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

Muhammad Hammad[1], Tao Yue[1,2], Shaukat Ali[1], Shuai Wang[1]

[1]Simula Research Laboratory, Oslo, Norway
[2] University of Oslo, Oslo, Norway
{hammad, tao, shaukat, shuai}@simula.no

Abstract. The Object Constraint Language (OCL) is frequently used to specify additional constraints on models, in addition, to the ones enforced by semantics of the models. It is a well-known fact that due to the lack of familiarity with OCL, practitioners and even researcher to some extent are reluctant in using OCL. To help practitioners and researchers in writing OCL constraints for their specific problem at hand, we developed a tool called interactive OCL (iOCL) for interactively specifying constraints on a given model. The basic philosophy behind the tool is to present only those details (e.g., operations) of OCL to modelers that are valid at a given step of constraint specification process, in addition to helping modelers with its syntax. Our ultimate aim is to reduce the effort required to specify constraints, subsequently lowering down training cost and increasing the correctness of the constraints. iOCL is a web-based application that integrates other tools including Eclipse OCL for validation and evaluation of OCL constraints, and EsOCL for automatically generating valid instances of models that satisfy the specified constraints.

1. Introduction

To successfully apply a model-based engineering (MBE) solution in practice, the key challenge to overcome is to construct required models in a cost-effective manner. For example, applying a model-based testing solution requires test engineers to construct test ready models in a particular modeling language (e.g., Unified Modeling Language (UML) [8]), from which executable test cases can be generated. Constructing such test ready models in a cost-effective manner is the key factor that determines whether the proposed approach can be successfully applied in practice. In the past, we have developed some MBE solutions [1, 2, 3, 4, 5, 6], most of which are based on UML and its profiles, and moreover some of which require using the Object Constraint Language (OCL) to specify various constraints such as state invariants on states of a state machines. Due to less acquaintance with OCL and being declarative, practitioners and researchers are hesitant to use the OCL. To assist practitioners and researchers in specifying OCL constraints, we developed a tool called interactive OCL (iOCL) to interactively specify constraints.

The underlying idea behind iOCL is to guide users through constraint specification process step by step, in an interactive manner. There are three types of user operations in iOCL: selection, basic value input, and text input. Through these three user operations, a user interacts with iOCL to specify OCL constraints. The overall aim of iOCL is to minimize the use of the value input and text input operations and maximize the use of the selection operation; therefore, chances for users to make errors can be reduced. In addition, it decreases the extent of OCL knowledge required from a modeler. The selection user operation is performed by making a choice from a list of available options provided by iOCL that are valid at a given step in constraint specification step. Depending on the type of the UML model element, an association end multiplicity, or even the type of a collection resulting from one or more navigations from the contextual classifier, iOCL dynamically updates selection options at a given step of a specification process. Regarding the basic value input user operation, users are prompted to input basic values for basic types when necessary, at a given step of a specification process. For example, iOCL displays a text box for users to input an integer value for an integer type of properties. In terms of the text input user operation, there are two situations, where users can input text in a text box.

iOCL aims to automate an OCL constraint specification process as much as possible. The key automation of iOCL is to dynamically shortlist available options at any given step, such that a user can perform the selection user operation in a more efficient way. With this objective in mind, iOCL systematically checks the UML model, the already specified partial OCL constraint and the current step of the specification process. The second key automation feature of iOCL is to categorize types of constraints, OCL operations, and properties and only display relevant ones at a given step. This feature helps to reduce specification effort and potential errors that a user might make when manually writing an OCL constraint. iOCL also allows a user to roll back to previous steps of a specification process. Moreover, iOCL automatically takes cares of bracket pairing and automatically fills out left and right brackets thus assisting a modeler with syntax.

iOCL is a web-based application, which is built on various existing technologies: Eclipse Modeling Framework (EMF) [9], Eclipse OCL [10], Eclipse UML2 [11], and EsOCL [3]. With iOCL, specified OCL constraints can be validated to check their syntax, and evaluated to check their correctness with predefined instances and/or with the automated generated ones with the help of EsOCL, an OCL constraint solver. Interested users can try iOCL here: http://dnat.simula.no:50753/IOCL

2. Architecture of iOCL

The architecture of iOCL is presented in Figure 1. As shown in the figure, iOCL has five key functionalities: 1) reading an UML model, on which OCL constraints can be specified, 2) specifying constraints, 3) validating specified constraints, 4) evaluating the correctness of specified constraints, and 5) exporting specified OCL constraints and relevant information. A UML is first loaded and then parsed. The obtained model elements are stored in a model repository, which are queried by *Specifier* to support the interactive specification of OCL constraints. The implementation of *Constraint Validator* and *Evaluator* relies on three existing technologies: Eclipse UML2, Eclipse OCL and EsOCL [5]. EsOCL is a search-based OCL solver that uses various heuristics defined based on constructs of OCL that are implemented as a fitness function.

