iOCL

USER'S MANUAL

iOCL: An Interactive Tool for Specifying, Validating and Evaluating OCL Constraints

Muhammad Hammad[1], Shuai Wang[1], Tao Yue[1,3], Shaukat Ali[1] and Jan Nygård[2]

[1]Simula Research Laboratory, Oslo, Norway
[2]Cancer Registry of Norway, Oslo, Norway
[3]University of Oslo, Oslo, Norway

Revision Sheet

Release No.	Date	Revision Description
Rev. 1	12/04/2017	Version 1

This document aims at presenting an interactive tool (named iOCL) for specifying, validating and evaluating OCL constraints, which includes three key parts, i.e., 1) an overview of iOCL (Section 1); 2) how to use iOCL for specifying OCL constraints (Section 2); and 3) an example to illustrate using iOCL to specify OCL constraints (Section 3).

1. An Overview of iOCL

iOCL is designed as a web-based tool application for specifying OCL constraints in an interactive way. More specifically, iOCL takes an UML model as input to identify domain knowledge (i.e., classes and attributes that OCL constraints can work on) and produces a number of OCL constraints based on the domain expertise. Moreover, iOCL includes in total 21 panels that are listed as below, which will be presented in detail in Section 2 to show how to employ iOCL for specifying OCL constraints.

1. Model Input
2. Model Explorer
3. Context Selection
4. Constraint Type
5. Context Operation
6. Attribute Scope
7. Attribute Selection
8. Predefined Operations
9. Operation Type Selection
10. Collection Operation Selection
11. Operation Selection
12. Value Type
13. Value Input
14. Enumeration Selection
15. Enumeration Literal Selection
16. Parameter Selection
17. Condition Type
18. Priority Operation
19. Constraint Panel
20. Utility Functions
21. Class Instance Panel

2. Using iOCL

This section presents how iOCL is employed to support specifying OCL constraints in an interactive manner, which includes model input (Section 2.1) and OCL constraint specifying (Section 2.2).

2.1 Model Input

Figure 1 shows a screenshot of iOCL overview that includes *Model Input* panel and *Model Explorer* panel. More specifically, as the first step, it requires uploading a particular UML model file (e.g., RoyalAndLoyal.uml) within *Model Input* panel. Afterwards, iOCL will parse the uploaded UML model file and enable several main panels as shown in Figure 2, which include 1) *Query* panel that displays the ongoing process of OCL constraint specifying; 2) *Context Selection* panel that is designed to to select the context that a specific OCL constraint works on, e.g., the context will be *Membership* if a user want to add constraint on *Membership* class; and 3) *Utility Functions* panel that includes functions for validating and evaluation of the specified constraints, downloading constraints, downloading models, and specifying a new OCL constraint.

It is also worthy mentioning that after uploading a specific UML model file, *Model Explorer* panel is enabled as shown in Figure 2 (right part), where all the classes with corresponding attributes, associations, operations and instances included in the UML model file can be explored.

