Science of Computer Programming 149 (2017) 3–8

Contents lists available at ScienceDirect

Science of Computer Programming

www.elsevier.com/locate/scico

Original software publication

IOCL: An interactive tool for specifying, validating and evaluating OCL constraints

Muhammad Hammad [a], Tao Yue [a,b,*], Shuai Wang [a], Shaukat Ali [a], Jan F. Nygård [c]

[a] Simula Research Laboratory, Norway
[b] University of Oslo, Norway
[c] Cancer Registry of Norway, Norway

ARTICLE INFO

Article history:
Received 20 January 2017
Received in revised form 5 July 2017
Accepted 19 July 2017
Available online 16 August 2017

Keywords:
OCL constraints
Interactive tool
UML models

ABSTRACT

The Object Constraint Language (OCL) is commonly used for specifying additional constraints on models, in addition, to the ones enforced by the semantics of the models. However, a lot of practitioners and even researchers are reluctant in using OCL to some extent due to the lack of sufficient familiarity with OCL. To facilitate practitioners and researchers in specifying OCL constraints, we designed and developed a web-based tool called interactive OCL (iOCL) for interactively specifying constraints on a given model. The core idea behind iOCL is to present and display only relevant details (e.g., operations) of OCL to users at a given step of constraint specification process, in addition to helping modelers with its syntax. We evaluated iOCL using a real-world case study from Cancer Registry of Norway and the results showed that iOCL can significantly reduce the time required to specify OCL constraints and decrease the possibility of making syntactic errors during the specification process. Thus, we conclude that iOCL can facilitate the process of OCL constraint specification. Interested users can try iOCL at: http://iocl.zen-tools.com/.

* Corresponding author.
E-mail addresses: hammad@simula.no (M. Hammad), tao@simula.no (T. Yue), shuai@simula.no (S. Wang), shaukat@simula.no (S. Ali), Jan.Nygard@kreftregisteret.no (J.F. Nygård).

http://dx.doi.org/10.1016/j.scico.2017.07.006
0167-6423/

Software metadata

(executable) Software metadata description	
Current software version	V1.1
Permanent link to executables of this version	https://github.com/ScienceofComputerProgramming/SCICO-D-17-00035
Legal Software License	GNU General Public License (GPL)
Computing platform/Operating System	Java Virtual Machine
Installation requirements & dependencies	Java 8, Tomcat 8
If available Link to user manual – if formally published include a reference to the publication in the reference list	http://dnat.simula.no:50753/IOCL/docs.iocl
Support email for questions	hammad@simula.no, tao@simula.no, shuai@simula.no, shaukat@simula.no

Code metadata

Code metadata description	
Current Code version	V1.1
Permanent link to code/repository used of this code version	https://github.com/ScienceofComputerProgramming/SCICO-D-17-00035
Legal Code License	GNU General Public License (GPL)
Code Versioning system used	git
Software Code Language used	Java
Compilation requirements, Operating environments & dependencies	Mac OS or Windows, Intellij idea IDE, Maven, Tomcat 8
If available Link to developer documentation/manual	http://dnat.simula.no:50753/IOCL/docs.iocl
Support email for questions	hammad@simula.no, tao@simula.no, shuai@simula.no, shaukat@simula.no

1. Introduction

Model-Based Engineering (MBE) aims at employing models (specified in for example Unified Modeling Language (UML) [8]) to facilitate the process of software development (e.g., model-based testing (MBT) [2][4]), which has been researched to a large extent in the recent decades [1–6]. To successfully apply a MBE solution in practice, the key challenge is to construct required models in a cost-effective manner. For example, applying a MBT solution requires test engineers to construct test ready models using a particular modeling language (e.g., UML), from which executable test cases can be generated [2]. In the past, we have developed a number of MBE solutions [1–6], most of which are based on UML and its profiles, and some of which require employing Object Constraint Language (OCL) to specify various constraints such as state invariants on states of a state machine (e.g., [1][2]). Based on our experience of applying OCL, we observe that practitioners and researchers are sometimes hesitant to apply OCL due to its declarative nature. Therefore, it is essential to seek an efficient and user-friendly means to assist practitioners and researchers to specifying OCL constraints.

To specify OCL constraints in an efficient way, we designed and developed a web-based application called interactive OCL (iOCL) built on the top of several existing technologies: Eclipse Modeling Framework (EMF) [9], Eclipse OCL [10], Eclipse UML2 [11], and EsOCL [2]. Notice that we designed iOCL as a web application with the aim at easing users' access to the tool without a need to download, install and configure before using. The core idea of iOCL is to guide modelers through constraint specification process step by step in an interactive manner. Generally speaking, iOCL consists of three types of user operations, i.e., *selection operation*, *basic value input operation* and *text input operation*. More specifically, the *selection operation* is to perform a choice from a list of available options provided by iOCL that are valid at a given step when specifying an OCL constraint. Notice that iOCL dynamically updates selection options based on the type of an UML model element, an association end multiplicity, or even the type of a collection resulting from one or more navigations from the contextual classifier. For the *basic value input operation*, users are required to input basic values for basic types (e.g., Integer, Boolean) at a given step of a specification process. In terms of the *text input operation*, a free text box is provided to users for input. Through these three user operations, a user dynamically interacts with iOCL during the process of OCL constraint specification. Notice that the goal of iOCL is to minimize the use of the *basic value input operation* and *text input operation* and maximize the use of the *selection operation* and thereby reducing the potential chances for users to make syntactic errors. Furthermore, more use of the *selection operation* can decrease the extent of OCL knowledge required since users will be guided by the available options provided by iOCL when specifying OCL constraints.

We have conducted a pilot study in the form of controlled experiments, which is reported in [16], with the aim at evaluating iOCL using a real-world case study from Cancer Registry of Norway (CRN).[1] Results of the pilot study show that 1) iOCL can manage to significantly reduce the time effort required for OCL constraint specification; 2) iOCL can decrease the possibility to make syntactic errors during the specification process and 3) the medical experts from CRN opinionated that iOCL is easy to use for specifying OCL constraints.

This paper is an extended version of a tool demonstration paper [14] and the key differences are summarized as follows: 1) the motivation of iOCL is detailed and a brief background knowledge related with UML and OCL is presented for a wider range of readers; 2) the architecture, key features (functionalities) and implementation of iOCL is further elaborated;

[1] Cancer Registry of Norway: https://www.kreftregisteret.no.