Search algorithms, e.g., Genetic Algorithms and (1+1) Evolutionary Algorithm, can use such fitness function to guide for solving OCL constraints.

The implementation of the iOCL tool is divided into two parts: *Front End* and *Back End*. The *Back End* of iOCL implements the five functionalities described above. The *Front End* of iOCL is in charge of the interaction with users via user interfaces, which is implemented with JavaServer Faces (JSF) [12], which is a well-known framework for developing user interfaces and web applications. The *Query Analyzer* is the action listener, which listens to all the actions from the *Back End*, and communicates with *Front End Controller*. Furthermore, iOCL defines a set of view templates, which are filled with properties and operations at runtime. Filled templates are returned to end users.

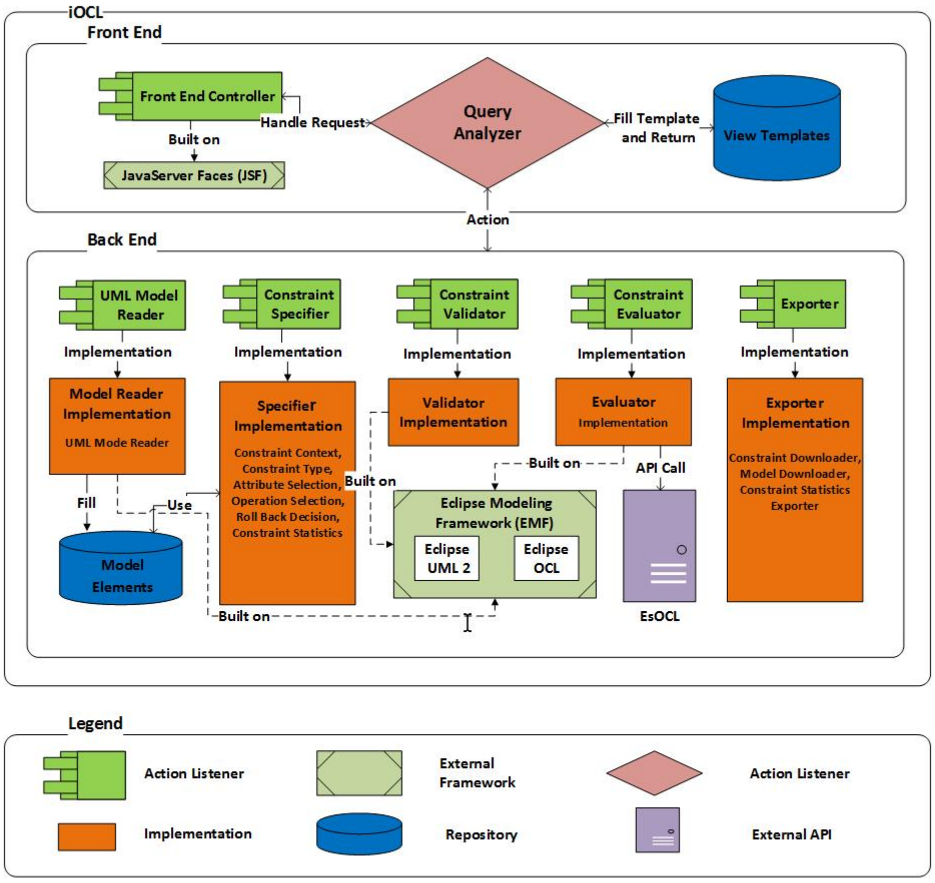
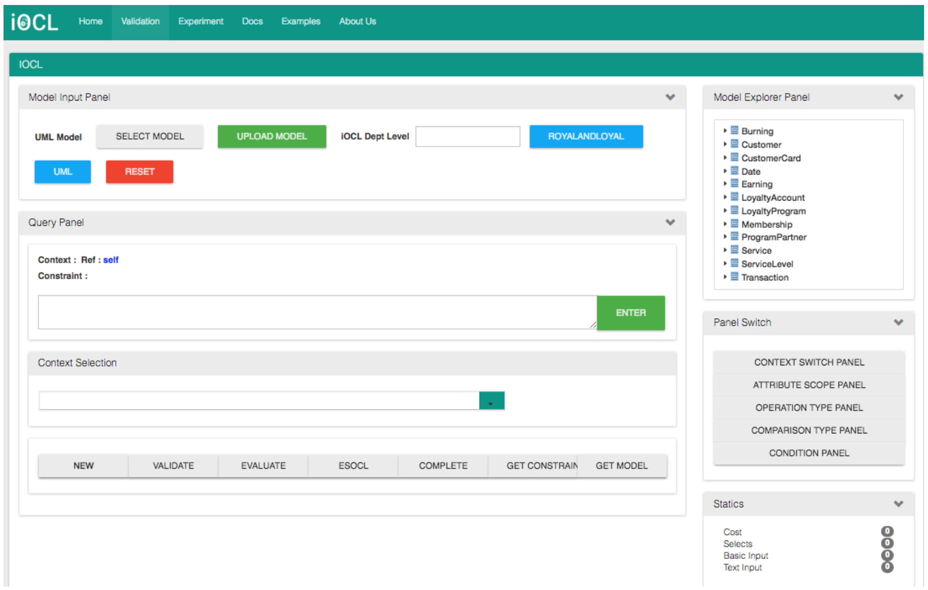


