EsOCL

MANUAL FOR USE AND DEVELOPMENT

EsOCL has been developed at the Certus Software V&V Center, Simula Research Laboratory, Norway. EsOCL and this manual are available a the EsOCL project website (………………).

**CONTACT**

Shaukat Ali

[shaukat@simula.no](mailto:shaukat@simula.no)

# ABSTRACT

To efficiently generate test data from OCL constraints, we implemented the search-based approach described above. Figure 5 shows the architecture diagram for our search -based test data generator tool named as Evolutionary Solver for OCL (EsOCL). EsOCL is developed in Java that interacts with an existing library, an OCL evaluator, called the EyeOCL Software (EOS) [38]. EOS is a Java component that provides APIs to parse and evaluate an OCL expression based on an object model. Our tool only requires interacting with EOS for the evaluation of constraints. We use EOS as it is one the most efficient evaluators currently available; however any other OCL evaluator may be used. EsOCL implements the calculation of branch distance (DistanceCalculator) for various expressions in OCL as discussed in Section 4, which aims at calculating how far are the test data values from satisfying constraints. For a constraint, the search space is defined by those attributes that are used in the constraint. This is determined by statically parsing a constraint before solving it and improves the search efficiency in a similar fashion to the concept of input size reduction [67]. The search algorithms employed are implemented in Java as well and include Genetic Algorithm, (1+1) Evolutionary Algorithm, and Alternating Variable Method (AVM). Note that our implementation of branch distance calculation corresponds to OCL semantics as specified in [2].

CONTENTS

# About EsOCL

EsOCL is developed as a project at the Simula Research Laboratory. EsOCL consists of a set of modules, including OCL parser and interpreter, UML model instance generator, branch distance calculator and ………..

## Supported Version of OCL

EsOCL generally supports OCL 2.3 as speciﬁed in [OMG12]. The features of OCL2.3 that EsOCL supports are listed as follows:

1. Primitive types: Integer, Real, String, and Boolean
2. Enumeration type, Collection type
3. Boolean operation: not, and, or, implies, xor
4. Relation operation: =, <>, <, <=, >, >=
5. Checking operation: includes, excludes, includesAll, excludesAll, isEmpty, notEmpty
6. Iteration operation: forAll, exists, isUnique, one
7. Tuples
8. Miscellaneous operation: oclIsTypeOf, oclIsKindOf, oclIsNew, oclIsUndefined, oclIsInvalid
9. User-defined operation

## Supported Models and OCL Constraints

EsOCL can load UML class diagrams generated by the Eclipse Modeling Development Tools (Eclipse MDT) []. The models are formed as XMI ﬁles named as \*.uml. The OCL constraints named as \*.ocl are also imported by EsOCL.

# Getting Started with EsOCL

## Development Configuration

As a standalone application, EsOCL depends on the eclipse EMF, UML featurs (stored in the folder “lib”) and the Dresden OCL tool (stored in the folder “plugins”).

## Process of EsOCL

1. **Prepare the model and OCL file**

The UML model file is generated by the Eclipse MDT and the corresponding OCL file should be organized as follows:

package package\_1:: package\_2:: package\_3

OCL constraints

endpackage

The item “package” should be in consistent with the UML model.

1. **Initial the environment**

All the initial functions is stored in the

tudresden.ocl20.pivot.standalone.facade\_3.1.0.jar

provided by Dresden OCL. The initial process is written in the constructor method of class SolveProblem. First of all, the initialize method should be invoked with the configure file (log4j.properties). After parsing the OCL constraints, we should also initial the evaluation environment for OCL constraints by

new OclInterpreterPlugin().

1. **Load the model and parse the OCL constraints**

The loadUMLModel method of class StandaloneFacade is invoked for loading the UML model from .uml file. Then the OCL expressions in the .ocl file can be parsed as a list by the parseOclConstraints method.

1. **Generate the model instances and initial the attribute array**

We implemented this module in the processProblem of class SolveProblem. Based on the OCL constraint, we firstly collect the attribute (buildInitialVes in the class VesGenerator) involved in the expression as ValueElement4Search. Then the cardinality numbers will be confirmed. We generate the attribute array obeying with the cardinality numbers (the method getVes4InsNumberArray in the class UMLModelInsGenerator).