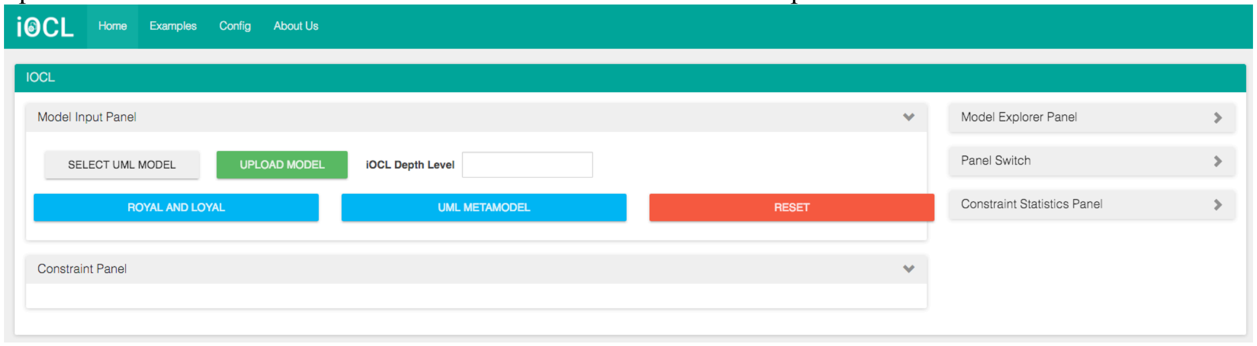


Figure 1 An overview of iOCL

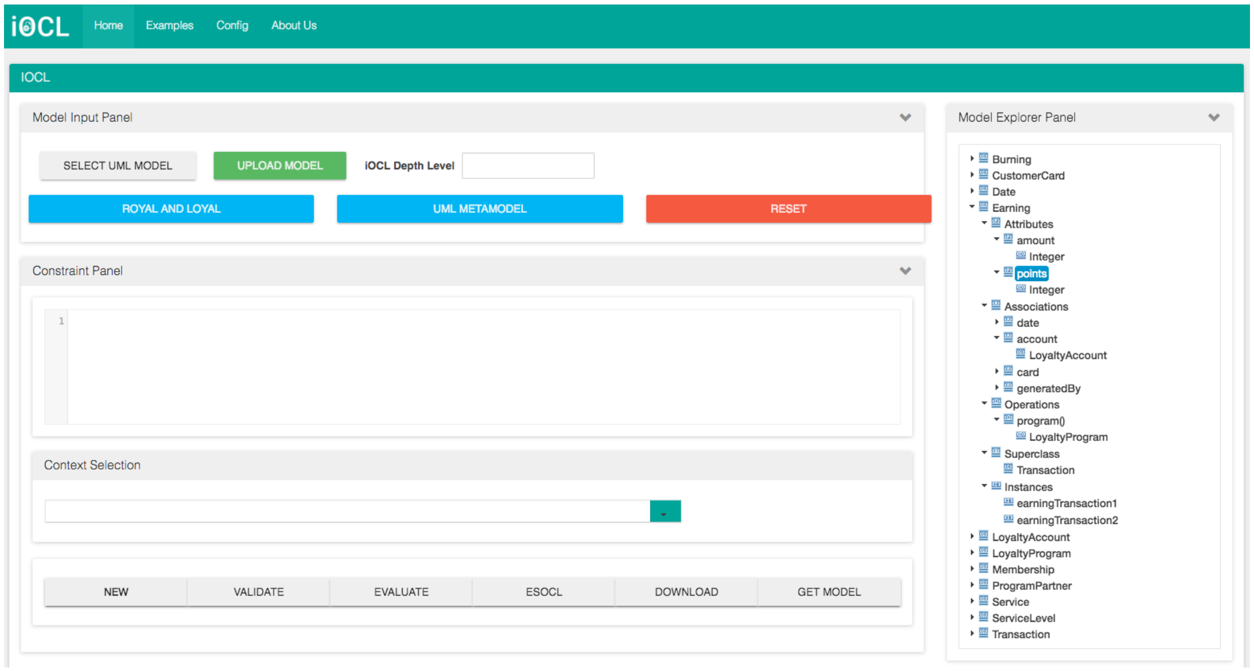


Figure 2 An overview of iOCL after uploading an UML model

Moreover, iOCL provides a *Model Explorer* panel (as a tree structure in right of Figure 2) with the aim to comprehensively display all the inputted UML elements including classes, instances, and attributes. In addition, using this panel, new instances of a class can be created by right clicking the class.

2.2 OCL Constraint Writing

This section details how to apply iOCL to specify OCL constraints in an interactive manner.

Context Selection. Once an UML model is uploaded, the next step is to select a context in which user want to write OCL constraint. iOCL provide interface to user to select context from upload model. The selected context is also use to fetch the corresponding classes that will be later used for providing models with only the relevant details at a given step of constraint specification process. Moreover, iOCL provides two means to select a context for a particular OCL constraint, which include: 1) clicking the context to be selected in the *Model Explorer* panel choosing the context in the drop-down menu in the *Context Selection* panel (left in Figure 3).

Constraint Type. After selecting a proper context, the following step is to specify an OCL constraint type (as shown in Figure 4), i.e., invariant, pre/post or operation body. If pre/post or body condition is selected, *Context Operation Selection* panel will be triggered and displayed with the aim to identify which kind of context operation should be selected as shown in Figure 6.

