Zen-Configurator

MANUAL FOR USE AND DEVELOPMENT

Zen-Configurator has been developed at the Certus Software V&V Center, Simula Research Laboratory, Norway.

**CONTACT**

Tao Yue: [tao@simula.no](mailto:tao@simula.no)

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

CONTENTS

# Zen-CC API Information

## Class ConsistencyChecking

### Attibute

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| model | IModel | The loaded model after reading the file |
| rulemag | RuleManager | The Singleton instance of rule manager |
| vpmag | VPManager | The Singleton instance of vp manager |
| searchflag | Boolean | to indicate whether it's in a search process |
| fixflag | boolean | used in the zen-fix experiment |

### Operation

#### InitialAnalysis

|  |  |  |
| --- | --- | --- |
| Description | | |
| It is used to initial the environment of loading models and parsing vps from the model, as well as parsing and interpreting the constraints. | | |
| Parameter Name | Type | Parameter Description |
| metamodelpath | String | The file path of model |
| oclpath | String | The file path of OCL constraints |

#### checkConsistency

|  |  |  |
| --- | --- | --- |
| Description | | |
| This operation is the interface for Zen-CC to integrate with Zen-Configurator. It’s used to update the dynamic validation trees. | | |
| Parameter Name | Type | Parameter Description |
| vp | VariationPointCC | the currently configured vp |
| instanceID | String | instance id for currently configured vp |
| value | String | configuration data for currently configured vp |

## Class VPManager

### Attibute

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| vpmap | Map<String, VPScope> | mapping between vp name and vp scope (mapping betweed vp id and related rule or rule nodes) |
| iD\_Element | HashMap<String,Element> | mapping between xml node id and xml element |
| vpName\_Instance | HashMap<String,String> | mapping between vp name and vp id |
| vpattr\_Context | HashMap<String,String> | mapping between cardinality vp and related class |
| father\_child | HashMap<String, ArrayList<String>> | used to store class generalizations of the model |
| allconfig | HashMap<String, VariationPoint> | configured variation point and corresponding id |
| vpintanceList | HashMap<String,String> | configured vp id and configuration data |
| vpinferedList | HashMap<String, String> | inferred vp id and inferred configuration data |
| vpConFree | int | to record the number of vps related with no constraint |

### Operation

#### initialIDEleMap

|  |  |  |
| --- | --- | --- |
| Description | | |
| It is used to iteratively parse the xml model file and build the mapping between xml element id and xml element (iD\_Element). Meanwhile it also captures the class generalization relationships (father\_child). | | |
| Parameter Name | Type | Parameter Description |
| e | Element | the root element of the xml model file |

#### initialCUMap

|  |  |  |
| --- | --- | --- |
| Description | | |
| It is used to iteratively parse the xml model file and build the mapping between the vp and corresponding initial instance (idvpName\_Instance); meanwhile it also captures the composition relationships and build the mapping between cardinality vp and related class (vpattr\_Context) | | |
| Parameter Name | Type | Parameter Description |
| rootClass | Element | the root class element of the xml model file |
| prefix | String | instance id for the root class |

#### (un)registerVP

|  |  |  |
| --- | --- | --- |
| Description | | |
| It is used to register or unregister an observer (rule or rule node) for a specific id of a vp | | |
| Parameter Name | Type | Parameter Description |
| vpname | String | the name of the vp to be registered |
| instanceID | String | the instance id of the vp instance to be registered |
| obs | Observer | the observer (rule node) to be registered |

#### getVPScope

|  |  |  |
| --- | --- | --- |
| Description | | |
| It is used to return the scope of a vp, i.e., mapping between vp id and a set of related rule or rule nodes | | |
| Parameter Name | Type | Parameter Description |
| vpname | String | the name of the vp |

#### getVPInstanceID

|  |  |  |
| --- | --- | --- |
| Description | | |
| It is used to get the initial instance id for a vp | | |
| Parameter Name | Type | Parameter Description |
| vpname | String | the name of the vp |