1. **Assign the values of diverse types into the instances**

In the search engine, it will generate the values into the different arrays of diverse types (the sequence is fixed by the order of attribute array in Step 4). Then we translate the value of each array element assigned into the attribute of model instance (the method getReAssignedUMLObjects of class UMLModelInsGenerator).

1. **Classify the OCL expressions to different types**

The OCL expressions can be handled by their operations. In the EsOCL, the types of expressions are listed in the Section 1.1 (the method classifyExp of class SolveProblem).

1. **Calculate the distance of OCL constraint**

The distance of OCL constraint can be calculated with model instances.

# API Information

## Class SolveProblem

### Attribute

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| model | IModel | The loaded model after reading the file |
| constraint | Constraint | The parsed OCL constraint |
| valueOfConstraints | ValueElement4Search[] | the attributes which should be delivered for searching process |
| initialVesForSearchList | List<ValueElement4Search> | the attributes involved in the constraint |
| vesGenerator | VESGenerator | The generator of attributes in the class |
| umlModelInsGenerator | UMLModelInsGenerator | The generator of class instances |
| I | int | The iteration times of search engine |

### Operation

#### Constructor

**Description:**

This operation is the constructor of this class. It is also used to initial the environment of loading models and OCL constraints, parsing and interpreting the constraints.

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| inputModelPath | String | The file path of model |
| inputOclConstraintsPath | String | The file path of OCL constraints |

#### processProblem

**Description:**

This operation is used to initial the attribute array. The array valuesOfConstraints is assigned with three types of value.

valuesOfConstraints[i][0]: min value of attribute

valuesOfConstraints[i][1]: max value of attribute

valuesOfConstraints[i][2]: type of attribute 0: Integer; 1: Boolean and Enumeration; 2: String; 3: Real

#### getFitness

**Description:**

This operation is used to calculate the distance of OCL constraint based on the given value array.

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| valueStr | String[] | The concrete value of each attribute |

#### classifyExp

**Description:**

This operation is used to identify the different kinds of expressions and calculate the distance

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| exp | Expression | The OCL constraint |
| imiObject | IModelInstanceObject | The related model instance |
| bdc | BDCManager | The manager for distance calculation which can store the OCL interpreter |

#### getAllAttributeConstraints

**Description:**

This operation is used to get all the attributes after confirming the number of objects

**Return:** ValueElement4Search[]

The ValueElement4Search array stores the attributes

#### getUMLResources

**Description:**

This operation is used to get the resource to resolve the type of UML model element

## Class UMLModelInsGenerator

### Attribute

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| attributeInsList | List<UMLAttributeIns> | this list is initialized for recording the attribute information after confirming the instance number |
| umlObjectInsList | List<UMLObjectIns> | this list contains the class instance |
| vesGenerator | VESGenerator | The generator of attributes in the class |

### Operation

#### getReAssignedUMLObjects

**Description:**

This operation is used to assign the concrete value into the *attributeInsList*.

**Return:** List<UMLObjectIns>

The class instances with assigned attributes

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| valueStr | String[] | The concrete value of each attribute |

#### getVes4InsNumberArray

**Description:**

This operation is used to generate the attribute array based on the number of class instances.

**Return:** ValueElement4Search[]

The attribute array

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| valueStr | String[] | The concrete value of each attribute |

#### buildUMLObjectFromVesList

**Description:**

This operation is used to generate the class instance from the attributes involved in the constraint

**Return:** UMLObjectIns

The class instance

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| initialVes4SameClassList | List<ValueElement4Search> | The attributes of the class |
| className | String | The class name |

#### getUMLObjects

**Description:**

This operation is used to find the list of *UMLObjectIns* from the *umlObjectInsList* based on the class name

**Return:** List<UMLObjectIns>

The list of class instance

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| className | String | The class name |

## Class VESGenerator

### Attribute

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| initialVesForSearchList | List<ValueElement4Search> | this list is initialized for recording the needed attribute information from .uml model number |
| iniVesGroupByClassMap | Map<String, List<ValueElement4Search>> | this map contains the attributes grouped by the class |
| enumerationList | List<UML2Enumeration> | The enumeration types in the .uml file |
| constraint | Constraint | The parsed OCL constraint |

### Operation

#### buildInitialVes

**Description:**

This operation is used to parse the attributes involved in the constraint and these attributes are not build with the number of class instance.

**Return:** List<ValueElement4Search>

The ValueElement4Search stores the necessary information about attribute.