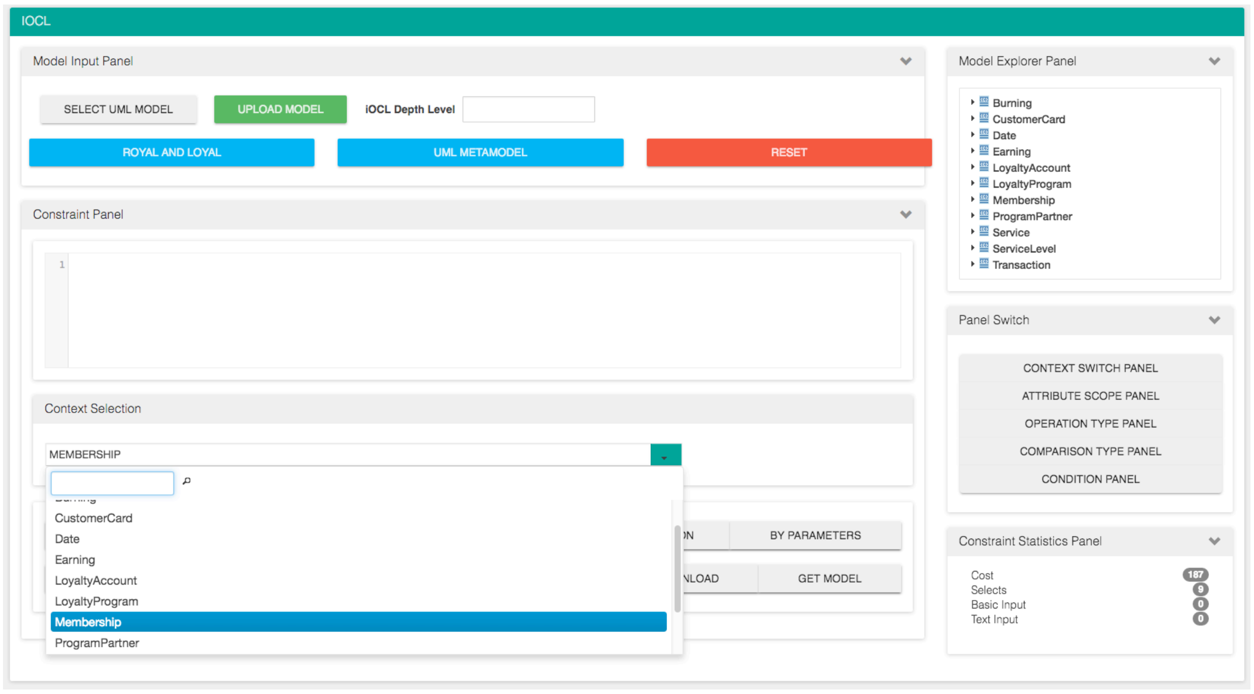


Figure 3 Context Selection Panel

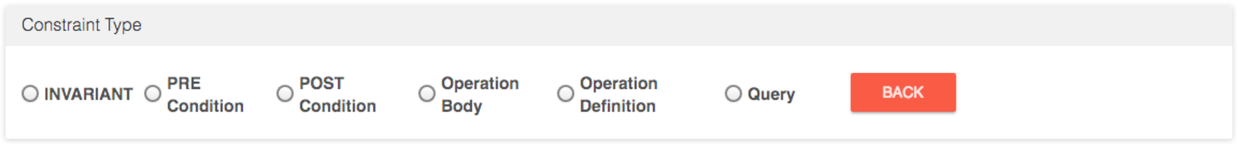


Figure 4 Constraint Type Panel

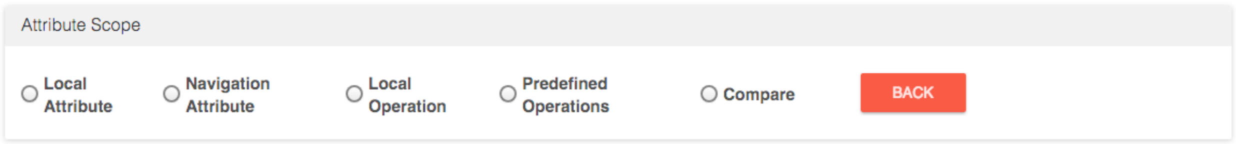
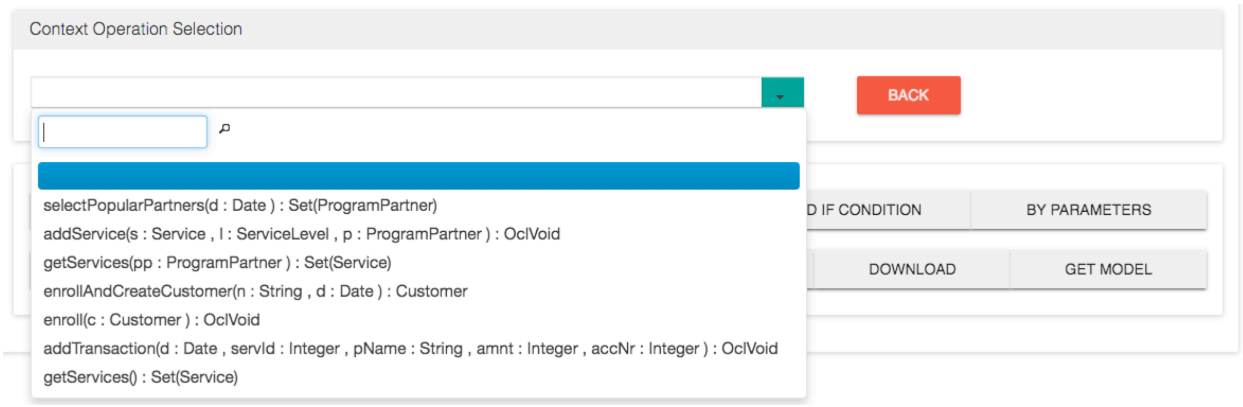


