Electronic Communications of the EASST
Volume 44 (2011)

Proceedings of the Workshop on OCL and Textual Modelling (OCL 2011)

OCL Tools Report based on the IDE4OCL Feature Model

Joanna Chimiak–Opoka, Birgit Demuth, Andreas Awenius, Dan Chiorean, Sébastien Gabel, Lars Hamann, Edward Willink

18 pages

Guest Editors: Jordi Cabot, Robert Clariso, Martin Gogolla, Burkhart Wolff
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

OCL Tools Report based on the IDE4OCL Feature Model

Joanna Chimiak–Opoka[1], Birgit Demuth[2], Andreas Awenius[3], Dan Chiorean[4], Sébastien Gabel[5], Lars Hamann[6], Edward Willink[7]

[1] University of Innsbruck, Austria, joanna.opoka@uibk.ac.at
[2] Technische Universität Dresden, Germany, birgit.demuth@tu-dresden.de
[3] EmPowerTec AG, Petershausen, Germany, andreas.awenius@empowertec.de
[4] Babes-Bolyai University, Romania, chiorean@cs.ubbcluj.ro
[5] CS Communication & Systems Group, Toulouse, France, sebastien.gabel@c-s.fr
[6] Universität Bremen, Germany, lhamann@informatik.uni-bremen.de
[7] Eclipse Modeling Project, ed@willink.me.uk

Abstract: Previously we have developed the idea of an Integrated Development Environment for OCL (IDE4OCL). Based on the OCL community's feedback we have also designed and published an IDE4OCL feature model. Here we present a report on selected OCL tools developed by the authors and their teams. Each author gives an overview of their OCL tool, provides a top level architecture, and gives an evaluation of the tool features in a web framework. The framework can also be used by other potential OCL users and tool developers. For users it may serve as an aid to choose a suitable tool for their OCL use scenarios. For tool developers it provides a comparative view for further development of the OCL tools. Our plans are to maintain the collected data and extend this web framework by further OCL tools. Additionally, we would like to encourage sharing of OCL development resources.

Keywords: OCL, IDE4OCL, feature model, OCL tool

1 Introduction

For almost 15 years the Object Constraint Language (OCL) has been extensively discussed and used in multiple contexts. At the beginning it was used only on paper without parsing and any kind of checking. Subsequently OCL became a language supported by a range of serious tools and tool environments. However, most OCL users experienced that the OCL tools did not fulfill all of their desired requirements. Therefore in 2009, we started to work on an idea for an Integrated Development Environment for OCL (IDE4OCL). The overview of our research process is depicted in Fig. 1.

In the first step we did a systematic requirements analysis for an IDE4OCL [CDSR09]. We provided a definition of domain concepts from the pragmatic perspective of how OCL is used. It included an OCL tools landscape overview describing the interactions between an IDE4OCL and other modelling and development tools, use cases for each of the components, and 21 features of an IDE4OCL.

In 2010, based on an extensive online survey[1], comments of the survey participants and in-

[1] The survey is still open and available at http://squam.info/survey/index.php?sid=11161. The statistical evaluation of the collected and analysed feedback from over 100 respondents is to be published.

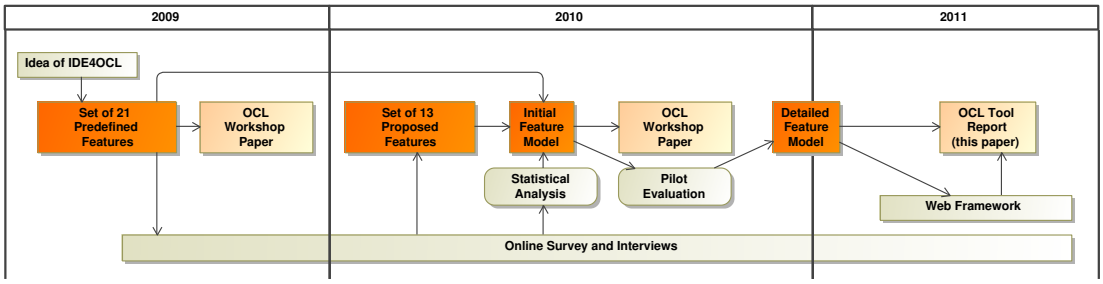

Figure 1: An overview of the research related to the idea of an IDE4OCL.

terviews within the OCL community we analysed, extended and evaluated the set of features. Then we categorized this consolidated set of features into groups of features. The three top level categories are: language and model support, user–friendly support and architectural support. Furthermore all features were designated as mandatory, optional or alternative. This categorization resulted in a feature model [CD10a]. We initially used the feature model for a structured description of tools developed in Dresden and Innsbruck in a joint developers meeting. In this meeting we made slight improvements to the feature model.

This year in order to collect data from the community of tool providers, we developed a web framework that enables choice of tools based on feature selection[2]. To extend the scope of our research we decided to do an analysis of well–known OCL tools from the IDE4OCL perspective. Our ultimate goal is to look for academic and industrial partners who are willing to collaborate in the further development of the IDE4OCL vision within an open source project.

The remainder of this paper is structured as follows: In the next section we present used methodology. In Section 3, we give systematic descriptions of the selected OCL tools. Section 4 gives an analysis of these descriptions along with the data collected in the web framework. Section 5 summarizes the results of the OCL tools report and proposes the next steps in OCL tool development.

2 Methodology

In this section we provide a description of our methodology for selection of the OCL tools (Section 2.1), gathering tool descriptions (Section 2.2) and the web framework to collect detailed data (Section 2.3).

2.1 Selection Criteria

From our perspective an IDE4OCL tool is the core component that provides a user–friendly front end to other OCL related tools (Fig. 2). The main functionality of an IDE4OCL is to support the user in the specification, evaluation and verification of OCL statements as well as in project management.

[2] It is available at http://ide4ocl.opoki.com/featuremodel/.