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| model | IModel | The loaded model after reading the file |

#### value4PPType

**Description:**

This operation is used to obtain the int label of each type.

**Return:** int

The int label of each type

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| type | org.eclipse.uml2.uml.Type | The type of attribute |

## Class OCLExpUtility

### Attribute

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| OP\_COMPLEX\_SELECT\_SIZE | String | The complex operation like “select->size” |
| OP\_COMPLEX\_SELECT\_ITERATE | String | The complex operation like “select->forAll” |
| OP\_BOOLEAN | String[] | The Boolean operations |
| OP\_COMPARE | String[] | The Relation operations |
| OP\_ITERATE | String[] | The Iteration operations |
| OP\_CHECK | String[] | The Check operations |
| OP\_SELECT | String[] | The Select operations |
| OP\_MISCELLANEOUS | String[] | The Miscellaneous operations |
| OP\_BOUND | String[] | Operations use in the bound value strategy |
| vesGenerator | VESGenerator | The generator of attributes in the class |
| typeArray | String[][] | The bound value type of comparison expression |
| comb | int[][] | The composition array of bound value |
| boundIndex | int | The indicator for bound value strategy |
| oceMap | Map<OperationCallExpImpl, Integer> | This map stores the comparison expression with its right original value |

### Operation

#### getResultCollection

**Description:**

This operation is used to obtain the interpretation result as a collection .

**Return:** Collection<IModelInstanceElement>

A collection with the type of IModelInstanceElement in the interpretation result of OCL expression

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| oclAny | OclAny | the interpretation result of OCL expression |

#### getResultNumericValue

**Description:**

This operation is used to obtain the numeric values for different IModelInstanceElements.

**Return:** double

The numeric values for different types of IModelInstanceElements

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| imiElement | IModelInstanceElement | The IModelInstanceElement in the interpretation result of OCL expression |

#### getOppositeOp

**Description:**

This operation is used to obtain the opposite operation corresponding with the given operation.

**Return:** String

The name of the opposite operation

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| opName | String | the given operation |

#### isComplexType

**Description:**

This operation is used to determine whether the OCL expression belongs to the complex operation

**Return:** String

The name of complex operation

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| exp | OclExpression | the given OCL expression |

#### isBelongToOp

**Description:**

This operation is used to determine whether the given operation belongs to the given type of operation

**Return:** Boolean

Whether the given operation belongs to the given type of operaiton

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| opName | String | the given operation |
| ops | String[] | the given types of operation |

#### getASC4String

**Description:**

This operation is used to obtain the ASCII of each character in the given string.

**Return:** int[]

Each Integer value in this array is the ASCII of the character in the given string.

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| str | String | the given string |

#### printOclClause4Depth

**Description:**

This operation is used to represent the tree structure of OCL expression

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| e | EObject | the root element of OCL expression |

#### getPropertyInCons

**Description:**

This operation is used to identify the attribute (PropertyCallExpImpl) involved in the constraint.

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| constraint | Constraint | The parsed OCL constraint |

#### buildIndexArray4Bound

**Description:**

This operation is used to identify the comparison expression for bound value strategy and generate the composition of bound value for each expression.

**Return:** int

This number will indicate the execution times of search process.

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| constraint | Constraint | The parsed OCL constraint |

#### generateBoundValue

**Description:**

This operation is used to modify the right value of comparison expression. The value is generated based on the *comb* array. comb[boundIndex][i] will provide the index for confirming the bound value of right part. (0: right-1; 1: right; 2: right+1)

#### restoreOriginalValue

**Description:**

This operation is used to restore the original value of each comparison expression for generating the bound value.

#### buildBoundTypeArray

**Description:**

This operation is used to generate the type array of comparison expressions. The type is corresponding to the bound value:

typeArray[i][0]-----value-1

typeArray[i][1]-----value

typeArray[i][2]----- value+1

**Return:** String[][]

This type array.

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| oceSet | Set<OperationCallExpImpl> | The expression set |

## Class Utility

### Attribute

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| K | double | The K value mentioned in the paper |
| modelDoc | Document | The document built by DOM4j |

### Operation

#### formatValue

**Description:**

This operation is used to format the double value.

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| value | double | The value needed to format |

#### formatRealValueWithoutZero

**Description:**

This operation is used to format the right decimal part of real value.