Figure 5 Attribute Scope Panel



selectPopularPartners(d : Date) : Set(ProgramPartner)
addService(s : Service , l : ServiceLevel , p : ProgramPartner) : OclVoid
getServices(pp : ProgramPartner) : Set(Service)
enrollAndCreateCustomer(n : String , d : Date) : Customer
enroll(c : Customer) : OclVoid
addTransaction(d : Date , servId : Integer , pName : String , amnt : Integer , accNr : Integer) : OclVoid
getServices() : Set(Service)

Figure 6 Context Operation Panel

Attribute Scope. As mentioned before, using *Attribute Scope* Panel (Figure 5), it allows users to choose Local Attribute, Association, and Operation that user want to add to the constraint. When selecting Local Attribute, the *Attribute Selection* panel (Figure 7) will be enabled to allow users to select particular properties that will be constrained by an OCL constraint.

Attribute Selection. This panel (Figure 7) lists all the properties for selecting proper attributes to be constrained by a particular OCL constraint. More specifically, iOCL provides three ways for selecting relevant properties, which include: 1) select a property in the drop-down menu (Figure 7); 2) search a property in the search textbox (Figure 7); and 3) select an attribute in the *Model Explorer* panel. Once an attribute is selected, iOCL will display 1) the *Attribute Scope* panel if the selected attribute is association and the association multiplicity is one; or 2) the *Collection Operation* panel if the multiplicity is more than one (as shown in Figure 8). In addition, iOCL will enable the *Operation Type* panel (Figure 9) if the selected attribute is primitive type.