## Class VPScope

### Attibute

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| vptonode | HashMap<String, ArrayList<Observer>> | mapping between vp instance id and a set of related observers (rule nodes or rules) |
| validnum | long | the number of constraints evaluated to be true |
| invalidnum | long | the number of constraints evaluated to be false |
| undefinednum | long | the number of constraints evaluated to be undefined |

### Operation

#### getScopeMap

|  |  |  |
| --- | --- | --- |
| Description | | |
| Return a list of observers for this vp instance id, i.e., a set of rules or rule nodes | | |
| Parameter Name | Type | Parameter Description |
| instanceid | String | instance id of a vp instance |

#### setScopeMap

|  |  |  |
| --- | --- | --- |
| Description | | |
| set a list of observers (rules or rule nodes) for a vp instance with instanceid | | |
| Parameter Name | Type | Parameter Description |
| instanceid | String | instance id of a vp instance |
| list | ArrayList<Observer> | the list of observers to be added for this instanceid |

#### addObserver/removeObserver

|  |  |  |
| --- | --- | --- |
| Description | | |
| Add or remove an observer for a vp instance with id being instanceid | | |
| Parameter Name | Type | Parameter Description |
| instanceid | String | instance id of a vp instance |
| o | Observer | the observer to be added or removed |

#### notifyObservers

|  |  |  |
| --- | --- | --- |
| Description | | |
| This is invoked in ConsistencyChecking::checkConsistency, and is used to notify and update all the observers registered for currently configured vp instance | | |
| Parameter Name | Type | Parameter Description |
| arg | Object | this parameter capsulate the type of vp scope (either rule or rule node), configured instance id and configuration data for this instance with the following two forms:  "RULETYPE"+"\*"+instanceID+"\*"+value  or  "NODETYPE"+"\*" + instanceID+"\*"+value |

## Class RuleManager

### Attibute

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| rulenum | int | total number of trees/rules |
| rulemap | Map<String, Rule> | mapping between rule id and the rule |
| rmanager | RuleManager | The Singleton instance of rule manager |
| model | IModel | model for the product line, used to parse the constraints |
| vpcount | int | total number of vps related with all the trees |
| patterncount | int | the total number of static inference patterns of all the trees |

### Operation

#### InitialAllTress

|  |  |  |
| --- | --- | --- |
| Description | | |
| It is used to parse each OCL constraint in the constraint file and transform it into an initial dynamic validation tree | | |
| Parameter Name | Type | Parameter Description |
| model | IModel | product line model (\*.uml) |
| oclpath | String | The file path of OCL constraints |

#### getAllnodeNum

|  |
| --- |
| Description |
| Return the total number of rule nodes in the dynamic validation forest |

## Class Rule

### Attibute

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| constraint | Constraint | corresponding OCL constraint |
| context | String | qualified name of the context of the rule |
| shortContext | String | short name of the context of the rule |
| ruleID | Boolean | to indicate whether it's in a search process |
| fixflag | boolean | used in the zen-fix experiment |

### Operation

#### ConstructTree

|  |  |  |
| --- | --- | --- |
| Description | | |
| It’s used to parse the ocl constraint and re-construct a tree where the leaf node is atomic proposition with scope. The scope for iterator node is the cardinality variation point, the scope for root node is the cardinality/type vp related with the context and the scope for compound node is null. | | |
| Parameter Name | Type | Parameter Description |
| constraitn | Constraint | The OCL constraint to be transformed into a tree |

#### getChildNodes

|  |  |  |
| --- | --- | --- |
| Description | | |
| Iteratively construct child nodes | | |
| Parameter Name | Type | Parameter Description |
| root | RuleNode | the root node of the tree |
| parent | RuleNode | the parent of those child nodes |
| consNodes | EList<Object> | a list of OCL objects corresponding to the children |
| parantNodeID | String | the node id of the parent |

