo Sanchez-Barbudo Herrera, adolfosbh(at)opencanarias.com)

Summary:

Problem's text: /instantiatedClass: Class [0..1](from InstanciationExp)

discussion: wrong description.

Suggestion: replace the text above by "/instantiatedClass: Class [1](from InstantiationExp)"

note that in /extent reference, "(from InstanciationExp)" must be also replaced by (from instantiationExp)"

Resolution:

Yes and correct the erroneous bounds change on the spurious redefinition.

Revised Text:

In 8.2.1.24 ObjectExp and the QVT 1.1 models replace

```
/instantiatedClass: Class [0..1](from InstanciationExp)
```

Indicates the class of the object to be created or populated.

```
/extent: Variable [0..1](from InstanciationExp)
```

by

```
/instantiatedClass: Class [1](from InstantiationExp)
```

Indicates the class of the object to be created or populated.

```
/extent: Variable [0..1](from InstantiationExp)
```

Disposition: **Resolved**

Issue 13279: Page 89: Figure 8.6.

Source:

Open Canarias, SL (Mr. Adolfo Sanchez-Barbudo Herrera, adolfosbh(at)opencanarias.com)

Summary:

Problem's text:

1.ReturnExp::value association my have multiplicity [0..1] in the diagram

2.TryExp::catchClause association should be called exceptClause

Discussion:

1. It seems that the returned value of a return expression might be optional.

2. Since the diagram and textual descriptioon are different, I'm not sure which was the original intention of the name of this reference. Reading the description's text

and the opposite role name (exceptOwner), I guees that the name must be "exceptClause".

suggestion:

1.In ReturnExp::value association replace mutiplicity 1 by multiplicity [0..1]. In the text is well described.

2.In TryExp class change change the "catchClause" by "exceptClause"

- Page 90: Section 8.2.2.3 ComputeExp

Problem's text: body : OclExpression [1] {composes, ordered}

Discussion: Ordered doesn't make sense in a univalued reference.

Suggestion: remove "ordered".

Resolution:

Yes.

Yes.

Yes.

Already correct in the QVT 1.1 models.

Revised Text:

In Figure 8.6 replace the “catchClause” role from TryExp by “exceptClause”.

In Figure 8.6 replace the “1” multiplicity on ReturnExp.value by “0..1”.

In 8.2.2.3 ComputeExp replace

```
body : OclExpression [1] {composes, ordered}
```

by

```
body : OclExpression [1] {composes}
```

Disposition: **Resolved**

Issue 13281: Page 93: Associations Section 8.2.2.7 ImperativeIterateExp.

Source:

Open Canarias, SL (Mr. Adolfo Sanchez-Barbudo Herrera, adolfosbh(at)opencanarias.com)

Summary:

Problem's text: target : Variable [0..1]

discussion: composes is missed. In the diagram is correctly represented the association's feature.

suggestion: add the following text to end of the line "{composes}

Resolution:

Yes.

Already correct in the QVT 1.1 models.

Revised Text:

In 8.2.2.7 ImperativeIterateExp replace

```
target : Variable [0..1]
```

by

```
target : Variable [0..1] { composes }
```

Disposition: **Resolved**

Issue 13287: Page 105: Associations Section 8.2.2.24 Typedef.

Source:

Open Canarias, SL (Mr. Adolfo Sanchez-Barbudo Herrera, adolfosbh(at)opencanarias.com)

Summary:

Problem's text: condition: OclExpression [1]{composes}

discussion: the condition is optional.

suggestion: Replace "[1]" by "[0..1]".

Resolution:

Yes.

Already correct in the QVT 1.1 models.

Revised Text:

In 8.2.2.24 Typedef replace

```
condition: OclExpression [1]{composes}
```

by

```
condition: OclExpression [0..1]{composes}
```

Disposition: **Resolved**

Issue 13289: Page 106: Associations Section 8.2.2.29 DictLiteralExp.

Source:

Open Canarias, SL (Mr. Adolfo Sanchez-Barbudo Herrera, adolfosbh(at)opencanarias.com)

Summary:

Problem's text: part : DictLiteralPart [*] {composes,ordered}

discussion: Do the parts need to be ordered ?. The diagram doesn't show it.

suggestion: Remove ordered or update the diagram.

Resolution:

Yes. A dictionary does not need go be ordered.

Already correct in the QVT 1.1 models.

Revised Text:

In 8.2.2.29 DictLiteralExp replace

```
part : DictLiteralPart [*] {composes,ordered}
```

The list of parts contained by this dictionary.

by

```
part : DictLiteralPart [*] {composes}
The parts contained by this dictionary.
```

Disposition: **Resolved**

Issue 15977: abstract/concrete syntax for try/catch in clauses 8.2.2.13 & 8.2.2.14 lacks support for retrieving the exception caught.

Source:

NASA (Dr. Nicolas F. Rouquette, nicolas.f.rouquette(at)jpl.nasa.gov)

Summary:

Current abstract/concrete syntax for try/catch in clauses 8.2.2.13 & 8.2.2.14 lacks support for retrieving the exception caught.

That is, QVT1.1 is currently limited to the following style of try/catch logic:

```
try {
    // ...
} except (Exception) {
    // there is no syntax to bind the actual exception caught to a variable or to retrieve it in an except expression.
};
```

One possibility would be to introduce a variable in the catch expression (clause 8.2.2.14), e.g.:

```
try {
    // ...
} except (Exception e) {
    // do something with the exception caught: e
};
```

or:

```
try {
    // ...
} except (Exception1 e1, Exception2 e2) {
    // do something with the exception caught: e1 or e2
};
```

Resolution:

Yes. Unuseable caught exceptions are clearly of limited utility.

There can only be one exception variable name for many exception types; what is its type? Cannot be any of the listed types so it will have to be the common super type.

The inherited NamedElement::name could be used for the exception variable name, except that this could conflict with any policy that actually used the name to represent perhaps the text serialization of an OclExpression. Therefore introduce a new exceptionVariableName.

Of course the lower bound on the exception type should be 1. If a catch all is required then the type can be Exception. If a finally is required then we need a wrapper or specification enhancement.

Revised Text:

In 8.2.2.13 TryExp add

> The exceptClauses are searched in order to select the first exceptClause that provides an exception type to which the raised exception conforms. If an exceptClause is selected, its body is executed.

In 8.2.2.14 CatchExp add

The caught expression may be accessed in the body expression using the exceptionVariableName whose apparent (static) type is the most derived common super type of all catchable exception types.

In 8.2.2.14 CatchExp and the QVT 1.1 models add

```
exceptionVariableName : String [0..1]
```

The name by which the caught exception may be accessed.

In 8.2.2.14 CatchExp and the QVT 1.1 models change

```
exception: Type [*] {ordered}
```

to

```
exception: Type [+] {ordered}
```

In 8.4.7 change

```
<except> ::= 'except' '(' <scoped_identifier_list> ')' <expression_block>
```

to

```
<except> ::= 'except' '(' [<identifier> ':'] <scoped_identifier>
    [',' <scoped_identifier>]* ')' <expression_block>
```

Disposition: **Resolved**

Issue 19021: Inconsistent description about constructor names.