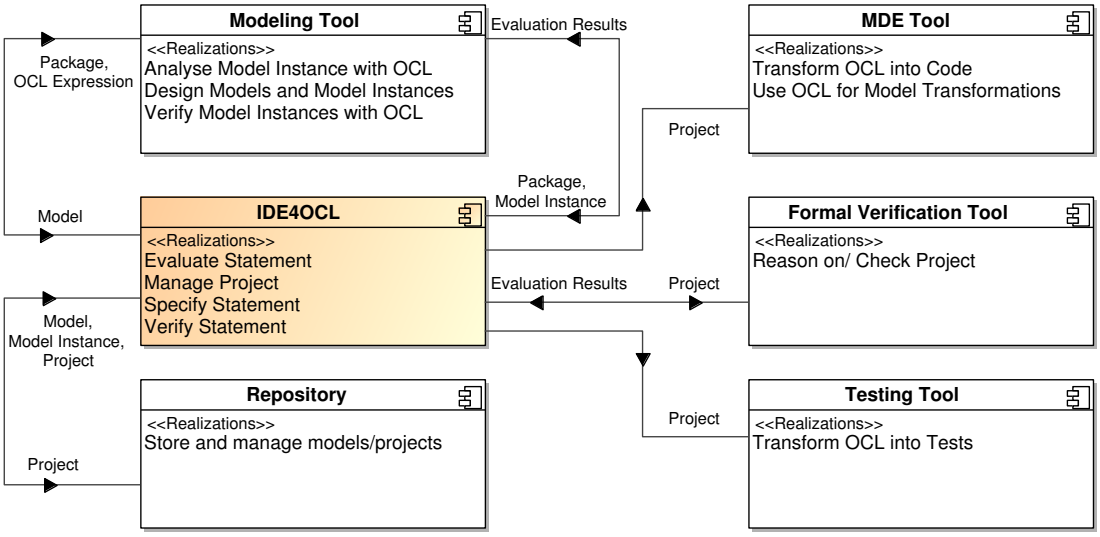

Figure 2: A simplified OCL tool landscape from [CDSR09], where an exact description of all components can be found.

We tried to select a representative collection of OCL tools best fitting to the idea of IDE4OCL (Table 1). We do not claim that our list is complete and a longer list covering various types of OCL tools is available at the OCL Portal[3]. There remain other OCL tools to be studied to get a full picture of the state of the art.

2.2 Gathering Descriptive Data

We contacted the developers of the selected tools and asked them to contribute to our report. Each tool developer gave a short characterization of the OCL tool including a top level architecture view of the OCL and related tool components in relation to the OCL tools landscape (Section 3). Moreover, each person evaluated the features of the respective tool based on the IDE4OCL feature model using our web framework (Section 2.3).

2.3 Gathering Detailed Data

For the purpose of this research we required a tool combining feature modelling and a survey engine. The technical requirement was multi–user / web access. As we could not find a suitable framework we developed one[4].

[3] List of OCL tools: http://st.inf.tu-dresden.de/oclportal/index.php?option=com_content&view=category&id=8&Itemid=26

[4] The platform was developed using Django (http://www.djangoproject.com/), Dojo toolkit (http://dojotoolkit.org/) and MySQL database (http://www.mysql.com/).

Table 1: An overview of OCL tools in the web framework.

Dresden OCL	: http://www.dresden-ocl.org/
Owner	: Technische Universitaet Dresden, Germany
Licence	: LGPL
Current Version	: 3.1.0 (released on 2011-01-17)
Supported OCL Version(s):	OCL 2.2 / OCL 2.3
Information provider(s)	: Birgit Demuth, Claas Wilke
Eclipse OCL	: http://wiki.eclipse.org/MDT/OCL
Owner	: Eclipse Foundation
Licence	: EPL v1.0
Current Version	: 3.1.0 (released on 2011-06-22)
Supported OCL Version(s):	OCL 2.3
Information provider(s)	: Ed Willink
Oclarity	: http://www.empowertec.de/products/oclarity/
Owner	: EmPowerTec AG
Licence	: Commercial license, but free of charge for any use
Current Version	: 2.2 (released on 2010-11-22)
Supported OCL Version(s):	OCL 2.0
Information provider(s)	: Andreas Awenius
OCLE	: http://lci.cs.ubbcluj.ro/ocle/
Owner	: Babes-Bolyai University of Cluj-Napoca
Licence	: LGPL
Current Version	: 2.04 (released on 2005-07-15)
Supported OCL Version(s):	OCL 2.0
Information provider(s)	: Dan Chiorean
SQUAM OCL	: http://squam.info/
Owner	: University of Innsbruck, arctis GmbH
Licence	: dual: academic and commercial
Current Version	: 0.8.0 (released on 2010-10-18)
Supported OCL Version(s):	OCL 2.0 (MDT OCL Eclipse Galileo)
Information provider(s)	: Joanna Chimiak-Opoka, Hannes Moesl
TOPCASED VF	: http://gforge.enseeiht.fr/projects/topcased-vf/
Owner	: TOPCASED consortium
Licence	: EPL v1.0
Current Version	: 4.3.0 (released on 2011-02-07)
Supported OCL Version(s):	OCL 2.3 (tooling based on Eclipse MDT OCL)
Information provider(s)	: Sebastien Gabel
USE	: https://sourceforge.net/projects/useocl/
Owner	: University of Bremen, Database Systems Group
Licence	: GNU General Public License (GPL)
Current Version	: 2.6.2 (released on 2010-11-02)
Supported OCL Version(s):	OCL 2.2
Information provider(s)	: Lars Hamann

The web framework provides the following **content and functionality** (accessible from separate [tabs]):
[About] —description of the process and list of contributors,
[Features] —list and description of OCL tools registered in the system,
[OCL Tools] —list and description of all features (predefined, proposed, and additional),
[Feature Model] —presentation or gathering information about features in each tool (Fig. 3),
[Statistics] —overview illustrations of the data collected in the system,
[Overview Table]—a spread sheet with data for all tools with sorting and filtering functionality.