#### getRuleNode

|  |  |  |
| --- | --- | --- |
| Description | | |
| Build a rule node based on the type of rule nodes; i.e., BooleanNode, CompundNode, IteratorNode | | |
| Parameter Name | Type | Parameter Description |
| consNode | EObject | the OCL object to be constructed to a node |
| parantNodeID | String | the node id of the parent |
| parent | RuleNode | the parent of the node to be constructed |
| root | RuleNode | the root node of the corresponding tree |

#### DoBranching

|  |  |  |
| --- | --- | --- |
| Description | | |
| Branch the trees according to the value of currently configured cardinality vp | | |
| Parameter Name | Type | Parameter Description |
| rulenode | RuleNode | the iterator rule node related with currently configured cardinality vp |
| num | int | the configuration data for this cardinality vp |

#### DoPrunning

|  |  |  |
| --- | --- | --- |
| Description | | |
| Prune the trees according to the value of currently configured cardinality vp | | |
| Parameter Name | Type | Parameter Description |
| rulenode | RuleNode | the iterator rule node related with currently re-configured cardinality vp |
| num | int | the configuration data for this cardinality vp |

#### DoPropagate

|  |  |  |
| --- | --- | --- |
| Description | | |
| propagate the evaluation values from one BooleanNode/leaf node upwards to the root node | | |
| Parameter Name | Type | Parameter Description |
| rulenode | RuleNode | the leaf rule node related with currently re-configured vp, and propagate from this node |

#### registerRule

|  |
| --- |
| Description |
| Register this rule to VPManager according the context of this rule, which maintains a mapping relationship between the vp (related with context, cardinality or type) and the rule. This operation is invoked during constructing the tree. |

#### update

|  |  |  |
| --- | --- | --- |
| Description | | |
| A rule is updated when configuring a vp related with the context of a rule. For example, if a cardinality vp is related with context of a rule then when this vp is configured to, for example 2, the corresponding tree should be updated into two trees. This operation is invoked by VPScope::notifyObservers | | |
| Parameter Name | Type | Parameter Description |
| arg0 | Observable | observable vp scope related with currently configured vp instance |
| arg1 | Object | this parameter capsulate the type of vp scope (rule), configured instance id and configuration data for this instance with the following form:  "RULETYPE"+"\*"+instanceID+"\*"+value |

## Class RuleNode

The abstract class for different types of rule ndoes.

### Attibute

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| parentId | String | parent node id of this rule node |
| parent | RuleNode | parent node of this node |
| childList | ArrayList<RuleNode> | child nodes of this node |
| nodeID | String | unique id for this node |
| ruleID | String | unique id for the rule containing this node |
| root | RuleNode | the root of the rule containing this node |
| value | Boolean | evaluation value of this node; null by default (undefined) |
| configured | boolean | flag to indicate whether this node is configured |
| constraintEle | OclExpression | ocl expression related with this rule node |
| booleannum | int | total number of Boolean nodes |
| logicnum | int | total number of Compound nodes |
| iteratornum | int | total number of Iterator nodes |
| distance | double | the distance of the configuration data from satisfying this node, 0 for the root node whose evaluation value is true or undefined; > 0 for the root node whose evaluation value is false |
| inferValue | Boolean | the static inference flag |

### Operation

#### copyTo

|  |  |  |
| --- | --- | --- |
| Description | | |
| Copy current rule node to a new rule node and copy all children of current node to the children of the new rule node | | |
| Parameter Name | Type | Parameter Description |
| newNode | RuleNode | the new rule node after copying the current node |

#### deleteChildNode

|  |  |  |
| --- | --- | --- |
| Description | | |
| Delete the child with certain node id of this node | | |
| Parameter Name | Type | Parameter Description |
| childId | String | id of the child node to be deleted |

#### deleteNode

|  |
| --- |
| Description |
| Delete this node and all its child nodes |

#### getCanInferedVPsNum

|  |  |  |
| --- | --- | --- |
| Description | | |
| Return the number of inferred vp from this node and all its children | | |
| Parameter Name | Type | Parameter Description |
| desiredValue | boolean | the desired value of this node |