Source:

Open Canarias, SL (Mr. Adolfo Sanchez-Barbudo Herrera, adolfosbh(at)opencanarias.com)

Summary:

Problem:

Specification first says in the Constructor concept description: "The name of the constructor is usually the name of the class to be instantiated. However this is not mandatory. Giving distinct names allows having more than one constructor."

Later on in the Constructor notation: "The name of the constructor is necessarily the name of the context type"

This is inconsistent.

Discussion:

Indeed, the notation section statement seems to be correct since:

1. Looks like other programming languages, like Java.

2. More importantly, the instantiation expression would not be so obvious when constructing new objects, and would required to be changed.

Example:

If we have the following constructors:

constructor MyClass::MyClassConstructor(name : String) { name := name }

constructor MyClass::MyClass(name : String) { name := name + "v2" }

How can the instantiation expression refer the different constructor ?

- new MyClass("abc")

- new MyClassConstructor("abc")

- new MyClass::Constructor("abc")

The referred class in a InstantiationExp would not be enough. Changing instantiation expression to provide different name constructor doesn't seem sensible.

Proposed solution:

In section 8.2.1.13

Replace:

"A constructor does not declare result parameters. The name of the constructor is usually the name of the class to be

instantiated. However this is not mandatory. Giving distinct names allows having more than one constructor."

by

"A constructor does not declare result parameters and its name must be the name of the class to be instantiated."

Resolution:

This was discussed on https://bugs.eclipse.org/bugs/show_bug.cgi?id=421621.

Unless we abandon constructor diversity completely, the current AS imposes a needless implementation difficulty by requiring dynamic resolution of a statically known constructor. This can be

avoided by augmenting InstantiationExp.instantiatedClass with InstantiationExp.referredConstructor, which can refer to any statically determined constructor. We can therefore relax the contradictory restrictions on constructor name spelling.

Unfortunately Constructor is not available in ImperativeOCL. Promoting Constructor to ImperativeOCL would appear easy, unfortunaely its superclass ImperativeOperation is also not available. Promoting ImperativeOperation requires … too hard. So we must instead introduce InstantiationExp.referredOperatiojn and redefine it in ObjectExp.

Revised Text:

In 8.2.1.13 Constructor notation change

The name of the constructor is necessarily the name of the context type

to

The name of the constructor is usually the name of the context type

In 8.2.1.24 ObjectExp add

`/initializationOperation : Constructor [0..1] (from InstantiationExp)`

The constructor that uses the arguments to initialize the object after creation. The constructor may be omitted when implicit construction occurs with no arguments.

In 8.2.2.23 InstantiationExp add

`initializationOperation : Operation [0..1]`

The initialization operation that uses the arguments to initialize the object after creation. The initialization operation may be omitted when implicit initialization occurs with no arguments.

In Figure 8.5 add

Operation (from EMOF)
unidirectional reference from InstantiationExp to Operation
-- forward role initializationOperation [0..1]
-- reverse role instantiationExp [*]

Disposition: **Resolved**

Disposition: Closed, no change

Issue 12519: Errors and anomalies in QVT 1.0 07-07-08 ZIP qvt_metamodel.emof.xml.

Source:

Nomos Software (Dr. Edward Willink, ed(at)willink.me.uk)

Summary:

Use of automated tooling to support comparison of the models developed initially as part of the Eclipse GMT/UMLX project and being transferred to the Eclipse QVT Declarative/QVT Operational Mappings Projects reveals the following errors and anomalies in qvt_metamodel.emof.xml in the 07-07-08 ZIP.

Note that these errors and anomalies are not the same as those separately reported for the QVT_1.0.mdl from which the EMOF was notionally auto-generated.

EMOF files resolving these anomalies are attached.

Discussion:

QVT 1.1 issued revised files based on Eclipse QVT contributions.

Issue 12518: QVT 1.2 is providing non-normative UML files.

Disposition: **Closed, No Change**

Issue 12520: Errors and anomalies in QVT 1.0 07-07-08 ZIP emof.ecore.

Source:

Nomos Software (Dr. Edward Willink, ed(at)willink.me.uk)

Summary:

Use of automated tooling to support comparison of the models developed

initially as part of the Eclipse GMT/UMLX project and being transferred to the Eclipse QVT Declarative/QVT Operational Mappings Projects reveals the following errors and anomalies in emof.ecore in the 07-07-08 ZIP.

Note that these errors and anomalies are not the same as those separately reported for the QVT_1.0.mdl from which the Ecore was notionally auto-generated.

An Ecore file resolving these anomalies is attached.

'nsURI' for 'EMOF' should be 'http://schema.omg.org/spec/MOF/2.0/emof.xml' rather than 'http:///emof.ecore'

'name' for 'EMOF' should be 'EMOF' rather than 'emof'

'name' for 'Property.isID' should be 'isID' rather than 'isId'

'Factory' should be defined

'ReflectiveCollection' should be defined

'ReflectiveSequence' should be defined

'Comment.body' should be defined

'Factory.package' should be defined

'Element.tag' should be undefined

'eOpposite' for 'Tag.element' should be undefined

'lowerBound' for 'Operation.class' should be '0' rather than '1'

'lowerBound' for 'Type.package' should be '0' rather than '1'

'lowerBound' for 'Property.class' should be '0' rather than '1'

'ordered' for 'Class.superClass' should be 'false' rather than 'true'

'ordered' for 'Comment.annotatedElement' should be 'false' rather than 'true'

'ordered' for 'Element.ownedComment' should be 'false' rather than 'true'

'ordered' for 'Operation.raisedException' should be 'false' rather than 'true'

'ordered' for 'Package.nestedPackage' should be 'false' rather than 'true'

'ordered' for 'Package.ownedType' should be 'false' rather than 'true'

'ordered' for 'Tag.element' should be 'false' rather than 'true'

'defaultValueLiteral' for 'Class.isAbstract' should be 'false' rather than undefined

'defaultValueLiteral' for 'MultiplicityElement.isOrdered' should be 'false' rather than undefined

'defaultValueLiteral' for 'MultiplicityElement.isUnique' should be 'true' rather than undefined

'defaultValueLiteral' for 'MultiplicityElement.lower' should be '1' rather than undefined

'defaultValueLiteral' for 'MultiplicityElement.upper' should be '1' rather than undefined

'defaultValueLiteral' for 'Property.isComposite' should be 'false' rather than undefined

'defaultValueLiteral' for 'Property.isDerived' should be 'false' rather than undefined

'defaultValueLiteral' for 'Property.isReadOnly' should be 'false' rather than undefined

'Element.container()' should be defined

'Element.equals(object)' should be defined

'Element.get(property)' should be defined

'Element.getMetaClass()' should be defined

'Element.isSet(property)' should be defined

'Element.set(property,object)' should be defined

'Element.unset(property)' should be defined

'Extent.elements()' should be defined

'Extent.useContainment()' should be defined

'Factory.convertToString(dataType,object)' should be defined

'Factory.create(metaClass)' should be defined

'Factory.createFromString(dataType,string)' should be defined

'ReflectiveCollection.add(object)' should be defined

'ReflectiveCollection.addAll(objects)' should be defined

'ReflectiveCollection.clear()' should be defined

'ReflectiveCollection.remove(object)' should be defined

'ReflectiveCollection.size()' should be defined

'ReflectiveSequence.add(index,object)' should be defined

'ReflectiveSequence.get(index)' should be defined

'ReflectiveSequence.remove(index)' should be defined

'ReflectiveSequence.set(index,object)' should be defined

'Type.isInstance(object)' should be defined

'URIExtent.contextURI()' should be defined

'URIExtent.element(uri)' should be defined

'URIExtent.uri(element)' should be defined

Unnavigable 'opposite' of 'Class.superClass' should be modelled

Unnavigable 'opposite' of 'Element.ownedComment' should be modelled

Unnavigable 'opposite' of 'Package.nestedPackage' should be modelled

Unnavigable 'opposite' of 'Property.opposite' should be modelled

Discussion:

These changes affect non-normative EMOF files which were corrected when QVT 1.1 issued revised files based on Eclipse QVT contributions.

