A SWOT Analysis of the Object Constraint Language

Jordi Cabot[1], Daniel Calegari[2], Robert Clarisó[3], Martin Gogolla[4], Antonio Vallecillo[5] and Edward D. Willink[6]

[1]*ICREA, Barcelona, Spain*
[2]*Instituto de Computación, Facultad de Ingeniería, Universidad de la República, Uruguay*
[3]*Universitat Oberta de Catalunya, Rambla del Poblenou 156, 08018 Barcelona, Spain*
[4]*University of Bremen, Germany*
[5]*ITIS Software, Universidad de Málaga, Spain*
[6]*Willink Transformations Ltd, Reading, England*

Abstract
The Object Constraint Language (OCL) is a textual language to describe constraints or queries over software models defined by applying the Unified Modeling Language (UML). Using OCL, it is possible to specify, for instance, integrity constraints in the form of class invariants, operation pre-conditions, or post-conditions. This paper presents a *SWOT analysis* (Strengths, Weaknesses, Opportunities, and Threats) of OCL, considering both its current state and future perspectives: specification, tool support, applications, competitors, and the community around it. This analysis was the result of the panel discussion at the *20th International Workshop on OCL and Textual Modeling (OCL'21).*

Keywords
UML, OCL, SWOT analysis

1. Introduction

The Object Constraint Language (OCL) [1, 2] is a textual modeling notation defined as an OMG standard (currently in its version 2.4). OCL is a complement to the graphical models in the Unified Modeling Language (UML), augmenting UML and other types of models to define complex expressions, queries, and constraints over software models.

To discuss recent findings related to OCL, the community organizes a yearly meeting, the "International Workshop on OCL and textual modeling". Previous editions of the OCL workshop have included panels where the discussion has focused on potential improvements of the OCL, either in its syntax, semantics, standard library, or tools. Some of these panel discussions have been published in the workshop proceedings, *e.g.*, from the OCL workshops that took place in 2014 [3], 2016 [4], 2017 [5] and 2019 [6].

OCL'2021: 20th International Workshop on OCL and Textual Modeling, June 25, 2021, Virtual Event
jordi.cabot@icrea.cat (J. Cabot); dcalegar@fing.edu.uy (D. Calegari); rclariso@uoc.edu (R. Clarisó); gogolla@uni-bremen.de (M. Gogolla); av@uma.es (A. Vallecillo); ed at willink.me.uk ((E. D. Willink)
https://jordicabot.com/ (J. Cabot); https://www.fing.edu.uy/~dcalegar/ (D. Calegari); https://robertclariso.github.io (R. Clarisó); http://www.db.informatik.uni-bremen.de/~gogolla/ (M. Gogolla); http://www.lcc.uma.es/~av/ (A. Vallecillo)
0000-0003-2418-2489 (J. Cabot); 0000-0001-9639-0186 (R. Clarisó); 0000-0003-4311-1117 (M. Gogolla); 0000-0002-8139-9986 (A. Vallecillo)

Table 1
Characterization of the four dimensions in a SWOT analysis

	Positive	**Negative**
Internal	Strengths	Weaknesses
External	Opportunities	Threats

In the OCL'21 workshop, the panel focused on analyzing the current state of the OCL, the tool ecosystem, and the community. This analysis used the *SWOT matrix* framework, a strategic planning technique where the status of a given initiative is studied from four different perspectives: Strengths, Weaknesses, Opportunities, and Threats. Each SWOT perspective can be classified according to two axes, as displayed in Table 1:

- Factors that are *internal* (inherent to the initiative) versus factors that are *external* (environment, competitors, ...).
- *Positive* factors (that represent an advantage to the initiative) versus *negative* factors (detrimental to the initiative).

2. SWOT analysis

In the following, we describe the result of the SWOT analysis. Each SWOT dimension is discussed in a different subsection. Then, a final subsection discusses controversial topics, *i.e.*, aspects that can be considered both an advantage or a disadvantage.