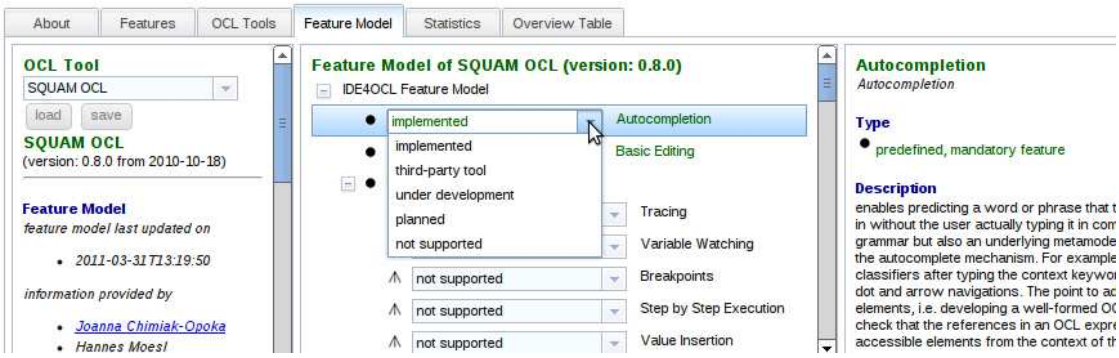

Figure 3: A partial screen shot of the data gathering view. From the left: tool description, feature model, and feature details.

In the *feature model* the following states of features are possible to select: *implemented*, *third–party tool*, *under development*, *planned*, *not supported*. We intentionally did not use *partially implemented* as it has too broad and fuzzy meaning. Data in the web platform was provided by the authors of this paper (registered users). This data is available to all visitors of the platform.

The web framework is intended both to be an aid for potential OCL users to choose the best tool for their concrete OCL use scenarios and to provide a comparative view to aid the development of OCL tools in the future. Our intention is to maintain the existing tool evaluations over the time as new releases of the OCL tools become available. Currently, data about seven OCL tools described in the next section is collected, but in the future we plan to include further OCL tools.

3 Selected OCL Tools

The subsequent sections give structured descriptions of the selected OCL tools (Table 1). Each tool description comprises a summary of its development history, a description of the tool(s) in the context of the OCL landscape, success stories and future development plans. In the illustrations of particular OCL tools we used stereotyped components (Fig. 2) relating to different components' roles[5].

[5] In the following we use for the *Formal Verification Tool* the acronym *FV Tool*.

3.1 Dresden OCL

DRESDEN OCL[6] is a mature toolkit widely used in teaching, research and practice. It supports the specification and evaluation of OCL constraints and queries, and can be used for several metamodels, on different metamodel layers and in several technical spaces [HDF02], [DW09], [WTW10].

In 1999, we started with the implementation of the standard library and a parser for OCL initially called DRESDEN OCL TOOLKIT. The idea was to provide an open-source third-party library of OCL tools that can be easily integrated with other modelling tools. The development of DRESDEN OCL now spans more than a decade and was mainly influenced by the progress in OCL research, the evolving OCL standard, and the evolution of the Model Driven Software Development tool landscape. Consequently, both the architecture of DRESDEN OCL and single tools in the toolkit were revised and re-engineered in an iterative process. From an architectural point of view we implemented three major releases. The commonality of the first and second release was that the specification and evaluation of OCL constraints was focused mainly on UML/MOF-based models. The third and current release (see Figure 4) accommodates the mainstream in model-driven development and is therefore adapted to Eclipse/EMF applications. It basically provides an EMFTEXT[7]-based OCL EDITOR (including an OCL PARSER) and an OCL INTERPRETER as parts of a future IDE4OCL as well as MDE Tools for Java code generation (OCL2JAVA), for Java code instrumentation (OCL2ASPECTJ) and for SQL code generation of database schemas and queries (OCL2SQL). User access to models is provided by the MODEL BUS.

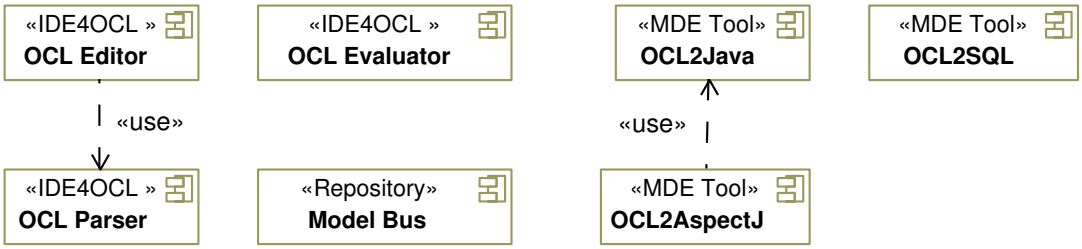

Figure 4: Dresden OCL tools.

DRESDEN OCL is integrated into several modelling tools such as ARGOUML and MAGICDRAW UML and has been used in many research projects [8].

We are currently integrating support for OCL refactorings into the OCL EDITOR. Furthermore, we have been experimenting with the integration of OCL into other languages [HJS+10]. In future, we plan to research debugging OCL expressions. We also plan scalability enhancement in OCL evaluation because case studies have shown that there are performance problems evaluating large OCL packages on large models and/or large collections of objects.

[6] http://www.dresden-ocl.org/
[7] http://www.emftext.org/
[8] http://www.dresden-ocl.org/index.php/DresdenOCL:SuccessStories

3.2 Eclipse OCL

The ECLIPSE OCL[9] project provides an implementation of the OMG OCL specification for use with Ecore and UML models on the Eclipse platform. The initial code contribution from IBM provided a Java API for parsing and evaluation for Ecore meta–models. Subsequent evolution has added support for UML meta–models, and an interactive evaluation console.

More recently, the Eclipse Xtext DSL tooling has been used to provide four different OCL editors (Fig. 5). The ESSENTIALOCL EDITOR provides minimal expression capability and is embedded as the input editor for the OCL CONSOLE which supports interactive evaluation of queries over models loaded elsewhere in Eclipse. The OCLINECORE EDITOR supports editing OCL embedded within an Ecore meta–model. The embedded OCL is executed when invariants are checked, operation bodies executed or property derivations evaluated. The COMPLETEOCL EDITOR supports the OMG syntax for independent OCL documents. Both OCLINECORE and COMPLETEOCL EDITORS may be used to define semantic validations for a DSL developed with Xtext. The OCLSTANDARDLIBRARY EDITOR maintains standard and custom library definitions. The IMPACT ANALYZER exploits the formality of OCL to optimize re–evaluation of OCL expressions over models in response to model element changes; 1000–fold improvements have been measured.