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| afterDecimal | String | The value needed to format |

#### getLowAndUpperValueForProperty

**Description:**

This operation is used to get the min and max value of attribute.

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| className | String | The class which the attribute belongs to |
| attrName | String | The attribute name |

#### getElementID

**Description:**

This operation is used to get the id of model element in the .uml file.

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| fatherElementName | String | The father element of specified element |
| elementName | String | The element name |

#### getFixedNumberOfCardinality

**Description:**

This operation is used to confirm the value of cardinality.

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| assVes | ValueElement4Search | The attribute element |

## Class BDC4CompareOp

### Attribute

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| utility | Utility | The tool class for simple functions |
| interpreter | OclInterpreter | The interpreter for interpreting the OCL expression |
| oclExpUtility | OCLExpUtility | The OCL tool class |

### Operation

#### handleCompareOp

**Description:**

This operation is used to calculate the distance of relation expression.

**Return:** double

The distance of relation expression

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| env | IModelInstanceObject | the environment variable of OCL expression like “self” |
| leftExp | OclExpression | The left expression of operator |
| rightExp | OclExpression | The right expression of operator |

#### handleCollectionEquality

**Description:**

This operation is used to calculate the distance of collection equality.

**Return:** double

The distance of relation expression

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| env | IModelInstanceObject | the environment variable of OCL expression like “self” |
| left | OclAny | The left expression of operator |
| right | OclAny | The right expression of operator |
| opName | String | The operator name |

#### handleComplexSelectSizeOp

**Description:**

This operation is used to calculate the distance for Select() followed by Size().

**Return:** double

The distance of relation expression

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| env | IModelInstanceObject | the environment variable of OCL expression like “self” |
| sizeExp | OperationCallExpImpl | The left expression of relation operator |
| leftResult | double | The value of left expression interpreted by  OCL interpreter |
| rightResult | double | The value of right expression interpreted by OCL interpreter |
| opName | String | The operator name |

#### compareOp4Numeric

**Description:**

This operation is used to calculate the distance for numeric relation.

**Return:** double

The distance of relation expression

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| diversity | double | The diversity value bwteen left and right value |
| opName | String | The operator name |

## Class BDC4IterateOp

### Attribute

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| utility | Utility | The tool class for simple functions |
| interpreter | OclInterpreter | The interpreter for interpreting the OCL expression |
| oclExpUtility | OCLExpUtility | The OCL tool class |

### Operation

#### handleIteratorOp

**Description:**

This operation is used to calculate the distance of iteration expression. It is used to distinguish the type of iteration expression.

**Return:** double

The distance of relation expression

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| env | IModelInstanceObject | the environment variable of OCL expression like “self” |
| iteratorExp | IteratorExpImpl | The iteration expression |

#### handleComplexSelectIterateOp

**Description:**

This operation is used to calculate the distance for Select() followed by forALL and exists.

**Return:** double

The distance of relation expression

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| env | IModelInstanceObject | the environment variable of OCL expression like “self” |
| iteratorExp | IteratorExpImpl | The iteration expression |

#### forAllOp

**Description:**

This operation is used to calculate the distance for “forAll” expression.

**Return:** double

The distance of relation expression

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| env | IModelInstanceObject | The environment variable of OCL expression like “self” |
| envArray | IModelInstanceElement[] | The environment variables for “forAll” expression like “self.C1” |
| forAllIterators | List<Variable> | The iterator variables for “forAll” expression |
| selectIterator | Variable | The iterator variables for “select” expression |
| forAllParaExp | OclExpression | The Boolean expression in “forAll” expression |
| selectParaExp | OclExpression | If this expression belongs to the complex type, the p2 is the Boolean expression in “select” expression |

#### existsOp

**Description:**

This operation is used to calculate the distance for “exists” expression.

**Return:** double

The distance of relation expression

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| env | IModelInstanceObject | The environment variable of OCL expression like “self” |
| envArray | IModelInstanceElement[] | The environment variables for “exists” expression like “self.C1” |
| existsIterators | List<Variable> | The iterator variables for “exists” expression |
| selectIterator | Variable | The iterator variables for “select” expression |
| existsParaExp | OclExpression | The Boolean expression in “exists” expression |
| selectParaExp | OclExpression | If this expression belongs to the complex type, the p2 is the Boolean expression in “select” expression |

#### isUniqueOp

**Description:**

This operation is used to calculate the distance for “isUnique” expression.