#### getNodeNum

|  |
| --- |
| Description |
| Return a 3-length array correspondingly indicating the number of Boolean nodes, compound nodes and iterator nodes from this node and all its children. |

#### isReady

|  |  |  |
| --- | --- | --- |
| Description | | |
| The abstract method to be implemented by all its sub classes. To checked whether a node is ready to be update | | |
| Parameter Name | Type | Parameter Description |
| arg0 | String | the parameter encapsulating currently configured vp |

#### register

|  |  |  |
| --- | --- | --- |
| Description | | |
| The abstract method to be implemented by all its sub classes. Register this node with the currently configured vp by id | | |
| Parameter Name | Type | Parameter Description |
| instanceID | String | id of the currently configured vp |

#### update

|  |  |  |
| --- | --- | --- |
| Description | | |
| The abstract method to be implemented by all its sub classes. Update this node according to the currently configured vp | | |
| Parameter Name | Type | Parameter Description |
| arg0 | Observable | the object being observed by this node |
| arg1 | Object | this parameter capsulate the type of vp scope (either rule or rule node), configured instance id and configuration data for this instance with the following two forms:  "RULETYPE"+"\*"+instanceID+"\*"+value  or  "NODETYPE"+"\*" + instanceID+"\*"+value |

## Class BooleanNode

### Attibute

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| vpInstanceIDList | ArrayList<String> | a list of vp instance id related with this Boolean node, i.e., scope of this node |
| configedVPList | ArrayList<String> | a list of configured vp instance id related with this Boolean node |
| iterators | List<Variable> | iterators if this Boolean node is a child of an iterator node |
| propertyList | ArrayList  <PropertyCallExpImpl> | a list of UML properties related with this Boolean node |
| id\_name | HashMap<String,String> | mapping between vp instance id and vp name |
| classid\_Object | HashMap  <String,UMLObjectIns> | mapping between class instance id and corresponding uml class |
| attrid\_Object | HashMap  <String,UMLAttributeIns> | mapping between attribute instance id and corresponding uml attributes |
| vpType | VariationPType | indicating the vp type of this node; either *Type*, *Cardinality* or *Attribute* |
| inferedvalue | String | the value that can be inferred |
| left | int | the value in the left of the operator; only used for cardinality vp |
| right | int | the value in the right of the operator; only used for cardinality vp |
| processed | boolean | flag to indicate whether this node is processed |
| isRegistered | boolean | flag to indicate whether this node is registered |

### Operation

#### register

|  |  |  |
| --- | --- | --- |
| Description | | |
| Register this node with a vp to be configured by vp’s name and vp’s instance id. . Implement the abstract method of RuleNode::register | | |
| Parameter Name | Type | Parameter Description |
| instanceID | String | id of the currently configured vp instance |

#### update

|  |  |  |
| --- | --- | --- |
| Description | | |
| Update this boolean node; i.e., evaluate this Boolean node to be *true* or *false*. Implement the abstract method of RuleNode::update | | |
| Parameter Name | Type | Parameter Description |
| arg0 | Observable | the object being observed by this node |
| arg1 | Object | this parameter capsulate the type of vp scope (rule node), configured instance id and configuration data for this instance with the following form:  "NODETYPE"+"\*" + instanceID+"\*"+value |

#### buildInstance

|  |
| --- |
| Description |
| It’s used to build partial product model (object model) related with this Boolean node based on the configuration data. |

#### calculateScope

|  |
| --- |
| Description |
| It is used to parse this Boolean node and capture different types of vps related with this Boolean node; meanwhile it also identify the static inference flag which is used to enable inference (Zen-CI) |

#### calculateDistance

|  |  |  |
| --- | --- | --- |
| Description | | |
| It is used to compute the branch distance, i.e., the distance of a set of configuration data from satisfying this Boolean node. Return a double-typed value. | | |
| Parameter Name | Type | Parameter Description |
| left | OclAny | evaluation result for the left part of the Boolean operation corresponding to this Boolean node |
| right | OclAny | evaluation result for the right part of the Boolean operation corresponding to this Boolean node |