Disposition: **Closed, No Change**

Issue 12521: Errors and anomalies in QVT 1.0 07-07-08 ZIP essentialocl.ecore.

Source:

Nomos Software (Dr. Edward Willink, ed(at)willink.me.uk)

Summary:

Use of automated tooling to support comparison of the models developed initially as part of the Eclipse GMT/UMLX project and being transferred to the Eclipse QVT Declarative/QVT Operational Mappings Projects reveals the following errors and anomalies in emof.ecore in the 07-07-08 ZIP.

Note that these errors and anomalies are not the same as those separately reported for the QVT_1.0.mdl from which the Ecore was notionally auto-generated.

An Ecore file resolving these anomalies is attached.

'nsURI' for 'EssentialOCL' should be ' http://schema.omg.org/spec/QVT/1.0/essentialocl.xml' rather than ' http://www.schema.omg.org/spec/OCL/2.0/essentialocl'

'name' for 'EssentialOCL' should be 'EssentialOCL' rather than 'essentialocl'

'name' for 'ExpressionInOcl.contextVariable' should be 'contextVariable' rather than 'context'

'name' for 'Variable.representedParameter' should be 'representedParameter' rather than 'bindParameter'

'NavigationCallExp' should be defined

'OpaqueExpression' should be undefined

'TypeType' should be defined

'CollectionKind::Collection' should be defined

'eSuperTypes' for 'ExpressionInOcl' should be 'TypedElement' rather than 'OpaqueExpression'

'eSuperTypes' for 'PropertyCallExp' should be 'NavigationCallExp'

'eSuperTypes' for 'AnyType' should be 'Type' rather than 'Class','Type'

'lowerBound' for 'CollectionType.elementType' should be '1' rather than '0'

'upperBound' for 'ExpressionInOcl.parameterVariable' should be '-1' rather than '1'

'abstract' for 'CollectionType' should be 'false' rather than 'true'

'containment' for 'TupleLiteralPart.attribute' should be 'false' rather than 'true'

'ordered' for 'CollectionLiteralExp.part' should be 'false' rather than 'true'

'ordered' for 'ExpressionInOcl.parameterVariable' should be 'false' rather than 'true'

'ordered' for 'LoopExp.iterator' should be 'false' rather than 'true'

'ordered' for 'TupleLiteralExp.part' should be 'false' rather than 'true'

Unnavigable 'opposite' of 'CallExp.source' should be modelled

Unnavigable 'opposite' of 'CollectionRange.first' should be modelled

Unnavigable 'opposite' of 'CollectionRange.last' should be modelled

Unnavigable 'opposite' of 'EnumLiteralExp.referredEnumLiteral' should be modelled

Unnavigable 'opposite' of 'ExpressionInOcl.bodyExpression' should be modelled

Unnavigable 'opposite' of 'ExpressionInOcl.contextVariable' should be modelled

Unnavigable 'opposite' of 'ExpressionInOcl.parameterVariable' should be modelled

Unnavigable 'opposite' of 'ExpressionInOcl.resultVariable' should be modelled

Unnavigable 'opposite' of 'IfExp.condition' should be modelled

Unnavigable 'opposite' of 'IfExp.elseExpression' should be modelled

Unnavigable 'opposite' of 'IfExp.thenExpression' should be modelled

Unnavigable 'opposite' of 'IterateExp.result' should be modelled

Unnavigable 'opposite' of 'LoopExp.body' should be modelled

Unnavigable 'opposite' of 'OperationCallExp.argument' should be modelled

Unnavigable 'opposite' of 'OperationCallExp.referredOperation' should be modelled

Unnavigable 'opposite' of 'PropertyCallExp.referredProperty' should be modelled

Unnavigable 'opposite' of 'Variable.initExpression' should be modelled

Unnavigable 'opposite' of 'VariableExp.referredVariable' should be modelled

Discussion:

These changes affect non-normative EssentialOCL files which were corrected when QVT 1.1 issued revised files based on Eclipse QVT contributions.

Disposition: **Closed, No Change**

Issue 17538: Consider submitting the QVTO profile out of UML Profile for NIEM, section 9-2 to QVT 1.2

Source:

NASA (Dr. Nicolas F. Rouquette, nicolas.f.rouquette(at)jpl.nasa.gov)

Summary:

Section 9-2 in the UML Profile for NIEM Beta2 document describes an interesting diagrammatic notation for describing the salient organization of a QVTO transformation.

Based on the notation shown in figures 9-2, 9-3, 9-4 and others, this notation clearly involves some kind of UML profile for describing a QVTO transformation whose stereotypes

include <<OperationalTransformation>>, <<MappingOperation>>, <<disjuncts>> and <<inherits>>. The figures in section 9 make a compelling illustration of the utility of a UML Profile for QVTO Transformation.

I believe this UML profile for QVTO is a novel contribution of the UML Profile for NIEM FTF; unfortunately, the document does not describe it and this QVTO Transformation profile

is not mentioned anywhere in the UML Profile for NIEM inventory or in any of the machine readable artifacts.

Discussion:

This would be an interesting enhancement and could perhaps form an Annex. However it is beyond the scope for the active members of this RTF. There has been no response from other RTF members to a couple of help-wanted requests, so this enhancement can be closed for lack of interest.

Disposition: **Closed, No Change**

Disposition: Duplicate / Merged

Issue 13266: Page 72, Figure 8-2.

Source:

Open Canarias, SL (Mr. Adolfo Sanchez-Barbudo Herrera, adolfosbh(at)opencanarias.com)

Summary:

Problem's text: in MappingParameter class: "refiningParameter" and "refinedDomain".

discussion: while a mappingOperation refines a Relation, a mappingParamer should "refer" a RelationDomain. In the text of MappingParameter section, the concept of "refers" is used several time instead of "refines".

suggestion: replace "refiningParameter" and "refinedDomain" by "referringParameter" and "referredDomain".

Resolution:

Duplicates in part Issue 12527.

Disposition: **See issue 12527 for disposition**

Issue 13271: Page 83: Section 8.2.1.22 MappingCallExp.

Source:

Open Canarias, SL (Mr. Adolfo Sanchez-Barbudo Herrera, adolfosbh(at)opencanarias.com)

Summary:

Problem's text: Superclasses OperationCallExp
discussion: It should extend ImperativeCallExp instead. The diagram is well showed.
suggestion: Replate "OperationCallExp" by "ImperativeCallExp".