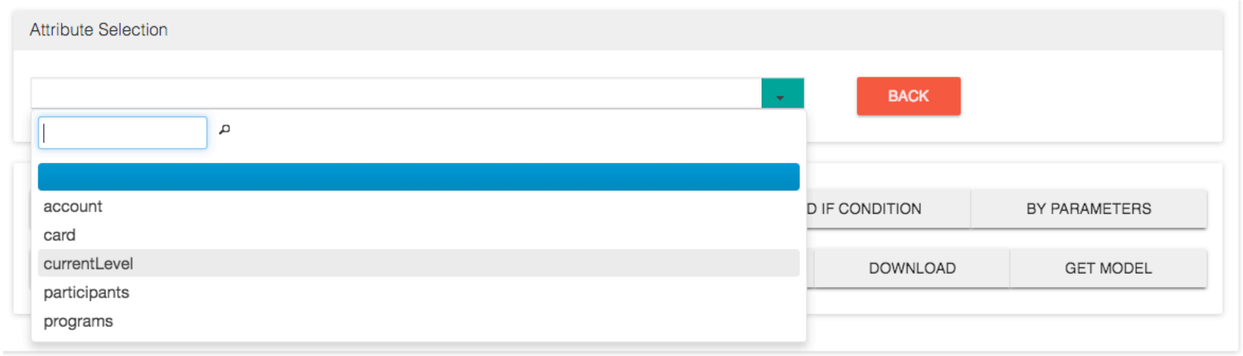


Figure 7 Attribute Selection Panel

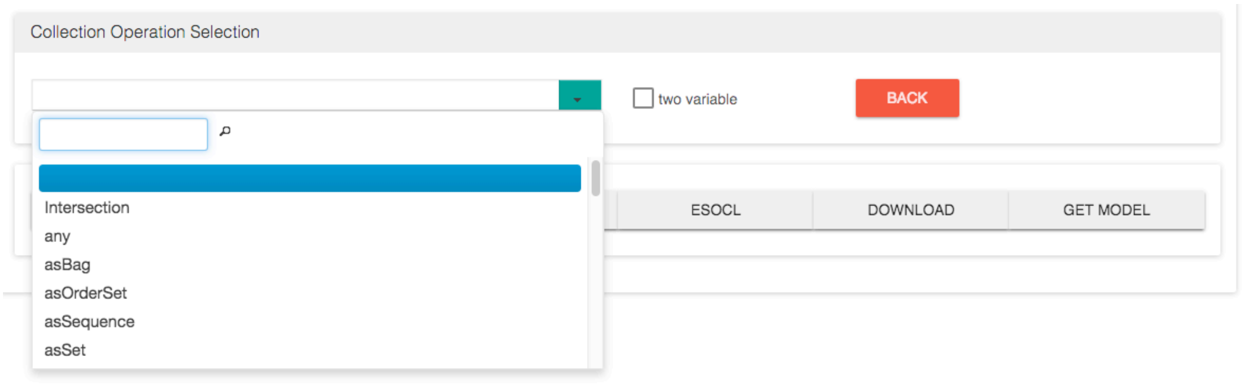


Figure 8 Collection Operation Panel

Operation Type. Within this panel (Figure), it allows users to choose one of two operation types, i.e., Comparison Operation and Arithmetic Operation. More specifically, Equality Comparison is to specify operations among properties (Figure), e.g., equal, not equal, less than, greater than while Value Operation is to specify operations between a property and a value (Figure 8). Notice that determining a value Arithmetic depends on the type of a property, i.e., if the property type is number, iOCL will provide operations related with number type such as add, subtract, multiply and divide operations. If the property type is string, iOCL will provide operations related with string type, e.g., concat, substring and length of string.

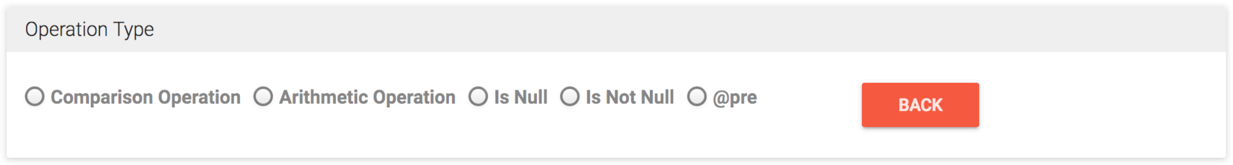
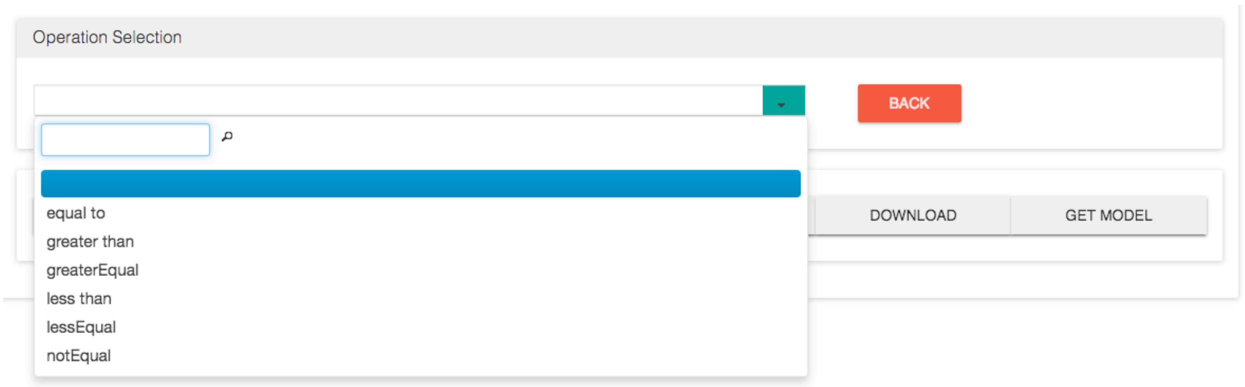