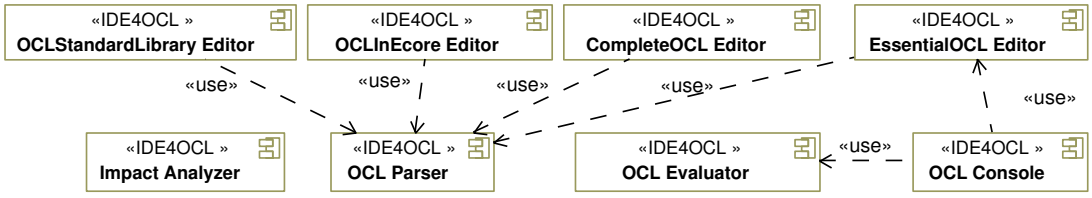

Figure 5: Eclipse OCL tools

ECLIPSE OCL is used by a variety of Eclipse projects[10] such as Connected Data Objects (CDO) to support server–side OCL queries on a model repository, by MODISCO for enhanced model browsing and by the Business Intelligence and Reporting Tools (BIRT) to support integration of model content in reports. The OCL PARSER was made extensible so that ACCELEO (MOFM2T), QVTc, QVTr and QVTo can exploit the OCL grammar and parsing. The personnel overlap between ECLIPSE OCL and the OMG OCL RTF has led to ECLIPSE OCLprototyping a candidate solution to many problems of ambiguity and under–specification in the OCL 2.3 specification. The Indigo (June 2011) ECLIPSE OCL release includes an extensible modeled OCL Standard Library [Wil11b] and a UML–derived intermediate pivot meta–model [Wil11a]. The pivot meta–model resolves significant issues in XMI persistence and useability of Complete OCL.

The new tooling is being used to fully model the OCL specification, with the multiple goals of debugging the specification, exploiting OCL in the auto–generation of Xtext–based tooling [Wil10] and defining new APIs that may be shared by alternative OCL implementations. OCL code generation is planned so that OCL embedded in Ecore meta–models can execute directly as Java rather than as interpreted OCL.

[9] Also known as ECLIPSE MDT/OCL, http://wiki.eclipse.org/MDT/OCL
[10] http://www.eclipse.org/projects/

3.3 OCLE

The Object Constraint Language Environment (OCLE[11]) is a tool meant to support MDE. OCLE enables developers to improve the quality of software applications by direct-engineering and design by contract. OCLE supports the construction of compilable UML models (complying with the UML WFRs) and their refinement by adding observers and OCL assertions. Following the validation of OCL specifications on appropriate model instantiations, the specifications in question can be translated into Java and injected within application code.

Initially designed within the NEPTUNE[12] project, OCLE was conceived to support validation of UML models against WFRs, including the specification, testing and improvement of rules defined at the metamodel level and the debugging of UML models in case of rule violation. Later on, it was extended to support OCL specifications at the user model level. Unlike all the other tools presented in this paper, OCLE's development was stopped in 2005; however, the tool continues to be used in both education and research (see [CBCS04, CPP08, PBO07, Wah08, Jam05, Woo05]). Below, we describe the main features distinguishing OCLE from other OCL tools.

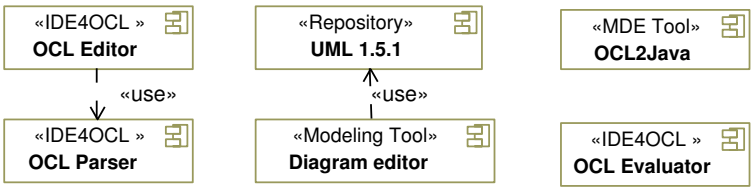

Figure 6: OCLE tools.

The metamodel is fully compliant with the UML 1.5.1 standard. The OCL 2.0 specification is almost entirely implemented, including some functionalities of the OCL 2.3 release, such as semantic closure. Models and their instantiations can be either constructed in an easy and intuitive manner using diagram editors, or can be imported by means of the XMI 1.0 or 1.1 standards. OCLE is a standalone tool, the only resource required being JRE or JDK 1.5, or a later version. OCLE enables an unrestricted access to the UML metamodel structure, Additional Operations and Well Formedness Rules. Extending the behavior of OCL types (including primitive types) can be easily accomplished, by specifying the new observers by means of a `def let` declaration in an "`*.ocl`" file, inside the namespace *package* `Foundation::Data_Type ...` *endpackage*. Reusing such extensions is easy: simply attach the corresponding "`*.ocl`" file(s) to the desired project. Therefore, OCLE supports the reuse of OCL specifications written at the metamodel level. The compilation and evaluation of literal OCL expressions does not require an active model. The Objects obtained as result of evaluating entire OCL specifications, or subexpressions only, can be navigated in an automated manner, from the editor containing the specification, to the Evaluation tab, model browser and corresponding diagrams. OCLE supports test-driven specification development. As future work, we plan to extend the OCL support to Model Driven Architecture-based languages, as well as to implement the new OCL 2.3 release.

[11] http://lci.cs.ubbcluj.ro/ocle

[12] the IST 1999-20017 FP5 EU research project, see http://neptune.irit.fr

3.4 SQUAM OCL

SQUAM OCL[13] stands for Systematic QUality Assessment of Models with OCL. It is a framework integrating heterogeneous modelling environments, and supporting model quality analysis with user–defined OCL libraries. Our mission is to make writing OCL expressions easy, and thus contribute to improvement of the pragmatics and increase of the usage scope of this language. As the aim of the framework is to enable user–friendly specification and evaluation of OCL expressions, it is convergent with the idea of an IDE4OCL.

Development of the current version of SQUAM OCL started at the University of Innsbruck at the end of 2007 and resulted in an advanced OCL editor [Ars08] based on Eclipse MDT OCL. The user–friendly editor enabled usage of user–defined OCL libraries, OCL unit tests and OCL documentation [CO09]. This part is available as the open source community edition (CE). Later, up to the end of 2010, the development of further components was continued together with our industrial partner, arctis[14], within the SoftNet competence network[15]. This part is available under academic / evaluation / commercial licence as the professional edition (PE).