Resolution:

Duplicates in part Issue 12527.

Disposition: **See issue 12527 for disposition**

Disposition: Transferred

Disposition: Deferred

Issue 11690: Section: 7.13.5

Source:

Siegfried Nolte siegfried(at)siegfried-nolte.de

Summary:

The "import" feature of Relations Language is not yet explained. And there is no example for it, too. For instance what does the "unit" in "import <unit>;" mean ?

Discussion:

OCL 2.5 should resolve the semantics of an import for Complete OCL. The QVTr (and QVTc) at least syntaxes should be compatible.

Disposition: **Deferred**

Issue 12213: Relations Language: how will metamodels get into a transformation scrip

Source:

Siegfried Nolte siegfried(at)siegfried-nolte.de

Summary:

Concerning to Relations Language, how will the metamodels get into a transformation script ? Is there a technique similar to Operational Mappings using metamodel declaration and modeltypes ? The RL sample transformation script in annex A.1 (pp 164) doesn't declare the metamodels. The OM sample (A.2.3) does. There is some syntax for declaring and using metamodels and modeltypes in OM (pp118), there isn't for RL (pp38).

Initial Response

I don't think QVTo is any different to QVTr.

Although A.2.3 happens to provide syntax for metamodels these have names that are distinct from the subsequent transformation and are separated by an explanatory paragraph.

How short names such as "UML" are resolved to a particular version, location and representation of a meta-model is tool-specific.

http://www.eclipse.org/gmt/umlx/doc/EclipseAndOMG08/ModelRegistry.pdf

describes one re-usable solution to the problem. It is used by QVTr and QVTc editors.

Discussion:

I now consider the failure of QVTr, QVTc and Complete OCL to provide an ability to import a Document URI is a language bug and should not depend on external implementation magic. As for Issue 11690: OCL 2.5 should resolve the semantics of an import for Complete OCL. The QVTr (and QVTc) at least syntaxes should be compatible.

Disposition: **Deferred**

Issue 12370: Section 8.7.1a production rule seems to be missing

Source:

Open Canarias, SL (Mr. Adolfo Sanchez-Barbudo Herrera, adolfosbh(at)opencanarias.com)

Summary:

a production rule seems to be missing: <module_element> ::= <modeltype> Since, a transformation file can define several transformations and libraries (modules), it is desirable having the possibility of defining modeltypes exclusively to a module. These "local" modelTypes should belong to the scope of a module, and it shouldn't be accessible to the remaining defined modules (unless the use of extension mechanisms is specified).

Discussion:

It is not clear whether this is necessary or just a convenience.

Disposition: **Deferred**

Issue 13054: MOF-QVT 1.0: 7.11.3.6 (and 7.11.1.1) BlackBox operation signature difficulties

Source:

Nomos Software (Dr. Edward Willink, ed(at)willink.me.uk)

Summary:

In 7.11.3.6 (and 7.8) the ordering of parameters is implied but not explicit. Presumably the first is the output (enforced direction) so:

package QVTBase

context TypedModel

def: allUsedPackage() : Set(EMOF::Package)

= self.dependsOn.allUsedPackage()->asSet()->union(self.usedPackage)

endpackage

package QVTRelation

context RelationImplementation

inv RootNodeIsBoundToRootVariable : self.inDirectionOf.allUsedPackage()->includes(self.impl.ownedParameter->at(1).type._package)

endpackage

This is not satisfied by almost the last line of RelToCore in 10.3:

enforce domain core me:OclExpression{} implementedby CopyOclExpession(re, me);

which seems to have output second.

More significantly CopyOclExpession(re, me) is not meaningful as a black box signature, since it is a static function and EMOF has no way to define static functions. Perhaps it is a query, for which the declaration was omitted from the example. If this is the case, it should be made clear that black box operations must be declared internally as queries and bound externally to static operations of the transformation.

This semantic clumsiness could be resolved, if, within a transformation,

relation, domain or query, self is bound to a transformation instance. Queries would then be normal non-static operations of the transformation (class) and the implementedBy operation call would be a normal implicit call via self of a non-static transformation operation or query. (A RelationCallExp would also deviate less from OCL than it currently does.) This would also allow a transformation to extend a Utility class that provided the required black box operations. Since queries and relations are declarative, it is not obvious that there need be any prohibition on the content of an extended Class; if the Class has properties, these cannot mutate during a query or relation match, so the properties are ok; they might even permit useful behavioural tailoring. For instance an 'inherited' Boolean mangledNames property could influence the mapping of names between input and output.

The RelToCore example can then be mended by declaring that:

RelToCore(...) extends utils::CopyUtilities

and externally binding the utils model name to a package that has a CopyUtilities class with suitable a CopyOclExpession operation.

Discussion:

I would like to see a working implementation of this before resolving.

Disposition: **Deferred**

Issue 13082: current abstract syntax of ImperativeOCL introduces a couple of unclear situations

Source:

Not recorded

Summary:

Major Problem:

(1) The current abstract syntax of ImperativeOCL introduces a couple of unclear situations. This may lead to incompatible QVT implementations.

Further Problems:

(2) Control flow constructs introduced by ImperativeOCL are redundant compared with existing conventional OCL constructs.

(3) Several OCL equivalence rules break when ImperativeOCL is present.

Detailed problem description:

(1) The current abstract syntax of ImperativeOCL introduces a couple of unclear situations. This may lead to incompatible QVT implementations. In the abstract syntax, ImperativeOCL expressions / statements are inherited from OclExpression. Therefore, conventional OCL expressions may (and will) contain sub-expressions that are actually ImperativeOCL expressions. In conventional OCL, the interpretation of an expression under a given environment is a value. In ImperativeOCL, the interpretation of an expression is a value and a new environment (state,variables). This extended interpretation is not given for conventional OCL expressions, leading to undefined operational semantics of those expressions.

Example: Given the following compute expression:

```
compute(z:Boolean) {
var x : Boolean := true
var y : Boolean := true
if ((x:=false) and (y:=false)) { ... }
z := y
}
```

What is the value of this expression: is it true or false (It depends on whether the 'and' operator is evaluated short-circuit or strict). The situation is similar for the evaluation of the other logical connectives, forAll, and exists when these expressions contain imperative sub-expressions.

(2) Control flow constructs introduced by ImperativeOCL are redundant compared with existing conventional OCL constructs. Some of the new language features in ImperativeOCL such as forEach and the imperative conditional are not really necessary. Their effect can already be achieved using conventional OCL expressions:

For example:

```
company.employees->forEach(c) { c.salary := c.salary * 1.1}
```

has the same effect as

```
company.employees->iterate(c; r:OclAny=Undefined |
c.salary := c.salary * 1.1
)
```

and

```
if ( x < 0 ) { x := 0 } else { x := 1 } endif
```

is the same as

if x < 0 then x := 0 else x := 1 endif

(3) Several OCL equivalence rules break when ImperativeOCL is present.

In conventional OCL, several equivalence rules well known from logic hold. Allowing OCL expression to contain imperative sub-expressions breaks these equivalence rules.