2.1. Strengths

- OCL provides a useful and (mostly) well-defined notation for expressing constraints and Boolean expressions on UML models. Besides, OCL is easy to use, *e.g.*, the core concept is the navigation through a class structure via a simple dot notation.
- OCL applies to many different tasks in the context of software modeling: model querying, code generation, testing, verification and validation, model transformation, ...
- OCL is a "technology-independent" language for defining constraints. Thus, it can be used in a wide variety of projects using different technology stacks.
- OCL is very close to First-Order Logic (FOL). It provides an underlying formal semantics and makes it more amenable to computer scientists or developers with a background in logic. Furthermore, it is possible to adapt and/or reuse tools and solvers for FOL to analyze OCL specifications. For instance, we can benefit from efficient SAT-based solvers through a reduction to finite structures.
- Moreover, OCL has an important property: it is a *side-effect free* language, *i.e.*, evaluating an OCL constraint does not alter the model in any way. It facilitates a high-quality robust analysis.
- OCL counts on three complete implementations: USE [7], Eclipse OCL [8] and Acceleo [9]. OCL dialects, such as EOL [10], also help to experiment with new features and mechanisms.

- OCL is the best available language of its kind. There is a lack of suitable general-purpose alternatives. Paraphrasing Bran Selic's comment regarding UML: "OCL is the worst modeling language, except for all the others."

2.2. Weaknesses

- *Tool ecosystem*
  - Implementations of OCL could be improved: some OCL tools are incomplete or incompatible with the standard.
  - Many existing tools are research prototypes developed by academics. It means that interoperability, documentation, and support are lacking, hampering industrial or realistic use.
  - Even though a few tools exist (*e.g.*, verification and validation tools or a Java code generator in Eclipse OCL), there is a lack of a complete ecosystem of tools that use OCL as input to provide valuable services. It would provide value to dedicating time to formalize the OCL expressions.
- *Complexity of the OCL language*: The OCL 2.4 specification is 262 pages long and offers some features that are complex to understand and/or implement.
  - Multivalued Boolean expressions (true, false, null, invalid) increase the complexity of evaluating properties. For instance, special values need to be taken into account when considering short-circuits.
  - OCL features a complicated type system. In particular, navigation through a class structure is easy, but understanding the type of the results at the end of the navigation chain can be hard due to complicated typing rules.
  - OCL includes a complex hierarchy of collection types. The choice of operations and the type hierarchy among them sometimes creates confusion, *e.g.*, OrderedSet is not a subclass of Set.
  - OCL sometimes seems to offer too much expressive power, as there are many ways to write the same constraint. This is good for people learning and writing OCL expressions, but bad for building tools as the tools need to cope with all these potential variations.
- *Shortcomings of the OCL*: Although OCL includes almost all desirable features, it still presents some limitations and unresolved issues. For example:
  - There are no modularity mechanisms, which could help in the definition of large specifications.
  - There are no constructs for importing or reusing other specifications completely or partially: each OCL specification should be complete on its own.
  - As a consequence of the previous shortcomings, there are no reusable OCL libraries besides the OCL standard library. In particular, there is no library of standard mathematical functions.
- There is a strong dichotomy between OCL's "divine" and "material" nature, which are impossible to reconcile.

  - Under its “divine” nature in the Heaven of logic, it works under the umbrella of first-order logic as a side-effect free language, with commutative logical operators, and where datatypes are unbounded and perfect (*e.g.*, no overflow or rounding issues).
  - Under its “material” nature, which appears when used down in Earth as part of the realization of modeling languages, some issues need to be considered: short-circuit evaluation of expressions under the presence of errors or float precision and rounding.
  - Frustration occurs when OCL is used in one world with the nature of the other.
- *Community*
  - The team that supports OCL is very small. Despite being a “community” project, it basically rests on the shoulders of one single person. And it has been like this for a while with no perspective for a short-term improvement.
  - The community behind the language is also small. Moreover, it consists mostly of academic researchers with little industrial backing that could help popularize the language or provide more resources to support it.