Figure 1 iOCL Architecture

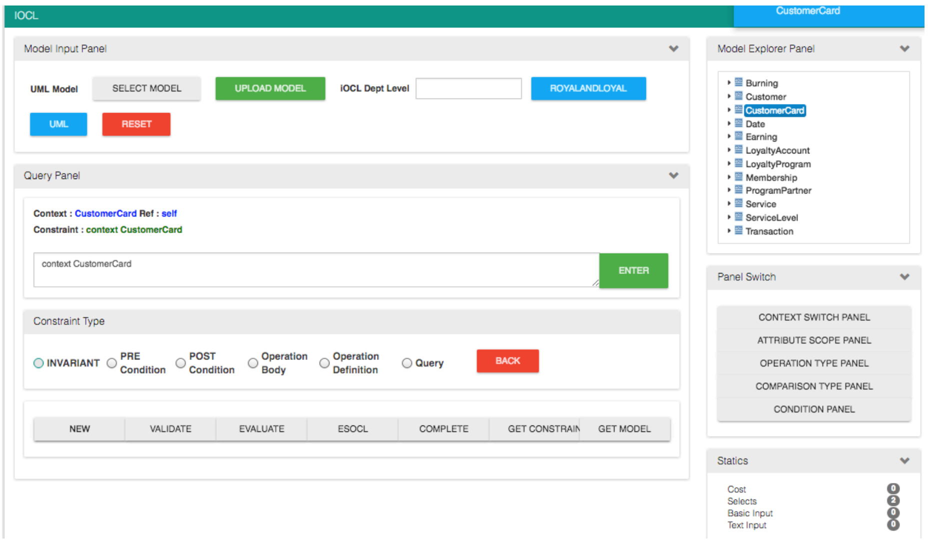
3. Tool Demonstration Details

In this section, we present screenshots of the key functionalities of iOCL. The YouTube video can be found from the link below:

https://www.youtube.com/watch?v=Wgi9YYMp7Q4



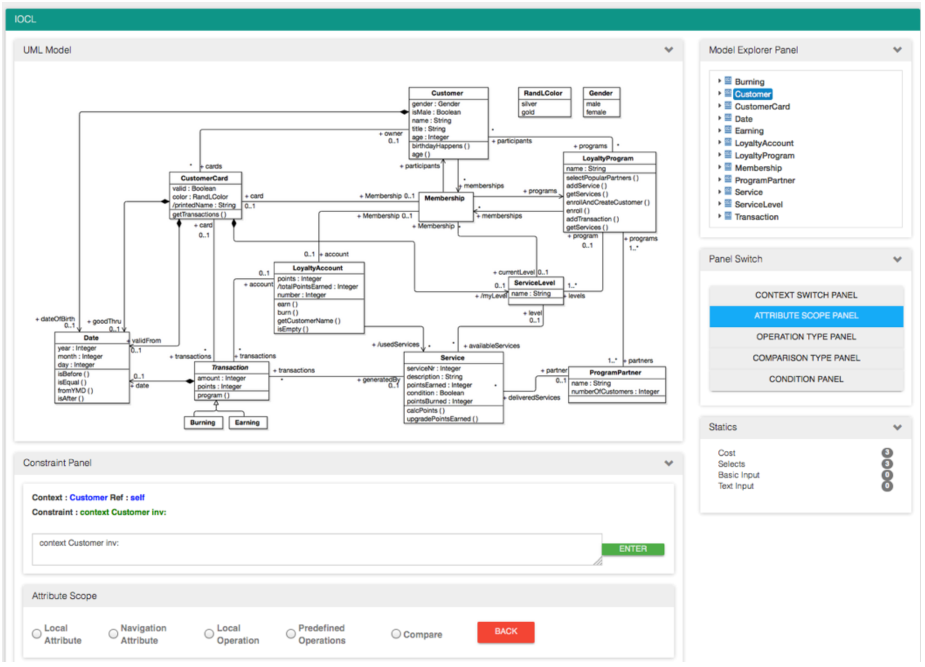
The screenshot above displays the initial interface of the tool. One can select a model, which can be either .uml or .ecore file, to start with. For the convenience, in the current interface, we also provide two examples available: the Royal and Loyal case study and the UML 2.4 metamodel. The original Royal and Loyal case study is from [13]. As shown on the right hand side, all of the model elements of a loaded model are automatically displayed in the list, from which one can select one as the contextual element to specify an OCL constraint. One can also select the context from the dropdown *Context Selection* list.