Examples:

let x=e1 in e2 equiv. e2 { all occurences of x replaced by e1 }
e1 and e2 equiv. e2 and e1

These equivalences do not necessarily hold if e1 or e2 are allowed to have side-effects.

Proposed solution:

(A) - (The cheap solution.) State textually that conventional OCL expressions (as described in the OMG OCL spec.) are not allowed to have side effects unless used as part of a top level ImperativeOCL expression. Therefore, even in a system supporting ImperativeOCL, class invariants, and pre- and postconditions (e.g.) will not be allowed to contain ImperativeOCL sub-expressions.

State explicitly that the redundant flow control statements have been introduced (solely) to write concise imperative programs and that the side-effect free forms of conditional evaluation ('if-then-else-endif' and 'iterate') shall not be used to program side-effects (instead, the ImperativeOCL forms shall be used).

(B) - (Major rework.) Rework the abstract syntax to reuse OCL expressions by composition rather than by inheritance. Imperative expressions (=> rename to 'statements') then may contain sub-statements and OCL expressions; OCL expressions are reused unchanged from the OCL spec (no imperative sub-expressions, no side-effects).

These issues have been discussed on the MoDELS 2008 OCL Workshop, more details can be found at http://www.fots.ua.ac.be/events/ocl2008/PDF/OCL2008_9.pdf

Discussion:

A rework may be necessary to clarify whether the claim that QVT (and consequently QVTo) is an extension of OCL needs to be amended.

Disposition: **Deferred**

Issue 13158: QVT Relations and working with stereotypes.

Source:

Siegfried Nolte siegfried(at)siegfried-nolte.de

Summary:

QVT Relations and working with stereotypes: Is there something like a QVT-R standard library with methods on stereotypes in it? There is none in the specification; compare QVT-R, there are some methods. Are there some else options for accessing stereotypes with QVT-R?

Discussion:

OCL 2.5 should provide stereotype support that can be extended if necessary by QVTr

Disposition: **Deferred**

Issue 13168: Typedef aliases issue.

Source:

Open Canarias, SL (Mr. Adolfo Sanchez-Barbudo Herrera, adolfosbh(at)opencanarias.com)

Summary:

Creating aliases seems to be a good idea specially when dealing with complex types. However, for that purpose, it is not clear to me that we need to create a (useless) typedef just to represent an alias in the concrete syntax. In practice aliases are very useful when writing transformations (in concrete syntax), but they are useless in the abstract syntax (extra unnecessary classes in modules). One may still think that a typedef as an alias (no extra condition) may be needed to add operations to existing types, as specification suggests doing in order to include new operations for the OCL standard library predefined types. However, this still can be done by means of helpers. Suggestions: - Remove the statements related to aliases in the section 8.2.2.24 - Make Typedef.condition reference be mandatory in Figure 8.7 since, now, Typedef in the abstract syntax will be used to constrain existing types. - Add statements in the concrete syntax section, to clarify the use of aliases to rename existing types. Maybe, a new keyword (like alias) could be included to avoid confusions with the typedef one. - Change the grammar to adapt it to these suggestions. - Clarify in the Standard Library section that helpers will be used to add new operations to the OCL Std Lib predefined type.

Discussion:

It is hard to tackle this properly until the OCL type system is clearer. OCL 2.5 should define what it means to prepare a loaded metamodel for OCL usage. This definition can form the basis of a QVT clarification..

Disposition: **Deferred**

Issue 13180: section (8.3.2) is very confusing for the reader

Source:

Open Canarias, SL (Mr. Adolfo Sanchez-Barbudo Herrera, adolfosbh(at)opencanarias.com)

Summary:

his section (8.3.2) is very confusing for the reader. What does synonym means ? - Is just like a shorthand of the concrete syntax ? - Is just a new operation included in the QVTo Standard Library semantically equivalent?. Ideally, just one type/operation must exist (the OCL predefined type and the operation of an OCL predefined type), so that, writing Void (a type) or asType (operation) should produce the same AST as if I write OclVoid or oclAsType. With this idea, I'm really puzzled with the third paragraph of the section. Please revise this section so that it is less confusing.

Discussion:

Once OCL 2.5 provides an extensible modeled OCL Standard Library, there will be no need for this to be more than additional library definitions.

Disposition: **Deferred**

Issue 13181: ** QVTo Standard Library.

Source:

Open Canarias, SL (Mr. Adolfo Sanchez-Barbudo Herrera, adolfosbh(at)opencanarias.com)

Summary:

** QVTo Standard Library. Some operation's returned values would better return the TemplateParameterType **

Several stdlib operations would be better changed to avoid doing unnecessary castings on the returned value of the operation.

For AnyType::oclAsType(oclType) operation I could write:

var aClass : Class := anotherVar.oclAsType(Class);

However, for Model::objectsOfType(OclType) operation I can't do:

var aClassSet : Set(Class) := aModel.objectsOfType(Class);

I have to do the following, instead:

var aClassSet : Set(Class) := aModel.objectsOfType(Class).oclAsType(Set(Class));

Therefore, for end-user usability, I propose exploiting TemplateParameterType and changing some QVTo Standard Library operations

Element::subobjectsOfType(OclType) : Set(T)

Element::allSubobjects(OclType) : Set(T)

Element::subobjectsOfKind(OclType) : Set(T)

Element::allSubobjectsOfKind(OclType) : Set(T)

Element::clone() : T

Element::deepclone() : T

Model::objectsOfType(OclType) : Set(T)

Model::copy() : T

Model::createEmptyModel(): T

Note: this approach is made in the Object::asOrderedTuple() : OrderedTuple(T) operation.

Discussion:

This is the approach taken by the OCL Standard Library modeling in Eclipse OCL using the OclSelf type, which has a well-defined meaning whereas TemplateParameterType is magic.

A proper QVT Library model should use the modeling capabilities to be provided by OCL 2.5.

Disposition: **Deferred**

Issue 13252: QVTo Standard Lybrary and typedefs Issue. Extending OCL predefined types.

Source:

Open Canarias, SL (Mr. Adolfo Sanchez-Barbudo Herrera, adolfosbh(at)opencanarias.com)

Summary:

As interpretated from the especification (pag 104), the way of adding new operations to OCL predefined types is creating new Typedef instances

which must have the OCL predefined type as the base type. The new operations are added to this new typedef. However there are several problems:

1. The specification doesn't provide any name for these typedefs.

2. The specification doesn't specify which type (QVT typedef or OCL predefined type) should be used when referencing such OCL predefined types in a QVTo transformation.

Solution for 1).

Suggestion a: Name the typedef with the same name of the base type. This provokes name's clash with the predefined type's name, due to there are two different types from two standard libraries

which have the same name. A possible solution, would be expliciting that typedefs (aliases) will never clash its name with its base type.

Suggestion b: Name the tpyedef with a different name, such as QVToXXXXXX or XXXX_Alias.

Solution for 2).

Suggestion a: Taking the typedef as the referenced type in QVTo transformations.

Suggestion b: Taking the OCL predefined type as the referenced type in QVTo transformations.

Suggestion c: Considering resolution of issue 13168, so that only OCL predefined exists, and therefore, the only type which can be referenced.

It's a little bit weird having 2 different types (a type, and its alias typedef) which represent just the same type, specially when they are related by a reference.