Currently SQUAM OCL consists of two editions (CE, PE) and several plug–ins (Fig. 7). OCLEditor (CE) provides the front end functionality, with possibility to transform our library extension to the standard OCL syntax. OCLEvaluator (CE) enables pre–parsing and is used as a proxy to an OCL parser. It includes support of OCL libraries, OCL unit tests, OCL documentation [CO09] and integration of OCL back box extensions in Java or other languages. ModelingTool (PE) provides functionality to generate reports for sets of OCL queries, run and store them in a database. It additionally allows navigation from a model element to a query, and evaluation of this query. CheckStyle (PE) automatically runs queries and each time a predefined OCL expression is violated. The result of a query is listed in the problem view.

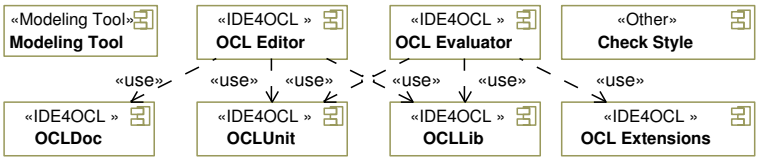

Figure 7: Main components of SQUAM OCL.

The SQUAM OCL approach was successfully used in teaching and research contexts. We used it to develop an OCL Standard Library course [CD10b], a set of UML class diagram metrics [CO09], quality [CAB10] and coverage criteria [FCZ+11] for domain specific languages defined as UML profiles. SQUAM OCL was also used with both Ecore and XML models.

Our short term goal is to integrate our OCLEvaluator with the Dreseden OCL parser. In the next version, the user should be able to select one of the two parsers. Another short term goal is to integrate our tool with a model repository tool developed within another project at our research group, and use OCL for evaluation of state diagrams and comparison of model versions.

[13] http://squam.info/

[14] arctis Softwaretechnologie GmbH, http://www.arctis.at/.

[15] Softnet Austria is a private research association cooperating with business and university partners to conduct and promote applied research in software engineering: http://www.soft-net.at/.

3.5 TOPCASED VF

TOPCASED[16] (Toolkit in OPen–source for Critical Application and SystEms Development) is a modular, open–source, Eclipse–based software environment providing methods and tools for critical embedded systems development. This platform relies on Eclipse Modeling Projects (EMP) to address concerns such as modelling, model transformation, requirements traceability, code generation, document generation and model verification [FGC+06].

Among all these issues, OCL is widely used by the TOPCASED Validation Framework[17] (VF) component to verify the consistency of industrial models at any time during the modelling phase. Developed between 2005 and 2009, TOPCASED VF offers a transparent integration with all TOPCASED modelers (UML, SysML, AADL, SAM,...). This set of plug–ins, exclusively based on ECLIPSE OCL, allows the user to write OCL rules, evaluate them on UML/MOF–based models and visualise the results in a dedicated human–machine interface.

The OCL EDITOR provides basic features such as syntax highlighting, content assist, problem/warning markers, comments and custom message support (Fig. 8). The OCL CHECKER evaluates OCL queries and stores the information inside a result model built on the fly. This result model is exploited through a multi–tab user interface where a distinction is made between check and metric rules. The verification results of these rules are reported into the Eclipse problem view. Corresponding decorators are displayed on graphical elements to help with identifying erroneous or incorrect elements. A double click on a problem marker sets the focus to the corresponding element represented in a diagram or in the outline. After any check operation, a textual or HTML report can be generated to keep track of the current state (OCL REPORTING). The OCL EVALUATOR can be considered as a tool assisting the end user to express complex OCL expressions. Custom message mechanisms may be used. They define a range of criticality levels to inform the user about violated constraints and can themselves contain OCL expressions. These interpreted messages are directly injected into the user interface where the results are presented.

Figure 8: The main tools included in TOPCASED VF.

The TOPCASED VF approach has been deployed and has been used in the context of large avionic development processes. It was notably the case in 2008 when SAM (Structured Analysis Model) [CGP09], a graphical language dedicated to functional division activities, was introduced in several pilot projects at Airbus[18] to support software specification. For this purpose, about eighty OCL rules have been defined to verify the static semantic of SAM models. Our future works will consist of offering the possibility to switch from an OCL library to another, as well as supporting other types of custom messages in order to enforce their usage.

[16] http://www.topcased.org/
[17] http://gforge.enseeiht.fr/projects/topcased-vf/
[18] http://www.airbus.com/

3.6 USE

The UML-based Specification Environment (USE)[19] supports developers in the early design stages to validate models specified in a subset of the UML [GBR07]. The first version of USE was published in 1998 as a Ph.D. project to provide an implementation of the formal OCL semantics. Since that, several extensions have been integrated into USE.

USE supports the definition, validation and runtime checking of OCL invariants, pre- and postconditions. USE employs a simple textual language to define models. Snapshots of models can be built manually by using a COMMAND LANGUAGE, semi-automatically with a built-in SNAPSHOT GENERATOR or by a Simple OCL-based Imperative Language (SOIL). The snapshot generator can be used to explore larger snapshots in an automatic way by specifying ASSL (A Snapshot Sequence Language) procedures which search for valid system states [GBR05]. It can be used to explore formal aspects like consistency of the specified model as described in [GKH09]. The created snapshots can be explored by evaluating OCL queries as well as in a visual way by combining OCL queries with diagrams, e. g., hiding or showing all instances returned by a query [GHXZ11]. The OCL PARSER performs a static check of OCL expressions when loading a model or when entering a query and reports encountered errors to the user, e. g., violations of type conformance. The OCL VALIDATOR can automatically check the validity of the defined STATIC constraints, (e. g., invariants) and dynamic constraints, (e. g., pre- and postconditions) after every change to the snapshots. Changes can be visualized by sequence diagrams. To debug OCL expressions USE provides an EVALUATION BROWSER which allows the user to examine the evaluation steps of an expression in detail. EXTENSIONS to the standard OCL types can be made by defining operations in Ruby. Specified constraints can be tested by running OCL UNIT TESTS against valid and invalid model instances similar to other xUnit frameworks [HG10].