Context : CustomerCard Ref : self
Constraint : context CustomerCard

context CustomerCard

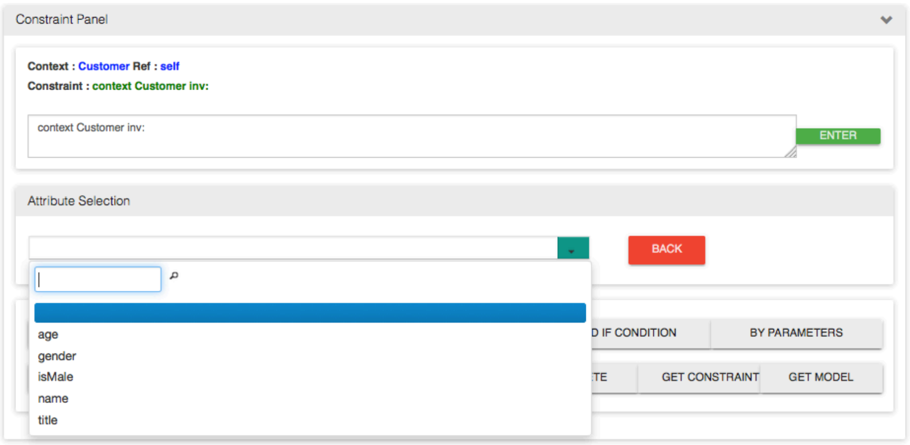
When one selects the context, iOCL automatically displays a list of options: *Invariant*, *Pre Condition*, *Post Condition*, etc. Of course, a user can also roll back to her/his previous selection via the *Back* button, shown in the screenshot above.



Context : Customer Ref : self
Constraint : context Customer inv:

context Customer inv:

From the screenshot shown above, when a user selects the context, a list of options is displayed in the *Attribute Scope* pane. When the user selects the *Local Attribute* option, a dropdown list is displayed to show all the local attributes of class *Customer*, as shown in the screenshot below.


Constraint Panel

Context : Customer Ref : self

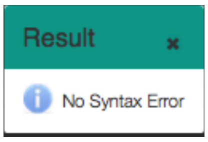
Constraint : context Customer inv:

context Customer inv:

Attribute Selection

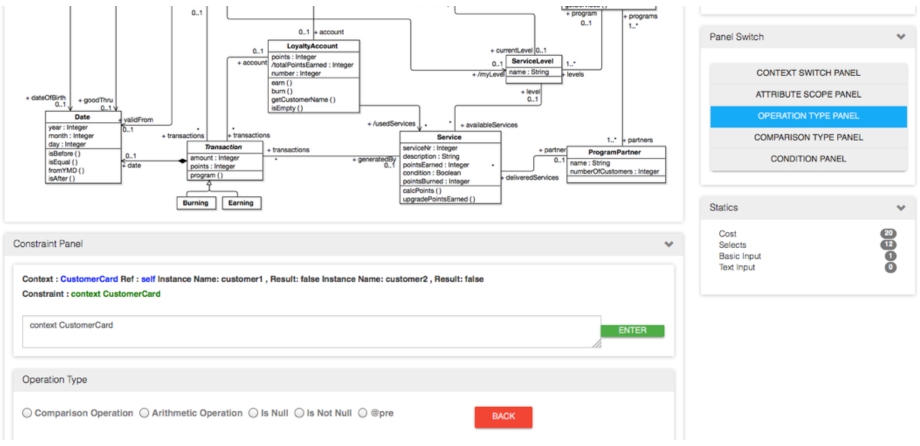
age
gender
isMale
name
title

When the user finishes the specification, she/he can click the *Validate* button to validate the syntactic correctness of the specified constraint and results will be returned to the user in a second, as shown in the screenshot below.