My solution's preference in order are:

Suggestion c: Just having one type to refer.

Suggestion a: Since the typedef "extends" the behaviour of the predefined type (adding new operations), the former must be the referred one.

Suggestion b: The OCL predefined type is referenced, but we must take into account that the operations added to the typedef are also available.

Discussion:

It is hard to tackle this properly until the OCL type system is clearer. OCL 2.5 should define what it means to prepare a loaded metamodel for OCL usage. This definition can form the basis of a QVT clarification..

Disposition: **Deferred**

Issue 14640: QVT 1.1 QVTr syntax mapping (correction to Issue 10646 resolution).

Source:

Nomos Software (Dr. Edward Willink, ed(at)willink.me.uk)

Summary:

The numerous problems identified in http://www.omg.org/archives/qvt-rtf/msg00094.html need to be addressed.

Apologies too for the long email. This is a very difficult area where precision is important. Provision of this resolution demonstrates the need for the resolution, but unfortunately the resolution has an erroneous precision that will make QVTr 1.1 unimplementable whereas QVTr 1.0 is just mildly ambiguous and conflicting.

Please do not include the resolution in QVT 1.1 without significantrework.

I found the definition of the "checkonly"/"enforce" to isCheckable/isEnforceable helpful, although different to three of my own intuitive guesses based on attempts to avoid errors in ModelMorf and Rel2Core examples.

The problems identified below are the result of a local review of the resolution. In the absence of a coherent Environment semantics it has not been possible to perform a global review. In particular, I was unable to review the specification for the arguably redundant bindsTo.

[Incidentally, the same resolution approach is needed for QVTc and QVTo].

Disambiguating rules

The resolution uses a similar approach to the OCL 2.0 specification, but neglects to provide any disambiguating rules although many are needed.

Environments

OCL and the resolution employ environments to carry definitions specific to particular CS constructs, so that a CS reference may be resolved with the aid of an environment or the AST.

In EssentialOCL, all definitions are resolvable with in the immutable AST with the exception of let and iterator expressions for which a nestedEnvironment() is used to carry the extra variable declaration with the aid of addElement(). The nestedEnvironment supports name occlusion from outer scopes.

In CompleteOCL, further definitions may be introduced by a definition constraint. The OCL specification provides no precise insight into how an environment changes to accommodate the definition. Should the name be added to the AST or to the environment? Is the name available for forward reference?

Environment::addElement is defined as a query operation thereby inhibiting side effects. Consequently usage such as

XX.env = YY.env.addElement(Z.name, Z.ast, false)

leaves the environment of YY unchanged and creates an extended environment for XX.

A series of such usages creates a series of progressively elaborated environments that support backward but not forward referencing. This was not a clear requirement of the QVT 1.0 specification and it prevents any variable declarations being introduced in an object template tree being resolved through environments in other object template trees or predicate expressions.

Imposition of no forward referencing seems very undesirable, particularly since the order of domains is imposed by the model parameter order. Imagine a Java program in which all methods had to be defined in a no-forward reference order.

As noted above, CompleteOCL neglected to define how AST definitions were added to the AST. QVTr must solve this problem since QVTr defines a hierarchically scoped AST rather than an annotation of a pre-existing AST.

I recommend a two stage approach. The inherited attributes section should first compute an environment from the pushed-down parent environment augmented by pull-ups from child constructs so that a complete immutable and consequently unambiguous environment is associated with each construct and then pushed-down to the children. During the pull-up the environment acquires a mapping from name to future-AST. During the push-down residual future-AST attributes are populated to give a valid AST.

Reference resolution

OCL uses lookup functions to resolve variable references. It is necessary either to overload the lookup functions so that the the distinct QVTr variable definition sites can be located in the AST, or to use some form of Environment::addElement where each definition is defined so that resolution in the environment is possible.

Details

=======

Throughout, many disambiguating rules are needed to define illegal semantics. For instance "any" is often used to select a syntactically valid value. Corresponding usage in the OCL specification has a disambiguating rule to clarify what the consequence of not "one" is.

My current best attempt at Disambiguating Rules is attached.

Environment::addElement takes three arguments, the third being a mayBeImplicit argument. This has been omitted throughout without explanation.

identifierCS

OCL defines SimpleNameCS. A degenerate mapping from identifierCS to SimplenameCS is required.

topLevelCS

The 'imported transformation' environment element is later referenced as 'imported transformations'.

Typo: TransformationListCS for transformationListCS in Synthesized attributes.

importListCS

Semantics of import conflicts must be defined.

unitCS

Typo: ast is not a Set.

Surely the import is of packages (enumerations or operations)or at least transformations (QVTo implementations) rather than necessarily relational-transformations?

transformationCS

---

ownedTag is not synthesized.

keyDeclListCS

Typo: wrong font in synthesized attributes

modelDeclCS

The [B] grammar:

modelDeclCS ::= modelIdCS ':' '{' metaModelIdListCS '}'

is missing.

keyDeclCS

Synthesized attributes appear to have experienced a copy and paste error while providing distinct part and oppositePart left hand sides.

keyPropertyCS

The synthesized attributes poke the parent.

Suggest: it would be clearer for the parent to gather and distribute children similar to the relation/query allocation by transformationCS.

relationCS

Transformation.extends does not appear to be transitive.

topQualifierCS

Suggest: a boolean or enumerated value rather than a string.

domainListCS

Typo: missing indentation.

primitiveTypeDomainCS

---

isCheckable, isEnforceable not synthesized.

objectTemplateCS

Variable needs to be added to relation to provide a container.

Variable needs to be added to relation environment to provide visibility.

collectionTemplateCS

Variable needs to be added to relation to provide a container.

Variable needs to be added to relation environment to provide visibility.

Suggest: last two if guards are redundant.

restCS

Variable needs to be added to relation to provide a container.

Non-_ named variable needs to be added to relation environment to provide visibility.

memberCS

Variable needs to be added to relation to provide a container.

Non-_ named variable needs to be added to relation environment to provide visibility.

whenCS

---

predicateListCS should be optional.

whereCS

---

predicateListCS should be optional.

ExtOclExpressionCS

This is not present in the QVTr or OCL grammar.

Presumably it represents the QVTr extension to OCL's OclExpressionCS.

However it is an extension, since at least RelationCallExpCS can be used in an ordinary OclExpressionCS using "not" or "and".

[A], [B], [C] should therefore follow on from OCL's

[A], [B], [C]..., [I].

RelationCallExpressionCS

How is a RelationCallExpressionCS distinguished from an OperationCallExpCS?

Discussion:

It is hard to tackle this properly until we have the model-driven specification generated technology that OCL 2.5 will pioneer.

Disposition: **Deferred**

Issue 15376: QVT 1.1 8.1.10 Errors in Examples

Source:

Nomos Software (Dr. Edward Willink, ed(at)willink.me.uk)

Summary:

The second example contains

"if result then return;"

which has a non-boolean condition expression and a missing endif.

In the first example, it is not clear that the revisit adds rather than overwrites.

In the third and fourth examples it is not clear why the second pass reuses the context for the first rather than creates new objects.

Discussion:

Help wanted.