Figure 9 Operation Type



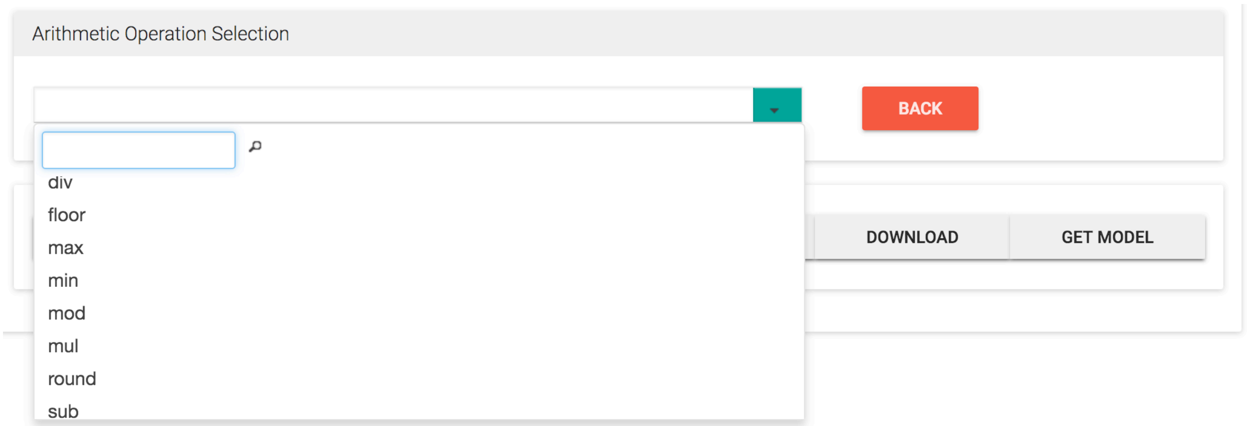


Figure 10 Comparison and Arithmetic Operation

Comparison Value Type. The *Value Type* panel (Figure 11) provides four options (By Value, By Class, By Enumeration and By Parameters) for constraining a particular attribute. More specifically, *By Value* means the attribute will be constrained by a specific value that can be inputted using the *Value Input* panel (Figure 12). For instance, if the value type is Integer (shown as Figure 12), only integer values can be taken as input for constraining the property. If the value type is Boolean, the entered value must be true or false. *By Class* indicates that the attribute will be constrained by another attribute of class by enabling the *Attribute Scope* panel that allows users to select attribute, Operation of class by enabling the *Local Operation* panel (Figure 13). *By Enumeration* and *By Parameters* indicate that the attribute will be constrained by which enumeration literal (Figure 14) or operation parameter, respectively.