Figure 9: USE tools

USE is successfully utilized in modelling courses at the University of Bremen and several national and international universities. As a library it is integrated into a German federal government software product called XGenerator as a validation and query engine. The XGenerator is used to transform profiled UML models into data exchange specifications and large parts of their documentation. A more detailed description of the XGenerator can be found in [BKG+08]. In the next release, USE will support more UML 2 features like relations between association ends, e. g., subsets and qualified associations. Further, the graphical part of USE is going to be improved. Future releases are going to provide a small but complete API to USE. This will make the integration of USE into applications as well as the development of plugins for USE easier.

[19] http://sourceforge.net/projects/useocl/

3.7 Oclarity

OCLARITY[20] is a standalone tool which is intended to provide an integrated authoring environment to learn OCL with little effort. It's focus is on ease of setup and use, not on a rich feature set. Oclarity only checks the validity of OCL expressions and provides no runtime support.

The first version of OCLARITY was implemented as an AddIn for RATIONAL ROSE, with the intent to create a commerical product. This version was released in 2003. After Rational was aquired by IBM we no longer saw a commercial perspective for OCLARITY and decided to make the existing functionality available as free software. During this step, we did a major redesign, to make OCLARITY independent of a specific modelling tool. This reworked version of OCLARITY was first released in 2008.

OCLARITY is a standalone program. It reads XMI files from supported modelling tools. Currently, ENTERPRISE ARCHITECT and MAGICDRAW are supported. OCLARITY cannot be integrated with other software nor does it integrate other OCL software.

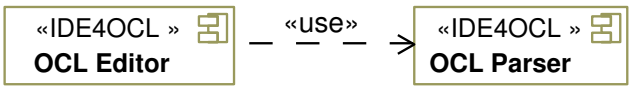

Figure 10: Oclarity tool

With OCLARITY we attempt to provide a faithful implementation of the OCL 2.0 standard. It is our intention to correct all reported errors concerning the language standard or other existing functionality. A migration to the OCL 2.3 standard might happen in the future, depending on interest and demand from the users. Otherwise, there are no plans to add new functionality to OCLARITY.

4 Data Analysis

Our intention is to collect data to enable users to select an OCL tool appropriate for their usage scenarios. The features in the feature model are of different granularities and their importance depends on the usage purpose. Because of this creating a general or quantitative measurement statistics suppresses important details and should not be used for tool comparison.

The validity of collected data is limited, as the information was provided directly by developers of the OCL tools, who may be less objective and critical than an average user. Moreover, there is no information collected about quality of implementation, tests and documentation. Thus no conclusion on maturity nor user–friendliness of tools can be drawn. Despite of these limitations, the collected data provide interesting insights related to architectures and available features of the presented OCL tools.

In this section we will give conclusions based on the tool descriptions included in this paper (Section 4.1) and the data collected in the web framework (Section 4.2).

[20] http://www.empowertec.de/products/oclarity/

4.1 Architectural Components

The main selection criteria for the presented tools was their potential to serve as an IDE4OCL, thus all of them have components providing this functionality (Fig. 11). The main purposes of the OCL tools are varied and can be illustrated by their **top level architectures**. They include components relating to different parts of the OCL landscape: OCLE, SQUAM and USE provide functionality of a *modelling tool*; USE additionally provides functionality for *formal verification*; and DRESDEN OCL and OCLE provide functionality of a *model driven engineering tool* and a *repository*.

OCL Tool	Core	Additional Components				
Dresden OCL	IDE4OCL	Modeling Tool	Repository	MDE Tool	FV Tool	Testing Tool
Eclipse OCL	IDE4OCL	Modeling Tool	Repository	MDE Tool	FV Tool	Testing Tool
Oclarity	IDE4OCL	Modeling Tool	Repository	MDE Tool	FV Tool	Testing Tool
OCLE	IDE4OCL	Modeling Tool	Repository	MDE Tool	FV Tool	Testing Tool
SQUAM	IDE4OCL	Modeling Tool	Repository	MDE Tool	FV Tool	Testing Tool
TOPCASED-VF	IDE4OCL	Modeling Tool	Repository	MDE Tool	FV Tool	Testing Tool
USE	IDE4OCL	Modeling Tool	Repository	MDE Tool	FV Tool	Testing Tool

Figure 11: Overview of OCL tool landscape coverage. Functionality of highlighted components (gray) is partially provided by particular tools.

Looking deeper into the **component architectures**, it is possible to see similarities and variations between the OCL tools and their way of supporting the IDE4OCL functionality (Table 2). Almost all tools have an OCL parser, an OCL evaluator and an OCL editor. All tools, except SQUAM and TOPCASED–VF, have their own OCL parsers. SQUAM and TOPCASED–VF are based on ECLIPSE OCL. All tools, except OCLARITY, have an OCL evaluator. And all tools, except USE have an own OCL editor. The OCL tools differ in components improving and extending OCL. ECLIPSE OCL provides an impact analyser to optimise re–evaluation of OCL expressions. SQUAM provides support of OCL Lib, OCL Unit, and OCL Doc [CO09] and black box extensions in Java or other languages. TOPCASED–VF provides OCL Checker and OCL Reporting to store OCL evaluation results inside a result model built on the fly and to generate a HTML report. USE provides OCL Unit and OCL Extensions in Ruby. The support of OCL unit tests in SQUAM OCL and USE differs as each tool implements own extensions for OCL [CO09, HG10].

Although it was not mentioned in the descriptions of the tools it is interesting to notice the **technical similarities** of the OCL tools. Almost all tools are written in Java, only OCLARITY uses the .NET framework. Implementations of DRESDEN OCL, ECLIPSE OCL, SQUAM OCL and TOPCASED VF use the EMF that enables to integrate various Eclipse tools related to modelling as *third–party tools*.

Table 2: OCL tools and their native components with functionality of an IDE4OCL.