2.3. Opportunities

- OCL enables defining incompletely specified software systems, which is useful for prototyping and exploring alternatives. It is possible to abstract implementations with class structures and OCL expressions, *e.g.*, defining (partial) class invariants and (partial) operation contracts that leave some parts of the behavior unspecified.
- Some aspects that have been identified as a weakness in the OCL specification (*e.g.*, the lack of libraries and import mechanisms or the lack of a library of standard mathematical operations) are also an opportunity to improve the language.
- *Verification and validation*: Currently, not all the potential reasoning and analysis capabilities are being exploited.
  - OCL could be used as the basis for much further analysis and automation tasks that could, for instance, provide the better model analysis (validation, verification, testing, instantiation).
  - OCL could benefit from improved tool support for interactive runtime analysis and verification, *e.g.*, tools that interactively suggest candidate fixes for modeling faults or that let users test alternative constraints.
- *Synergies with other modeling languages*
  - Software development is mostly textual-based, which generates a favorable ecosystem to all textual languages, like OCL.
  - OCL could be used as a modular add-on to other modeling languages and DSLs. The experience with ATL and QVT are two success stories. OCL saves users the need to learn yet another constraint language, and its functional nature makes it suitable for easy integration in many contexts.

  - In particular, we could explore other rule-based languages (e.g., RuleML, Web semantics) and communities to find potential synergies.
  - Another alternative is exploring SysML as a potential area of collaboration. The systems community seems to be more open to modeling than some subcommunities within the software engineering domain.
  - As a query language, it may have good opportunities to be used in developing domains where some extensions might make sense from a semantic point of view to specify constraints or query models. Examples of these domains include stream processing and big data, which need to deal with unbounded models.

2.4. Threats

- Tools are fading rather than strengthening: several valuable tools are no longer under active development, and others are no longer being maintained. Moreover, there is a lack of organizations that maintain and mature the tools.
- Slow (or none) adoption of open source projects.
- The level of abstraction of programming languages is rising, lessening the value of languages like OCL. While this is a threat for OCL, it is probably a positive side-effect of the existence of OCL.
- Few UML modelers know or use OCL and do not acknowledge the need for OCL. Rigor and formality are either explicitly neglected (when sketching models) or even realized in Java.
- Many users still do not understand the two distinct (and irreconcilable) natures of OCL. Namely, OCL is a *side-effect free* language, not to be used for programming. Still, many people ask for mechanisms that allow creating object instances. The identity of OCL and how to use and implement it needs to be better clarified. Otherwise, this "paranoia" can kill OCL.
- Related to the previous misconception and considering that OCL is used in various contexts (query language, verification, ...), maybe we are asking too much from OCL: it is perhaps as good as it should be, but no more.
- Related approaches such as Alloy [11] (based on relational logic) offer powerful analysis capabilities, both in terms of verification and validation.

2.5. Controversial topics

- OCL is unique in the sense that it includes a 4-valued logic [1]: besides true and false, the values *null* and *invalid* are also supported. Nevertheless, it is unclear whether having a 4-valued logic is a competitive advantage or a disadvantage versus other modeling languages. On the one hand, it allows the precise specification of different types of situations, *e.g.*, distinguishing the lack of information from an error. On the other hand, it is not intuitive for newcomers to the language, and some tools do not properly support it in the OCL ecosystem.
- OCL is not a language on its own, but fundamentally an add-on language that can be embedded into a modeling language (*e.g.*, UML) in which it provides navigability, query,

Table 2
Summary of the results of the SWOT analysis

Strengths	Expressive, usable and convenient query & constraint language for object-oriented software development
	Availability of several implementations
	Formal foundation based on first-order logic
	Navigation as a useful metaphor
	Lack of better alternatives
Weaknesses	Expressiveness brings complexity in some aspects of the language (navigation, types, collections, 4-valued logic): this hurts tool development
	Lack of *industrial* tool support
	Lack of desirable features (libraries, importing, modularity) in the language
	Lack of a complete tool ecosystem that adds value to existing OCL constraints
Opportunities	OCL supports describing incomplete specified systems
	Room for improvement in future versions of the language
	Improved verification and validation (*e.g.*, interactive analysis)
	Potential synergies with other languages (SysML, RuleML, business rules)
Threats	Lack of organisations involved in OCL tool development and maintenance
	Lack of awareness about the need for OCL in UML
	Misconceptions about the usage scenarios for OCL
	Strong competition from Alloy on the verification & validation side

and constraint specification features to other modeling languages. It allows OCL to add value to the extended modeling languages and makes OCL a reusable notation with well-grounded semantics. On the contrary, not being a stand-alone language hinders its wide adoption — particularly when OCL does not provide mechanisms to be incorporated into other modeling languages in a modular and reusable manner. Note that with so many textual notations of the UML, it should be somehow feasible to easily combine UML and OCL expressions in a single textual model.