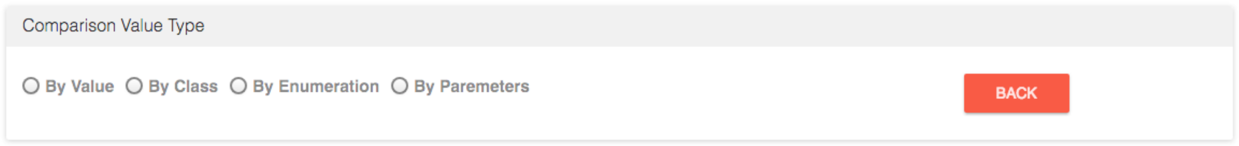


Figure 11 Comparison Value Type

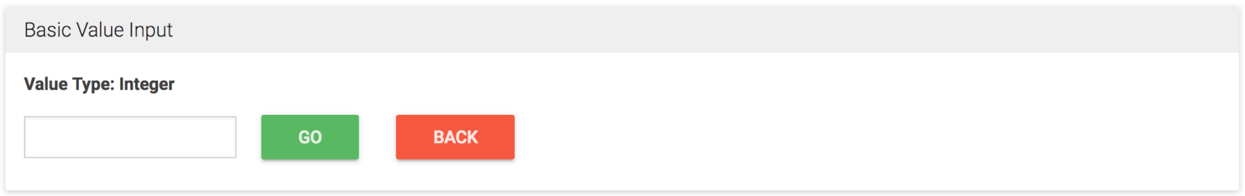


Figure 12 Value Input Panel

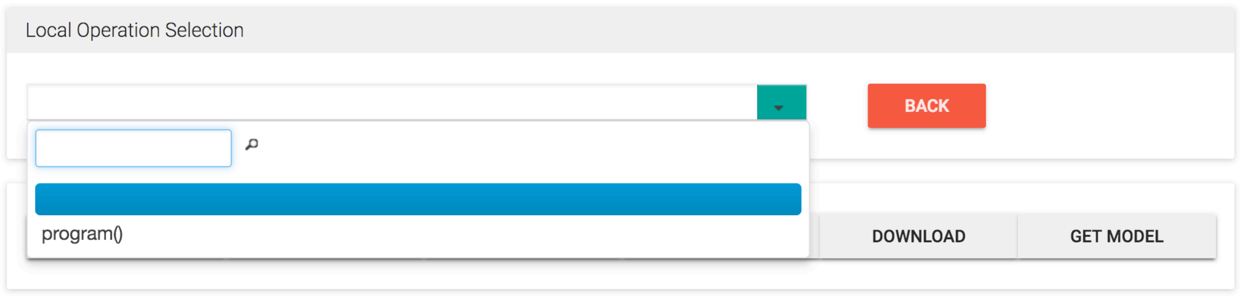


Figure 13 Local Operation Selection

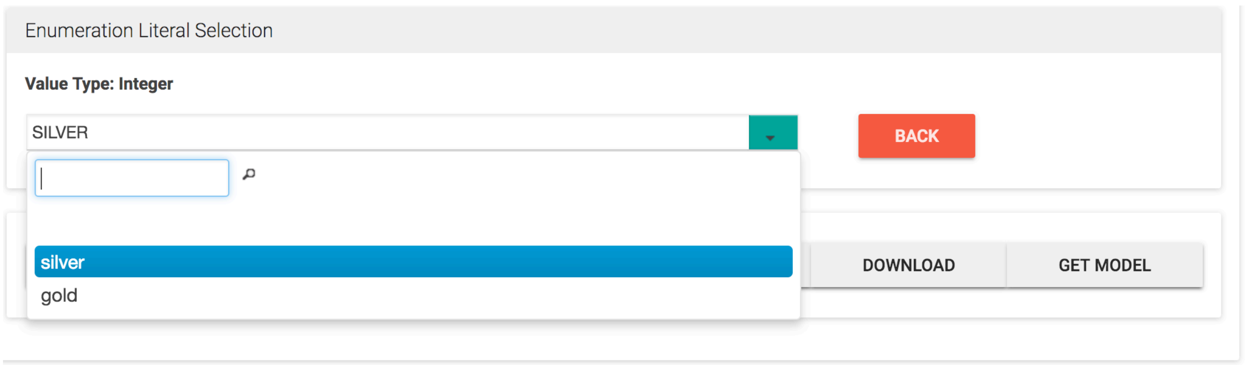


Figure 14 Enumeration Literal Selection

Condition Type. In addition, iOCL also supports several condition types as shown in Figure 15, i.e., “And”, “Or” and “Implies”, which aims to specific logical operations between segments within a constraint.

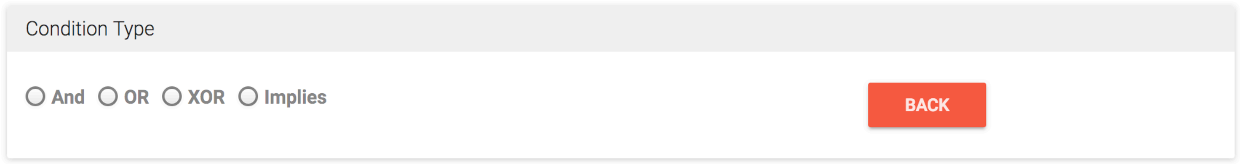
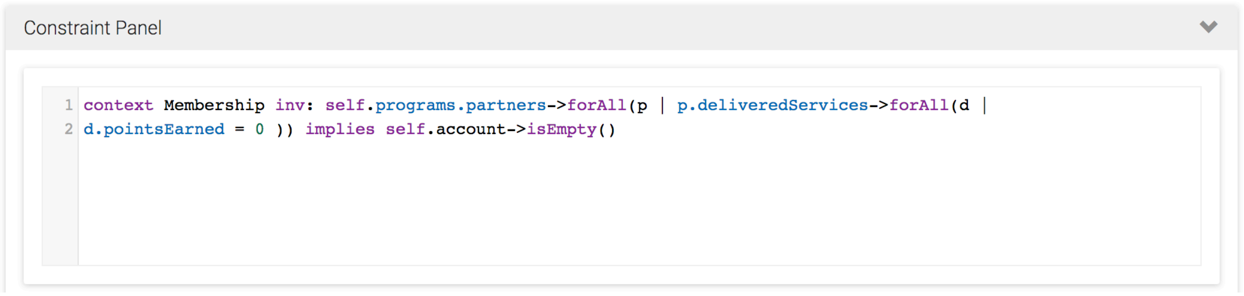


Figure 15 Condition Type Panel

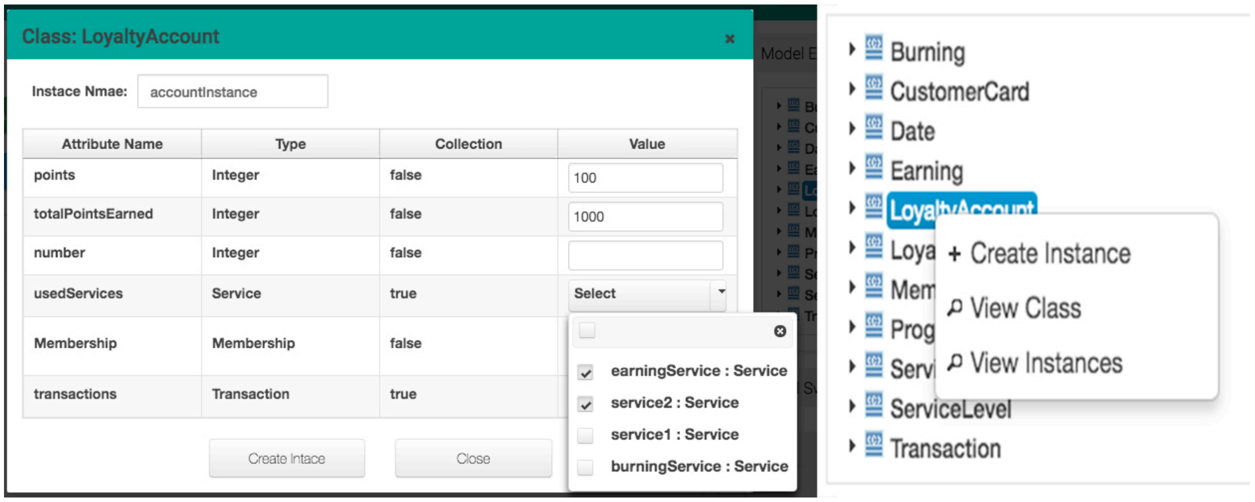
Constraint Panel. As shown in Figure 16, this panel shows the current process of OCL constraint specification, e.g., context and operations generated by iOCL. Within this panel, it is possible to allow users to modify the current constraint (generated by iOCL) in a manual way.



```
context Membership inv: self.programs.partners->forAll(p | p.deliveredServices->forAll(d |
d.pointsEarned = 0 )) implies self.account->isEmpty()
```