	Dresden OCL	Eclipse OCL	Oclarity	OCLE	SQUAM OCL	TOPCASED-VF	USE
OCL Parser	+	+	+	+			+
OCL Evaluator	+	+		+			+
OCL Editor	+	+	+	+	+	+	
Impact Analyser		+					
OCL Libraries					+		
OCL Doc					+		
OCL Extensions					+		+
OCL Checker						+	
OCL Reporting						+	

4.2 Available Features

We have analysed the availability of features in the presented OCL tools. Below we present only the insights interesting from our point of view, i.e. we focus on the feature coverage. The full list of available features and other statistics can be found on–line at http://ide4ocl.opoki.com/.

There are four features which are implemented **in all presented tools**: basic editing, document interface, syntax highlighting, and syntax compliance according different OCL specification versions. It is partially meets the main needs of OCL users, as syntax highlighting and basic editing were identified as the most important features [CD10a]. The remaining most wanted features are auto–completion, debugging and refactoring support. The auto–completion feature is implemented in four of the tools, under development in one and planned in another. Debugging is complex, has many sub–features and is a rather new feature in the context of OCL. Several tools support single sub–features of debugging.

We are also interested what features are **unique to single tools**[21]. DRESDEN OCL is the only tool that comes with an MDE tool supporting the model transformation feature. Its OCL2SQL tool not only generates SQL code evaluating OCL expressions but also transforms UML models to relational views (called *object views* [DHL01]). ECLIPSE OCL provides a hybrid OCL/MOF view. SQUAM OCL is the only tool providing generation of documentation out of OCL Doc comments [CO09]. This feature is under development in ECLIPSE OCL. And SQUAM OCL uniquely provides statement coverage support related to the usage of OCL definitions in OCL queries and OCL unit tests [CO09]. Several debugging issues are implemented only in USE (variable watching, value insertion, automating test cases), but they are planned in DRESDEN OCL, ECLIPSE OCL and OCLE. USE uniquely provides modularisation of language, but it will probably soon change, as is currently under development in DRESDEN OCL [WTZ10] and ECLIPSE OCL.

To recognise **emerging trends** it is interesting to have a look at features not implemented yet, but under development: Further and extensive refactoring techniques are under development in DRESDEN OCL. Standardisation related features, visibility and lexical scoping and XMI compliance, are under development in ECLIPSE OCL. And USE will be extended by a testing tool.

[21] Tools are presented in the alphabetical order.

5 Conclusion

This report is the third part in a series of preceding papers [CDSR09, CD10a] analysing the requirements of OCL users for an ideal Integrated Development Environment for OCL and the state of existing OCL tools. We refined and documented the IDE4OCL feature model within a web framework that is available online for the public, and which is extensible for further OCL tools. We then used the web framework to collect data from seven well–known OCL tools (DRESDEN OCL, ECLIPSE OCL, OCLARITY, OCLE, SQUAM OCL, TOPCASED VF, USE) describing their supported features according to the IDE4OCL feature model. Each of these tools is briefly textually described. Furthermore, the architectural components of each OCL tool are classified into the OCL tools landscape. This is surely a simplified view of the heterogenity of existing tools but provides a comparative view of the selected OCL tools. Both the analysis of the architectural components and of the features provide together a first comparative and quantitative evaluation of existing OCL tools.

The question for a qualitative comparison is a critical issue and could be a subject for further work. Firstly it should be noted again that all data were provided by the OCL tool developers themselves. This ensures a deep insight into the tool, but may obstruct an objective view of how effective a tool is. Furthermore, we noticed during the data collection that a few of our applied feature evaluation values (*implemented*, *third-party tool*, *under development*, *planned*, *not supported*) could be misunderstood and insufficient. Potentially in a next step, we should extend the evaluation values by *partially implemented* and *fully tested*. However, this would need first a solid discussion about the semantics of these values.

All in all, we are sure that the results of this OCL tools report both help potential OCL users to choose a suitable OCL tool, and highlight emerging trends in OCL tool development that are especially helpful for tool developers. In this sense, feedback and suggestions concerning all OCL tools are welcome. Please send emails to the authors/OCL tool developers if you want to get in touch with them.

Last but not least it should be emphasized that it is still our goal to promote and discuss possibilities of cooperation to drive the further development of OCL tool support that is suitable in any (meta-)modelling environments and languages, and for advanced industrial applications.

Acknowledgement We would like to express our gratitude to contribution of Darius Silingas and Nicolas F. Rouquette as co–authors of our first paper related to the idea of IDE4OCL [CDSR09]. We want to express our thanks to students and Ph.D. students at TU Dresden and University of Innsbruck for their support from the initial phase of this project: Claas Wilke, Michael Thiele and Hannes Moesl, Ekrem Arslan, Cornelia Haisjackl. We are also thankful for the encouraging discussion with Robert France we had at MODELS 2010. We would like to thank Jim Arlow for his help in the finalisation of the paper. The work of Dan Chiorean was supported by Romanian CNCSIS-UEFISCSU, project number PNII IDEI 2049/2008.

Bibliography

[Ars08] E. Arslan. Advanced OCL Editor. B.Sc. thesis, University of Innsbruck, 2008.

[BKG+08] F. Büttner, M. Kuhlmann, M. Gogolla, J. Dietrich, F. Steimke, A. Pankratz, A. Stosiek, A. Salomon. MDA Employed in a Joint eGovernment Strategy: An Experience Report. In Bailey (ed.), *Proc. 3rd ECMDA Workshop "From Code Centric To Model Centric Software Engineering" (2008)*. European Software Institute, `http://www.esi.es/modelplex/c2m/program.php`, 2008.

[CAB10] J. Chimiak-Opoka, B. Agreiter, R. Breu. Bringing Models into Practice: Design and Usage of UML Profiles and OCL Queries in a showcase. In Targamadze et al. (eds.), *Proc. of the 16th Int. Conf. on Information and Software Technologies, IT'2010*. Pp. 265–273. Technologija, Kaunas, Lithuania, April 2010.

[CBCS04] D. Chiorean, M. Borteş, D. Coruţiu, R. Sparleanu. UML/OCL tools - Objectives, Requirements, State of the Art - The OCLE Experience. In *Proceedings of 11th Nordic Workshop on Programming and Software Development Tools and Techniques*. Pp. 163–180. 2004.