3. Conclusions

This paper has presented a discussion of the current state of OCL and its future trends using the SWOT analysis framework. Table 2 provides a summary of the findings.

These results indicate relevant opportunities for future research and collaborations concerning OCL and issues that the OCL community should address.

Acknowledgments

The authors would like to thank the attendees of the OCL workshop for the interesting discussions during the panel.

References

[1] OMG, Object constraint language (OCL) version 2.4, 2014. URL: https://www.omg.org/spec/OCL/.

[2] J. Cabot, M. Gogolla, Object constraint language (OCL): A definitive guide, in: M. Bernardo, V. Cortellessa, A. Pierantonio (Eds.), Formal Methods for Model-Driven Engineering – 12th International School on Formal Methods for the Design of Computer, Communication, and Software Systems, volume 7320 of *LNCS*, Springer, 2012, pp. 58–90. doi:10.1007/978-3-642-30982-3_3.

[3] A. D. Brucker, T. Clark, C. Dania, G. Georg, M. Gogolla, F. Jouault, E. Teniente, B. Wolff, Panel discussion: Proposals for improving OCL, in: A. D. Brucker, C. Dania, G. Georg, M. Gogolla (Eds.), Proc. of the 14th International Workshop on OCL and Textual Modelling, co-located with MODELS 2014, Valencia, Spain, September 30, 2014, volume 1285 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2014, pp. 83–99. URL: http://ceur-ws.org/Vol-1285/paper09.pdf.

[4] A. D. Brucker, J. Cabot, G. Daniel, M. Gogolla, A. S. Herrera, F. Hilken, F. Tuong, E. D. Willink, B. Wolff, Recent developments in OCL and textual modelling, in: A. D. Brucker, J. Cabot, A. S. Herrera (Eds.), Proc. of the 16th International Workshop on OCL and Textual Modelling, co-located with MODELS 2016, Saint-Malo, France, October 2, 2016, volume 1756 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2016, pp. 157–165. URL: http://ceur-ws.org/Vol-1756/paper12.pdf.

[5] R. Bill, A. D. Brucker, J. Cabot, M. Gogolla, A. Vallecillo, E. D. Willink, Workshop in OCL and textual modelling – report on recent trends and panel discussions, in: M. Seidl, S. Zschaler (Eds.), Proc. of STAF 2017 Collocated Workshops, Marburg, Germany, July 17-21, 2017, volume 10748 of *LNCS*, Springer, 2017, pp. 297–301. doi:10.1007/978-3-319-74730-9_26.

[6] A. D. Brucker, G. Daniel, M. Gogolla, F. Jouault, C. Ponsard, V. Ramon, E. D. Willink, Emerging topics in textual modelling, in: A. D. Brucker, G. Daniel, F. Jouault (Eds.), Proc. of the 19th International Workshop in OCL and Textual Modeling (OCL 2019) co-located with MODELS 2019, Munich, Germany, September 16, 2019, volume 2513 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2019, pp. 91–104. URL: http://ceur-ws.org/Vol-2513/paper8.pdf.

[7] M. Gogolla, F. Büttner, M. Richters, USE: A UML-based specification environment for validating UML and OCL, Sci. Comput. Program. 69 (2007) 27–34. doi:10.1016/j.scico.2007.01.013.

[8] Eclipse Foundation, Eclipse OCL (Object Constraint Language), 2021. URL: https://projects.eclipse.org/projects/modeling.mdt.ocl.

[9] Obeo, Acceleo OCL, 2021. URL: https://wiki.eclipse.org/Acceleo/OCL.

[10] D. S. Kolovos, R. F. Paige, F. Polack, Epsilon Object Language (EOL), 2021. URL: https://www.eclipse.org/epsilon/doc/eol/.

[11] D. Jackson, Software abstractions: logic, language, and analysis, MIT press, 2012.