#### evaluate

|  |
| --- |
| Description |
| Only used when the vp related with this Boolean node is of Cardinality type. To evaluate the value of this Boolean node, either *true* or *false*. |

#### needRefresh

|  |  |  |
| --- | --- | --- |
| Description | | |
| Check whether the id for a vp instance need to be refreshed. The instance id of a vp is dynamically updated during configuration based on the id of its composing vp. | | |
| Parameter Name | Type | Parameter Description |
| vpID | String | current id for one vp instance related with this Boolean node |

#### needRegister

|  |  |  |
| --- | --- | --- |
| Description | | |
| Check whether the id for one vp instance is complete and hence ready to register itself with this Boolean node, such that this node could be updated when this vp instance is configured later. | | |
| Parameter Name | Type | Parameter Description |
| vpID | String | current id for one vp instance related with this Boolean node |

#### propagate

|  |  |  |
| --- | --- | --- |
| Description | | |
| Propagate the evaluation value of this Boolean node to its parents. Implement the abstract method of RuleNode::propagate | | |
| Parameter Name | Type | Parameter Description |
| child | RuleNode | child node of this Boolean node, should be null. |

#### refreshInstanceID

|  |  |  |
| --- | --- | --- |
| Description | | |
| The id of a vp instance is dynamically updated during configuration based on the id of its composing vp, i.e., currently configured vp instance. | | |
| Parameter Name | Type | Parameter Description |
| instanceID | String | id of currently configured vp instance |

#### setEnviroment

|  |  |  |
| --- | --- | --- |
| Description | | |
| Set the evaluation environment. For example set the object instance for self, or other iterator variables | | |
| Parameter Name | Type | Parameter Description |
| interpreter | OclInterpreter | ocl interpreter from Dresden OCL |
| instance | IModelInstance | model instance to be evaluted |

#### isReady

|  |  |  |
| --- | --- | --- |
| Description | | |
| Check whether a Boolean node is ready to be checked/evaluated. Implement the abstract method of RuleNode::isReady | | |
| Parameter Name | Type | Parameter Description |
| ID\_value | String | the information (id and configuration) about currently configured vp instance, with the form of “instanceID+"\*"+value” |

## Class IteratorNode

### Attibute

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| vpName | String | Name of the vp associated with current iterator node |
| vpInstanceID | String | Instance id of the vp associated with current iterator node |
| vpQName | String | Qualified name of the vp associated with current iterator node |
| vpClass | String | Corresponding class name relevant with this vp |
| cardNum | int | Current number of branches for this iterator node |
| isConfigured | boolean | A flag indicating whether this vp is configured or not |

### Operation

#### calculateScope

|  |
| --- |
| Description |
| It is used to parse this Iterator node and capture cardinality vp related with this Iterator node; |

#### register

|  |  |  |
| --- | --- | --- |
| Description | | |
| Register this Iterator node with a vp to be configured by vp’s name and vp’s instance id. . Implement the abstract method of RuleNode::register | | |
| Parameter Name | Type | Parameter Description |
| instanceID | String | id of the currently configured vp instance |

#### propagate

|  |  |  |
| --- | --- | --- |
| Description | | |
| Propagate the evaluation value of this Iterator node to its parents. Implement the abstract method of RuleNode::propagate | | |
| Parameter Name | Type | Parameter Description |
| child | RuleNode | child node of this Iterator node |

#### refreshInstanceID

|  |  |  |
| --- | --- | --- |
| Description | | |
| The id of a vp instance is dynamically updated during configuration based on the id of its composing vp, i.e., currently configured vp instance. | | |
| Parameter Name | Type | Parameter Description |
| instanceID | String | id of currently configured vp instance |

#### calculateDistance

|  |  |  |
| --- | --- | --- |
| Description | | |
| It is used to compute the branch distance, i.e., the distance of a set of configuration data from satisfying this iterator node. Return a double-typed value. | | |
| Parameter Name | Type | Parameter Description |
| ope | String | The operation name of this iterator node, such as forAll, exists, one, isUnique |