Figure 16 Constraint Panel

Instance Panel. As shown in Figure 17, an instance of a particular class (e.g., *LoyaltyAccount* class) can be created by right clicking on the class in the *Model Explorer* panel (right of Figure 17). In addition, we also provide two additional functionalities, i.e., *View Class* and *View Instances* for users to check the details of a class and its instances, respectively. By clicking *Create Instance*, the *Instance* panel will be triggered and displayed to provide all the necessary fields to be filled (e.g., the attribute *points*) for creating a model instance for a specific class.



Class: LoyaltyAccount

Instace Nmae: accountInstance

Attribute Name	Type	Collection	Value
points	Integer	false	100
totalPointsEarned	Integer	false	1000
number	Integer	false	
usedServices	Service	true	Select
Membership	Membership	false	
transactions	Transaction	true	

- [✓] earningService : Service
- [✓] service2 : Service
- [ ] service1 : Service
- [ ] burningService : Service

Create Intace   Close

Figure 17 Class Instance Panel

Utility Functions. With the *Utility Functions* panel (Figure 18), six functions are also supported by iOCL, which include:

1. Writing a new OCL constraint (***NEW***).
2. Validating the syntax of the specified constraint (***Validate***);
3. Evaluating the specified constraint with instance model (***Evaluation***);
4. Evaluating the specified constraint using the external EsOCL tool (***ESOCL***)
5. Downloading the specified constraint (***Downloading***)
6. Downloading the model with specified constrains (***GET Model***)

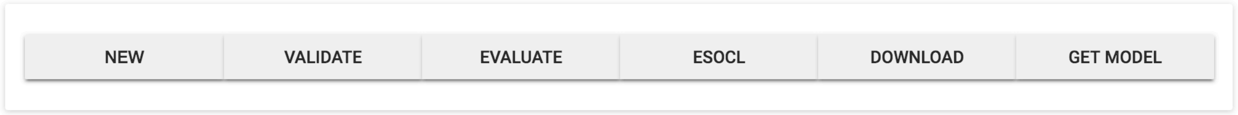


Figure 18 Utility Functions Panel

3. An Example

This section illustrates how to employ iOCL for specifying OCL constraints with a real example. Suppose We need to add an OCL constraint on the class of Loyalty Program of Membership and the plain text for this constraint is "the earned points provided by the Program Partner are 0 implies the Membership account is empty".

The detailed steps of using iOCL for specifying such constraint are listed as below.

Steps:

1) Choose the model file "Royal And Loyal" and upload the UML model by clicking the button "**Upload Model**"
2) Select the "Membership" context from the *Context Selection* panel
3) Select "**INVARIANT**" from the *Constraint Type* panel
4) Select "**Navigation Attribute**" from the *Attribute Scope* panel
5) Select "**programs**" attribute from the *Attribute Selection* panel
6) Select "**Navigation Attribute**" from the *Attribute Scope* panel

7) Select “**partners**” attribute from the *Attribute Selection* panel
8) Select “**forAll**” from the *Collection Operation Selection* panel
9) Select “**Navigation Attribute**” from the *Attribute Scope* panel
10) Select “**deliveredServices**” attribute from the *Attribute Selection* panel
11) Select “**forAll**” from the *Collection Operation Selection* panel
12) Select “**Local Attribute**” from the *Attribute Scope* panel
13) Select “**pointsEarned**” attribute from the *Attribute Selection* panel
14) Select “**Comparison Operation**” from the *Operation Type* panel
15) Select “**Equal to**” from the *Operation Selection* panel
16) Select “**By Value**” from the *Comparison Value Type* panel
17) Enter “**0**” in the textbox from the *Basic Value Input* panel
18) Click “**CLOSE BRACKET**” two time to close forAll Operation
19) Select “**Implies**” from the *Condition* panel
20) Select “**Navigation Attribute**” from the *Attribute Scope* panel
21) Select “**account**” attribute from the *Attribute Selection* panel
22) Select “**Predefined Operations**” from the *Attribute Scope* panel
23) Select “**IS Empty**” from the *Predefined Operations* panel
24) Click the “**Validate**” button to check the syntax of the specified constraint