**Return:** double

The distance of relation expression

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| envArray | IModelInstanceElement[] | The environment variables for “exists” expression like “self.C1” |
| uniqueIterators | List<Variable> | The iterator variables for “isUnique” expression |
| uniqueParaExp | OclExpression | The Boolean expression in “isUnique” expression |

#### oneOp

**Description:**

This operation is used to calculate the distance for “not” expression.

**Return:** double

The distance of relation expression

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| env | IModelInstanceObject | The environment variable of OCL expression like “self” |
| envArray | IModelInstanceElement[] | The environment variables for “one” expression like “self.C1” |
| oneIterators | List<Variable> | The iterator variables for “one” expression |
| selectIterator | Variable | The iterator variables for “select” expression |
| oneParaExp | OclExpression | The Boolean expression in “one” expression |
| selectParaExp | OclExpression | If this expression belongs to the complex type, the p2 is the Boolean expression in “select” expression |

## Class BDC4BooleanOp

### Attribute

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| utility | Utility | The tool class for simple functions |
| interpreter | OclInterpreter | The interpreter for interpreting the OCL expression |
| oclExpUtility | OCLExpUtility | The OCL tool class |
| numOfUndClauses | int | The number of undefined clauses in the OCL expression |

### Operation

#### handleBooleanOp

**Description:**

This operation is used to calculate the distance of Boolean expression. It is used to distinguish the type of Boolean expression.

**Return:** double

The distance of relation expression

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| env | IModelInstanceObject | the environment variable of OCL expression like “self” |
| exp | OclExpression | The Boolean expression |

#### classifyValue

**Description:**

This operation is used to classify the expression into the Relation or Boolean type

**Return:** double

The distance of relation expression

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| env | IModelInstanceObject | the environment variable of OCL expression like “self” |
| leftExp | OclExpression | The left expression of operator |
| rightExp | OclExpression | The right expression of operator |
| opName | String | The name of operator |

#### simplePropOrMiscOp

**Description:**

This operation is used to calculate the distance for miscellaneous operations.

**Return:** double

The distance of relation expression

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| env | IModelInstanceObject | the environment variable of OCL expression like “self” |
| exp | OclExpression | The OCL expression for miscellaneous operation |

#### notOp

**Description:**

This operation is used to calculate the distance for “not” expression.

**Return:** double

The distance of relation expression

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| env | IModelInstanceObject | the environment variable of OCL expression like “self” |
| exp | OclExpression | The OCL expression for “not” operation |

#### andOp

**Description:**

This operation is used to calculate the distance for “and” expression.

**Return:** double

The distance of relation expression

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| env | IModelInstanceObject | the environment variable of OCL expression like “self” |
| exp | OclExpression | The OCL expression for “and” operation |

#### orOp

**Description:**

This operation is used to calculate the distance for “or” expression.

**Return:** double

The distance of relation expression

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| env | IModelInstanceObject | the environment variable of OCL expression like “self” |
| exp | OclExpression | The OCL expression for “or” operation |

#### impliesOp

**Description:**

This operation is used to calculate the distance for “implies” expression.

**Return:** double

The distance of relation expression

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| env | IModelInstanceObject | the environment variable of OCL expression like “self” |
| exp | OclExpression | The OCL expression for “implies” operation |

#### xorOp

**Description:**

This operation is used to calculate the distance for “xor” expression.

**Return:** double

The distance of relation expression

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| env | IModelInstanceObject | the environment variable of OCL expression like “self” |
| exp | OclExpression | The OCL expression for “xor” operation |

## Class BDC4CheckOp

### Attribute

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| utility | Utility | The tool class for simple functions |
| interpreter | OclInterpreter | The interpreter for interpreting the OCL expression |
| oclExpUtility | OCLExpUtility | The OCL tool class |

### Operation

#### handleCheckOp

**Description:**

This operation is used to calculate the distance of check expression. It is used to distinguish the type of check expression.