Result

No Syntax Error

One can also switch from one panel to another if needed. As shown in the screenshot below, during the specification process, one can switch to *Operation Type* panel, which triggers the display of a list of operation types such as *Comparison Operation*.



Panel Switch

CONTEXT SWITCH PANEL
ATTRIBUTE SCOPE PANEL
OPERATION TYPE PANEL
COMPARISON TYPE PANEL
CONDITION PANEL

Statics

Cost	20
Selects	12
Basic Input	1
Text Input	0

Constraint Panel

Context : CustomerCard Ref : self Instance Name: customer1 , Result: false Instance Name: customer2 , Result: false

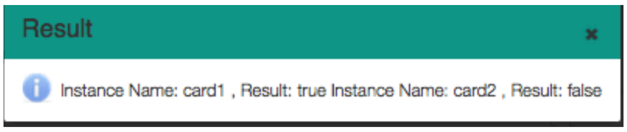
Constraint : context CustomerCard

context CustomerCard

Operation Type

○ Comparison Operation ○ Arithmetic Operation ○ Is Null ○ Is Not Null ○ @pre

One can also evaluate the specified constraint based on predefined instances by clicking the *Evaluate* button, as shown below. Relying on EsOCL, iOCL also helps in assessing the correctness of the specified constraint. If EsOCL can solve the constraint provides an instance, it means that the constraint is correctly specified and is solvable. Otherwise, there are two options, either the specified constraint may have an issue and a warning will be given to the end user or EsOCL wasn't successful in solving the constraint.



Instance Name: card1 , Result: true Instance Name: card2 , Result: false

4. Conclusion

We presented an interactive OCL constraint specification tool called interactive OCL (iOCL) with the aim of assisting modelers in interactively specifying OCL constraints. The underlying idea behind iOCL is to present only the relevant details to a modeler at a given step of constraint specification process in addition to pre-filling the syntax with the aim of reducing the OCL knowledge required to specify constraints and ultimately reducing the training cost of using OCL.

REFERENCES

[1] Wang, S., Lu, H., Yue, T., Ali, S., Nygård, J.: MBF4CR: A Model-Based Framework for Supporting an Automated Cancer Registry System. In: European Conference on Modelling Foundations and Applications, pp. 191-204. Springer, (2016)
[2] Hong, L., Tao, Y., Shaukat, A., Li, Z.: Model-based Incremental Conformance Checking to Enable Interactive Product Configuration. Information and Software Technology 25 (2015)
[3] Ali, S., Iqbal, M.Z., Arcuri, A., Briand, L.C.: Generating test data from OCL constraints with search techniques. IEEE Transactions on Software Engineering 39, 1376-1402 (2013)
[4] Ali, S., Yue, T., Iqbal, M.Z., Panesar-Walawege, R.K.: Insights on the use of OCL in diverse industrial applications. In: International Conference on System Analysis and Modeling, pp. 223-238. Springer, (2014)
[5] Iqbal, M.Z., Ali, S., Yue, T., Briand, L.: Applying UML/MARTE on industrial projects: challenges, experiences, and guidelines. SoSyM Journal 14, 1367-1385 (2015)
[6] Lu, H., Yue, T., Ali, S., Zhang, L.: Nonconformity Resolving Recommendations for Product Line Configuration. International Conference on Software Testing, International Conference on Software Testing, Verification and Validation (ICST), pp. 57-68, (2016)
[7] Object Constraint Language (OCL), http://www.omg.org/spec/OCL/
[8] Unified Modeling Language (UML), http://www.omg.org/spec/UML/2.5/
[9] Eclipse Modeling Framework (EMF), https://eclipse.org/modeling/emf/
[10] Eclipse OCL, http://wiki.eclipse.org/OCL
[11] Eclipse UML2, http://wiki.eclipse.org/MDT-UML2
[12] JavaServer Faces (JSF), http://www.oracle.com/technetwork/java/javaee/javaserverfaces-139869.html
[13] Jcabot/ocl-repository, https://github.com/jcabot/ocl-repository/