Disposition: **Deferred**

Issue 15390: QVT 1.1 8 Unclear mapping operation characteristics

Source:

Nomos Software (Dr. Edward Willink, ed(at)willink.me.uk)

Summary:

8.1.4 states "a mapping operation is always a refinement of an implicit relation" but 8.2.15 defines "refinedRelation: Relation [0..1] The refined relation, if any." Clearly a contradiction.

8.1.4. provides the REL_PackageToSchema example of how an implicit relation might be defined, but no other examples or synthesis rules are supplied.

enforce and checkonly appear in the REL_PackageToSchema example indicating that these define execution mode of a QVTo trnasformation, but there does not appear to be any description of how a transformation might be executed to for instance update an output model. Perhaps the 'output' parameter is 'inout' for create/update but 'out' for create/overwrite.

Discussion:

Help wanted.

Disposition: **Deferred**

Issue 15411: Unclear transformation rooting condition

Source:

Nomos Software (Dr. Edward Willink, ed(at)willink.me.uk)

Summary:

"The starting point is a major flaw in all three QVT languages at present and enabled Perdita Stevens to correctly conclude in http://www.springerlink.com/content/9x368617317l3q87/ that QVTr and QVTc are incompatible. QVT currently provides no way to distinguish whether for instance a check-mode transformation is a query of whether a transformed input pattern can be discovered in the output (e.g. a database lookup), or a validation that the transformed input exactly matches the output (e.g. an already transformed check). Both facilities are useful and so when a QVT transformation is invoked the invoker needs to specify what I call the 'rooting' condition in addition to the direction

Discussion:

This requires some research work.

Disposition: **Deferred**

Issue 15417: Rule Overriding in QVTr.

Source:

Dr. Maged Elaasar, melaasar(at)gmail.com

Summary:

The abstract syntax of QVTr allows a rule to be an override of another rule.

Rule::overrides: Rule [0..1]

The rule that this rule overrides.

The concrete syntax of QVT allows it too:

<relation> ::= ['top'] 'relation' <identifier>
['overrides' <identifier>]
'{'
....
'}'

However, the only semantics I can see for 'overrides' is in clause 7.11.1.4 that says:

"A rule may conditionally override another rule. The overriding rule is executed in place of the overridden rule when the overriding conditions are satisfied. The exact semantics of overriding are subclass specific. "

Questions:

1- Whtat are the overriding conditions? are they implied or specified and if so how?

2- I have not seen any other discussion of overrding in a subclass or Rule so not sure what is meant by "The exact semantics of overriding are subclass specific"?

3- I have not seen any example of using 'overrides' what so ever in the spec, shouldn't there be one?

4 - What is the semantics of overriding? is it related to inheritance in the OO sense ? I think QVTr needs a good "inheritance" model where you can relations can be called polymorphically.

Discussion:

[Comments from inadvertently raised Issue 15524]

You have reached the edge of the specification as written.

1: Yes
2: Yes
3: Yes
4: Yes

I gave some consideration to this for UMLX.

I felt that an abstract 'rule' could define a 'subrule' obligation, which would require an identical match signature, since if the override was narrower it would not fulfill the obligation and if it was wider the additional width would not be permitted by the invocation of the abstract 'rule'.

I felt that all concrete rules should always be matched to ensure that addition of extended functionality did not change previous behaviour. This complies with UMLX's all maximal matches philosophy. Keys in QVTr, Evolution Ids in UMLX can ensure that derived rules reuse inherited matches.

I think a transformation being both a package and a class introduces some difficult compatibility issues to be studied.

Transformation extension is also poorly defined giving additional imprecision when considering the combination of transformation extension and rule override.

My ideas for UMLX were not complete but I think that they may be sounder than QVTr's.

[If Transformation is just a Class, one area for study vanishes.]

Disposition: **Deferred**

Issue 15523: QVTr already has queries but they are much less user friendly than e.g. MOFM2T's equivelent.

Source:

Nomos Software (Dr. Edward Willink, ed(at)willink.me.uk)

Summary:

QVTr already has queries but they are much less user friendly than e.g. MOFM2T's equivelent for which the first parameter is a hidden self, or indeed QVTo.

Perhaps something closer to Complete OCL would do, allowing def'd attributes or operations.

Discussion:

Perhaps it is sufficient to allow a QVTr query to be invoked with its first argument as a syntactical source rather than argument.

Disposition: **Deferred**

Issue 15886: Specification of deletion semantics.

Source:

Institute for Defense Analyses (Dr. Steven Wartik, swartik(at)ida.org)

Summary:

I'm having trouble with the semantics of DELETE on p. 189 of the QVT Specification (v1.1). It reads in part:

FORALL OBJVAR IN MAKESET(<DOMAIN_K_VARIABLE_SET>) (

…

AND BELONGSTO(OBJVAR, MAKESET(<DOMAIN_K_VARIABLE_SET>))

I guess I don't understand MAKESET and BELONGSTO. First of all, <DOMAIN_K_VARIABLE_SET> is already a set, so what's the MAKESET function do? Second, the FORALL iterates OBJVAR over the results of the same MAKESET that BELONGSTO tests. So how can BELONGSTO be false? That is, I would assume BELONGSTO is defined as follows:

BELONGSTO(e, S) ° e ÎS

except that under this definition the expression above is always satisfied.

Any and all help appreciated. Thank you very much.

Discussion:

I see no justification for not using OCL to express the Annex B semantic. The OCL could then form part of real tests.

I see little prospect of progress here till we have a conformant working QVTr implementation.

Disposition: **Deferred**

Issue 18323: Trace data for an 'accessed' transformation

Source:

Christopher Gerking, cgerking(at)campus.upb.de

Summary:

The spec should clarify the interaction of

1.) explicit instantiation + execution of 'accessed' transformations, and

2.) trace records / resolving.

The following questions are of interest:

How does the initial trace for an 'accessed' transformation look like? Does it reuse the records previously created by the 'accessing' transformation, or does an 'accessed' transformation always start on an empty trace?

How does an 'accessed' transformation affect the trace? Are the trace records post-visible that were created during execution of the 'accessed' transformation, or is the trace of the 'accessing' transformation unchanged?

Both issues heavily affect the results of resolve-operations. Resolving object references after executing an 'accessed' transformation would be very practical.

Discussion:

Help wanted.

Disposition: **Deferred**

Issue 18324: No trace data for disjuncting mapping

Source:

Christopher Gerking, cgerking(at)campus.upb.de

Summary:

Trace data creation is specified to happen "at the end of the initialization section".

For disjuncting mappings, the initialization section is never executed, which prevents any trace data from being stored.

As a consequence, no resolution via resolve-in-expressions is possible on the disjuncting mapping, due to the missing trace record. This is problematic, since disjunction should be transparent from a resolver's point of view, i.e. it should not make a difference for resolution whether a mapping disjuncts or not.

Hence, some clarification is required whether trace records are deliberately avoided for disjuncting mappings (for whatever reason), or whether trace data must be created in another place than the init section in case of a disjuncting mapping.

Discussion:

Help wanted.

Disposition: **Deferred**

Issue 18325: Intermediate data not allowed for libraries

Source:

Christopher Gerking, cgerking(at)campus.upb.de

Summary:

The QVT spec says that "an operational transformation may use for its definition intermediate classes and intermediate properties."