**Return:** double

The distance of relation expression

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| env | IModelInstanceObject | the environment variable of OCL expression like “self” |
| opCallexp | OperationCallExpImpl | The check expression |

## Class ModelInsFileWriter

### Attribute

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| doc | Document | The document to store the instances |
| rootElement | Element | The root element of XML structure |
| ins\_index | int | The index of class instance |
| slot\_index | int | The index of slot |
| value\_index | int | The index of value |
| spec\_index | int | The index of specification |
| umlObjectInsList | List<UMLObjectIns> | The list of objects to be generated |
| insMap | Map<UMLObjectIns, String> | The map for storing the index for each class instance |

### Operation

#### Constructor

**Description:**

This operation is used to initial the *umlObjectInsList* and create the document.

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| umlObjectInsList | List<UMLObjectIns> | The list of objects to be generated |

#### createInstanceSpecification

**Description:**

This operation is used to create the instance.

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| instanceName | String | The instance name |
| instanceID | String | The instance id |
| classifiedID | String | The type id of instance |

#### createSlot

**Description:**

This operation is used to create the slot.

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| fatherElement | Element | The father element of slot |
| slotID | String | The slot id |
| definingFeatureID | String | The definingFeature id |
| valueID | String | The id of slot value |
| value | String | The slot value |
| instanceID | String | The instance id |
| type | String | The slot type |

#### createInstanceSpecification

**Description:**

This operation is used to create the specification.

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| fatherElement | Element | The father element of specification |
| specID | String | The specification id |
| value | String | The specification value |

#### createSlotFromUMLAttr

**Description:**

This operation is used to create the slot from the class attribute.

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| fatherElement | Element | The father element of slot |
| attr | Object | The class attribute |

#### writeToFile

**Description:**

This operation is used to write the document into the .uml file.

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| filePath | String | The path of .uml file |

## Class Combination

### Operation

#### mn

**Description:**

This operation is used to get the combination for a number array.

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| array | String[] | The number array as a string array |
| n | int | The needed combination number |

## Class Arrange

### Operation

#### perm

**Description:**

This operation is used to get the arrangement for a number array.

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| list | String[] | The number array as a string array |
| k | int | The start index of arrangement |
| m | int | The end index of arrangement |

## Class RmodelIns

### Attribute

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| nameCounter | int | It is used to calculate the number of instances |

### Operation

#### Constructor

**Description:**

This operation is the constructor of this class. It is also used to initial the model instance factory and add the model instance.

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| model | IModel | The model parsed from the .uml file |
| umis | List<UMLObjectIns> | The model instance generated from IModel |

#### addModelInstanceElement

**Description:**

This operation is used to build the IModelInstanceElement from the List<UMLObjectIns>. If it is the class object, it will be added into the ModelInstanceObject list.

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| object | Object | The object will be built as IModelInstanceElement |

## Class RModelInsFactory

### Attribute

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| model | IModel | The model parsed from the .uml file |
| cacheModelInstanceObjects | Map<Object, IModelInstanceObject> | The map storing the different types of |

### Operation

#### createModelInstanceElement

**Description:**

This operation is used to build the IModelInstanceElement based on the type of adapted. It will invoke the corresponding methods as followings:

createModelInstanceBoolean(UMLAttributeIns, Type)

createModelInstanceEnumerationLiteral(UMLAttributeIns, Enumeration)

createModelInstanceInteger(UMLAttributeIns, Type)

createModelInstanceReal(UMLAttributeIns, Type)

createModelInstanceString(UMLAttributeIns, Type)

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| adapted | Object | The object will be built as IModelInstanceElement |

## Class RModelInsObject

### Attribute

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| aumi | AbstUMLModelIns | It is used to build the RModelInsObject |
| modelInstanceFactory | IModelInstanceFactory | It is used to build the ModelInstanceElement based its type |

### Operation

#### asType

**Description:**

This operation is used to check whether the type can be casted in the model. If it can be done, the operation return the IModelInstanceElement with this type

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| type | Type | The given type which may be casted |

#### getProperty

**Description:**

This operation is used to obtain the property IModelInstanceElement based on the given property.

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| property | Property | The given property will be found as IModelInstanceElement |

## Class AbstUMLModelIns

### Attribute

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| name | String | The name of element |
| value | String | The value of element |

## UMLAttributeIns

This class just extends the AbstUMLModelIns to represent the attribute instance.

## UMLObjectIns

### Attribute

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| propertyMap | Map<String, Object> | The map for storing the attribute instances with their name |

### Operation

#### getPrimitivePropertyCollection

**Description:**

This operation is used to build the collection of primitive attributes

**Return:** Collection<UMLAttributeIns>

The collection of primitive attributes