[CD10a] J. Chimiak-Opoka, B. Demuth. A Feature Model for an IDE4OCL. *Electronic Communications of the EASST: OCL and Textual Modelling* 36, 2010. (presented at OCL Workshop).

[CD10b] J. Chimiak-Opoka, B. Demuth. Teaching OCL Standard Library: First Part of an OCL 2.x Course. 2010. (presented at EduSymp, to appear in Electronic Communications of the EASST).

[CDSR09] J. Chimiak-Opoka, B. Demuth, D. Silingas, N. F. Rouquette. Requirements Analysis for an Integrated OCL Development Environment. *Electronic Communications of the EASST: The Pragmatics of OCL and Other Textual Specification Languages 2009* 24, 2009. (presented at OCL Workshop).

[CGP09] A. Canals, S. Gabel, G. P. The SAM and OCL components of the TOPCASED project. In *Neptune'2009*. Paris, France, May 2009.

[CO09] J. Chimiak-Opoka. OCLLib, OCLUnit, OCLDoc: Pragmatic Extensions for the Object Constraint Language. In Schürr and Selic (eds.), *MoDELS*. LNCS 5795, pp. 665–669. Springer, 2009.

[CPP08] D. Chiorean, V. Petraşcu, D. Petraşcu. How My Favourite Tool Supporting OCL Must Look Like. *ECEASST* 15:1–17, 2008.

[DHL01] B. Demuth, H. Hußmann, S. Loecher. OCL as a Specification Language for Business Rules in Database Applications. *UML 2001-The Unified Modeling Language. Modeling Languages, Concepts, and Tools*, 2001.

[DW09] B. Demuth, C. Wilke. Model and Object Verification by Using Dresden OCL. In *Proceedings of the Russian-German Workshop Innovation Information Technologies: Theory and Practice, Ufa, Russia, July 25-31, 2009*. P. 81. Ufa State Aviation Technical University, Ufa, Bashkortostan, Russia, July 2009.

[FCZ+11] M. Felderer, J. Chimiak-Opoka, P. Zech, C. Haisjackl, F. Fiedler, R. Breu. Model Validation in a Tool–based Methodology for System Testing of Service–oriented Systems. *International Journal On Advances in Software*, May 2011. (to appear).

[FGC+06] P. Farail, P. Gaufillet, A. Canals, C. Le Camus, D. Sciamma, P. Michel, X. Crégut, M. Pantel. The TOPCASED project: a Toolkit in OPen-source for Critical Application and SystEms Development. In *European Congress on Embedded Real-Time Software*. Toulouse, France, May 2006.

[GBR05] M. Gogolla, J. Bohling, M. Richters. Validating UML and OCL Models in USE by Automatic Snapshot Generation. *Journal on Software and System Modeling* 4(4):386–398, 2005.

[GBR07] M. Gogolla, F. Büttner, M. Richters. USE: A UML-Based Specification Environment for Validating UML and OCL. *Science of Computer Programming* 69:27–34, 2007.

[GHXZ11] M. Gogolla, L. Hamann, J. Xu, J. Zhang. Exploring (Meta-)Model Snapshots by Combining Visual and Textual Techniques. In *Proc. 10th Int. Workshop on Graph Transformation and Visual Modeling Techniques (GT-VMT'2011)*. 2011.

[GKH09] M. Gogolla, M. Kuhlmann, L. Hamann. Consistency, Independence and Consequences in UML and OCL Models. In Dubois (ed.), *Proc. 3rd Int. Conf. Test and Proof (TAP'2009)*. Pp. 90–104. Springer, Berlin, LNCS 5668, 2009.

[HDF02] H. Hussmann, B. Demuth, F. Finger. Modular architecture for a toolset supporting OCL. *Sci. Comput. Program.* 44:51–69, July 2002.

[HG10] L. Hamann, M. Gogolla. Improving Model Quality by Validating Constraints with Model Unit Tests. In *Proc. 7th Int. Workshop on Model-Driven Engineering, Verification, and Validation (MODEVVA'2010)*. 2010.

[HJS+10] F. Heidenreich, J. Johannes, M. Seifert, M. Thiele, C. Wende, C. Wilke. Integrating OCL and Textual Modelling Languages. In *Proc. of Workshop on OCL and Textual Modelling (OCL2010)*. 2010.

[Jam05] R. James. HSBC - Software Practice Advancement - Architecture Day - EAI via MDA. 2005.
http://www.bcs-oops.org.uk/resources/mdaday/James-EAIViaMDA.pdf

[PBO07] R. Paige, P. Brooke, J. Ostroff. Metamodel-based Model Conformance and Multi-View Consistency Checking. *ACM Transactions on Software Engineering and Methodology* 13(3):13–61, 2007.
doi:10.1145/1243987.1243989

[Wah08] M. Wahler. *Model-Driven Software Development: Integrating Quality Assurance*. Chapter A Pattern Approach to Increasing the Maturity Level of Class Models, pp. 204–235. Idea Group Inc., 2008.
doi:10.4018/978-1-60566-006-6.ch009
http://kuznyechik.googlepages.com/wahler-maturity-2008draft.pdf

[Wil10] E. D. Willink. *Re-engineering Eclipse MDT/OCL for Xtext*. MODELS 2010, OCL Workshop, 2010.

[Wil11a] E. D. Willink. *Aligning OCL with UML*. TOOLS 2011, OCL Workshop, 2011.

[Wil11b] E. D. Willink. *Modeling the OCL Standard Library*. TOOLS 2011, OCL 2011 Workshop, 2011.

[Woo05] E. Woods. OCL Quick Reference. 2005.
http://www.artechra.com/media/writing/OCL1.5-quick-reference.pdf

[WTW10] C. Wilke, M. Thiele, C. Wende. Extending Variability for OCL Interpretation. In Petriu et al. (eds.), *Model Driven Engineering Languages and Systems*. Lecture Notes in Computer Science 6394, pp. 361–375. Springer Berlin / Heidelberg, 2010.

[WTZ10] C. Wende, N. Thieme, S. Zschaler. A Role-based Approach Towards Modular Language Engineering. *2nd International Conference on Software Language Engineering, (SLE 2009), Revised Selected Papers*, 2010.