Is intermediate data actually restricted to plain transformations? In other words, is intermediate data unsupported for libraries? The eclipse QVTo implementation sticks to this interpretation and actually supports intermediate data only for plain transformations, which is a limitation I don't see a reason for.

Discussion:

Help wanted.

Disposition: **Deferred**

Issue 18363: Undefined semantics for unsatisfied "when" and "where" in inherited mapping

Source:

Levy Siqueira, levy.siqueira(at)gmail.com

Summary:

In operational QVT, it is not clear what happens when the "when" or "where" clause of an inherited mapping does not hold.

Suggestion:

Considering inheritance is a type (or, maybe, a synonym) of generalization, it would be expected that the semantics of inheritance mimics the semantics of generalization in MOF. The UML, which defines generalization used by CMOF and EMOF, states: "... features specified for instances of the general classifier are implicitly specified for instances of the specific classifier. Any constraint applying to instances of the general classifier also applies to instances of the specific classifier." (UML Infrastructure 2.4.1, p.51). If the "when" and "where" clauses are considered as features of a mapping, the clauses of the inherited mapping should be implicitly specified. Similarly, if they are considered as constraints applying to a mapping, the clauses defined in the inherited mapping should apply to the inheriting mapping. Therefore, a possible solution in both situations is to consider that the "when" and "where" clauses must hold in the inheriting mapping.

Commentary:

An interesting discussion is if something similar to the Liskov substitution principle should be applied in this situation.

Discussion:

Yes. This is a very fundamental principle that should form part of the philosophical underpining in the first couple of paragraphs of the QVTo language exposition. (ditto QVTc, QVTr).

It is disgraceful that the specification of transformation extension and rule refinement is so poor.

For QVTc and QVTr I favour a principle that extension provides extension not replacement. But that may be too restricting.

QVTo is more permissive and practical so a pure principle may be inappropriate.

Research needed.

Disposition: **Deferred**

Issue 18912: Inconsistent multiple inheritance.

Source:

Nomos Software (Dr. Edward Willink, ed(at)willink.me.uk)

Summary:

The pragmatic decision to define Module/Transformation as inheriting both Package and Class violates UML inheritance.

For a Package: self.nestingPackage.nestedPackages->includes(self)

For a Class:

self.package.ownedTypes->includes(self)

But self cannot have two containers.

The problem is easily resolved by extending only Package and adding those Class features that are actually required.

Discussion:

A quick experiment suggests that Class features are important but Package features are incidental, so making Transformation extend just Class and requiring a containing Package to provide context could be a good solution. Further investigation is required.

(Similar problem for FunctionParameter).

Disposition: **Deferred**

Issue 19019: List and Dict are Classes rather than DataTypes

Source:

Nomos Software (Dr. Edward Willink, ed(at)willink.me.uk)

Summary:

Intuitively List and Dict are objects, so if you pass them to a mapping/query/helper as an inout parameter, the object in the caller may be updated by the call.

This is the behaviour of a Class Instance not a DataType Value.

Please use the open https://bugs.eclipse.org/bugs/show_bug.cgi?id=420150 to discuss this topic.

Discussion:

It would be nice to clean this up, but very dangerous to do so without evaluating the consequences for a real implementation and typical transformations.

Disposition: **Deferred**

Issue 19022: ObjectExp Abstract Syntax misses a ConstructorBody.

Source:

Open Canarias, SL (Mr. Adolfo Sanchez-Barbudo Herrera, adolfosbh(at)opencanarias.com)

Summary:

Problem:

ObjectExp enhances an InstationExp with additional abstract syntax and semantics what respect to the association of the object to be created/updated to a variable.

Likewise, the object expression provides concrete syntax to specify the set of expressions to be executed in order to intialize/update the properties of the object to be created. However this is not reflected in the abstract sytanx.

Discussion:

Indeed, ObjectExp should contain a containment reference to comprise the block of expressions used to initialize/update the object properties.

The best AS candidate is ConstructorBody. Note that this is supported by the own description of the ConstructorBody (section 8.2.1.18):

"A constructor body contains the implementation of a constructor operation or the implementation of an inline constructor

(see ObjectExp)."

This ConstructorBody should be optional, as a result of an enhancement I'll raise in a different issue.

Proposed resolution:

Add the following association to ObjectExp in section 8.2.1.24

body: ConstructorBody [0..1] {composes}

The object expression body comprising the expressions to initialize (or update) the properties of the object to be instantiated (or updated).

Resolution:

See also 19023. The InstantiationExp class seems to be poorly characterized causing difficulties for its derived classes. A clean up of this area needs to be prototyped before resolving this issue.

Disposition: **Deferred**

Issue 19023: Enhance ObjectExp to allow constructors invocation.

Source:

Open Canarias, SL (Mr. Adolfo Sanchez-Barbudo Herrera, adolfosbh(at)opencanarias.com)

Summary:

Problem:

ObjectExp seems to be an enhancement to the InstantationExp due to the additional semantics, however it gets limited due to the fact that every time we need to use it we have to define the constructor body (in constrast to the instantiation expression usage). Although, ObjectExp was conceived to inline object instantiations, this limitation is not justified.

However, this limitation could be removed by also allowing an ObjectExp-Constructor combination, in the same way we have for InstantiationExp. The problem relies on a flaky concrete syntax which we could enhance to exploit an ObjectExp with an already defined constructor operation:

constructor Column::ColumnConstructor (n:String,t: String) { name:=n; type:=t; }

object result1 : Column (“name”, “String”);

object result2 : Column (“age”, “Integer”);

Providing a constructor body (from ObjectExp) and a list of arguments (from InstantiationExp) should be prohibited in both, the concrete syntax and the abstract syntax. Regarding the abstract syntax this could be expression with a constraint.

context ObjectExp

inv : argument->size() > 0 implies body.oclIsUndefined()

Discussion:

This enhancement seems convenient with no apparent drawbacks, since the old ObjectExp usage remains valid.

Proposed solution:

In section 8.2.1.24 add the following subsection:

Constraints

If an object expression contains a constructor body, no arguments for a constructor are allowed (and vice versa):

context ObjectExp

inv: argument->notEmpty() > 0 implies body.oclIsUndefined() and

not body.oclIsUndefined() implies argument->isEmpty()

In section 8.2.1.24 add the following to the the end notation subsection:

Similarly to InstantiationExp, an object expression could be used to invoke a constructor operation, rather than inlining a constructor body:

object result1 : Column (“name”, “String”);

object result2 : Column (“age”, “Integer”);

Note that this notation allows us to use object expression to instantiate (or update) objects, while having a reusable constructor in order to initialize (or update) the properties of the object subject to be created (or updated).

In section 8.4.7:

Replace:

<object_exp> ::= 'object' ('(' <iter_declarator> ')')? <object_declarator>

<expression_block>

By:

<object_exp> ::= 'object' ('(' <iter_declarator> ')')? <object_declarator>

(<expression_block> | '(' (<declarator_list>)? ')')

Resolution:

See also 19022. The InstantiationExp class seems to be poorly characterized causing difficulties for its derived classes. A clean up of this area needs to be prototyped before resolving this issue.

Disposition: **Deferred**