#### update

|  |  |  |
| --- | --- | --- |
| Description | | |
| Update this Iterator node; i.e., branching or pruning. Implement the abstract method of RuleNode::update | | |
| Parameter Name | Type | Parameter Description |
| arg0 | Observable | the object being observed by this node |
| arg1 | Object | this parameter capsulate the type of vp scope (rule node), configured instance id and configuration data for this instance with the following form:  "NODETYPE"+"\*" + instanceID+"\*"+value |

## Class CompoundNode

### Operation

#### register

|  |  |  |
| --- | --- | --- |
| Description | | |
| The compound node itself won’t be registered. Register all the children of this compound node. Implement the abstract method of RuleNode::register | | |
| Parameter Name | Type | Parameter Description |
| instanceID | String | id of the currently configured vp instance |

#### propagate

|  |  |  |
| --- | --- | --- |
| Description | | |
| Propagate the evaluation value of this compound node to its parents. Implement the abstract method of RuleNode::propagate | | |
| Parameter Name | Type | Parameter Description |
| child | RuleNode | child node of this compound node. |

#### calculateDistance

|  |  |  |
| --- | --- | --- |
| Description | | |
| It is used to compute the branch distance, i.e., the distance of a set of configuration data from satisfying this compound node. Return a double-typed value. | | |
| Parameter Name | Type | Parameter Description |
| ope | String | The operation name of this compound node, such as *and*, *or*, *implies*, *not* etc. |

#### update

|  |  |  |
| --- | --- | --- |
| Description | | |
| Nothing to do since it will not be invoked by configuration of any type of vps. | | |
| Parameter Name | Type | Parameter Description |
| arg0 | Observable | the object being observed by this node |
| arg1 | Object | this parameter capsulate the type of vp scope (rule node), configured instance id and configuration data for this instance with the following form:  "NODETYPE"+"\*" + instanceID+"\*"+value |

# Zen-Fix API Information

## Class ConformanceFixing

### Operation

#### ConformanceFix

|  |  |  |
| --- | --- | --- |
| Description | | |
| Based on the non-conformities detected, it will build a non-conformity resolving problem and invoke a search algorithm to solve this problem. | | |
| Parameter Name | Type | Parameter Description |
| conindex | String[] | The flag to indicate the problem that contains a list of the value for the configured VP of the constraint closure. |

## Class Informity

### Attribute

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| violatedRule | ArrayList<RuleNode> | The list to record all the violated rules |
| ruleClosure | HashSet<RuleNode> | The variable to record the rules in the closure of the violated rules |

### Operation

#### setViolatedRule

|  |  |  |
| --- | --- | --- |
| Description | | |
| Used to add this rootnode to the violated rule list violatedRule | | |
| Parameter Name | Type | Parameter Description |
| rootnode | RuleNode | The constraint whose root node corresponds to this input rootnode. To be added into the violated rule list. |

#### unsetViolatedRule

|  |  |  |
| --- | --- | --- |
| Description | | |
| Clear all the violated rules from the violated rule list violatedRule | | |
| Parameter Name | Type | Parameter Description |

#### resetViolatedRule

|  |  |  |
| --- | --- | --- |
| Description | | |
| Update the violated rule list violatedRule. Remove all the rules whose evaluation result is true or null. | | |
| Parameter Name | Type | Parameter Description |

#### calculateClousre

|  |  |  |
| --- | --- | --- |
| Description | | |
| Calculate the closure of the violated rules in the violated rule list violatedRule. The definition and the way to calculate the closure can refer to the Zen-Fix paper. | | |
| Parameter Name | Type | Parameter Description |

## Class FixingRecommendation

### Attribute

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| violatedRule | List<RuleNode> | The list to record the root nodes of all the violated rules |
| ruleClosure | HashSet<RuleNode> | The list to record all the rules in the closure of the violated rules |
| configured | HashMap<String, List<Observer>> | All the configured vp instance id and all the rule nodes related with each configured vp instance |
| unconfigured | HashMap<RuleNode, List<String>> | All the rule nodes with one or more unconfigured vp insances |
| vpInfluence | HashMap<VariationPoint, Integer> | The variable to record the impact factor for each vp |
| inferpattern | ArrayList<RuleNode> | Not used anymore |
| vptoVariable | HashMap<String, Integer> | The key of this variable is the instance id of the configured vp instance; the value for each key is the index for this vp instance in the problem encoding space, e.g., for the lowerLimit\_[] or upperLimit\_[] |
| initialvalue | double [] | The initial double value for the user configured value. Will be transformed from String value to double value. |
| fileflag | static int | The flag for the file. Used during the whole configuration process to indicate the problem. |
| problemfilew | FileWriter | The file used to store each problem containing the value of the configured vp instance. |
| concreteprofilew | FileWriter | The file used to store each problem in detail containing the name and the value of the configured vp instance. |

### Operation

#### evaluate

|  |  |  |
| --- | --- | --- |
| Description | | |
| Evaluate the fitness of a solution during search. We defined four heuristics for resolving the nonconformities and the details can refer to the Zen-Fix paper. | | |
| Parameter Name | Type | Parameter Description |
| solution | Solution | The resolving solution generated by search algorithms with crossover and mutation operations. |

#### evaluateConstraints

|  |  |  |
| --- | --- | --- |
| Description | | |
| Evaluate the distance of a solution from satisfying all the rules in the closure of violated rules, in other words, all the conformance rules. solution.setNumberOfViolatedConstraint is used to set the number of violated rules by current solution; solution. setOverallConstraintViolation is used to set the branch distance of the current solution from satisfying all the rules. | | |
| Parameter Name | Type | Parameter Description |
| solution | Solution | The resolving solution generated by search algorithms with crossover and mutation operations. |

#### Code

|  |  |  |
| --- | --- | --- |
| Description | | |
| Return a double value from the String value of a configured vp according to the type of the vp. | | |
| Parameter Name | Type | Parameter Description |
| type | TemplateParameterType | The type of the vp. Can be one of the following: Attr\_BOOL, Attr\_Enum, Attr\_Cardinality or Class\_Inherit |
| vp | VariationPoint | The variation point to be coded |
| value | String | The configured value of the vp. |

#### decode

|  |  |  |
| --- | --- | --- |
| Description | | |
| Return a String value from the double value of a configured vp according to the type of the vp. | | |
| Parameter Name | Type | Parameter Description |
| value | double | The corresponding double value of the vp. |
| vp | VariationPoint | The variation point to be decoded |
| type | TemplateParameterType | The type of the vp. Can be one of the following: Attr\_BOOL, Attr\_Enum, Attr\_Cardinality or Class\_Inherit |

# Zen-CI API Information

## Class DecisionInference

### Attribute

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ruleTobeInferred | ArrayList<RuleNode> | The variable to record all the rules that can be inferred, i.e., whose evaluation result is null |
| inferredVPID | ArrayList<String> | The variable to record the id of all the inferred vp instances |

### Operation

#### doDecisionInference

|  |  |  |
| --- | --- | --- |
| Description | | |
| The interface to invoke the inference for all the rules that might be inferred. | | |
| Parameter Name | Type | Parameter Description |

#### getCanInferedVPsNum

|  |  |  |
| --- | --- | --- |
| Description | | |
| To calculate how many vp instances related with the children of the parent node can be inferred based on the desired value of the parent node. The way to calculate the number can refer to Zen-CI paper. | | |
| Parameter Name | Type | Parameter Description |
| assignedValue | boolean | Either true or false. The desired value of this input node |
| node | RuleNode | The parent node whose children can be inferred |

#### getNotComfiguredChildNum

|  |  |  |
| --- | --- | --- |
| Description | | |
| Return the number of the children of the parent node whose related vp instances are not fully configured. | | |
| Parameter Name | Type | Parameter Description |
| node | RuleNode | The parent node whose children can be inferred |

# EsOCL 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