Should Future Variability Modeling Languages Express Constraints in OCL?

Don Batory
University of Texas at Austin
batory@cs.utexas.edu

ABSTRACT
Since the mid-2000s, Propositional Logic (FL) has been the de facto language to express constraints in Feature Models (FMs) of Software Product Line (SPLs). FL was adequate because product configurations were formed by binary decisions including or not including features in a product. Inspired by both prior research and practical systems (e.g., SPLs that use KConfFig), future FMs must go beyond FL and admit numerical (and maybe even text) variables and their constraints.

The Object Constraint Language (OCL) is a general-purpose declarative constraint language for Model Driven Engineering (MDE), which admits virtually any kind of variable and constraint in metamodels. We should expect future FMs to be examples of MDE metamodels. This raises a basic question: Should OCL be used to express constraints of future variability modeling language(s)?

In this talk, I outline the pros and cons for doing so.

1 INTRODUCTION
I am no fan of the Object Constraint Language (OCL) and never have been. I find it inelegant and bloated. Using Eclipse OCL years ago, I recall different OCL implementations on different syntaxes and covered the OCL standard (at that time) with varying fidelity. (Note: The current 2.0 standard is a behemoth 262 page document [5]).

I tried to teach OCL to my undergrads, and that was an unpleasant experience for both me and my students. OCL and related tools were simply too complicated. As mentioned in [1], Eclipse tools:

1. Were unappealing—they were difficult to use even for simple applications.
2. Fostered a medieval mentality in students to use incantations to solve problems. Point here, click that, something happens.
3. Have a steep entry cost to use, teach, and learn—too high for my comfort (and I suspect my student’s as well).

I’m not alone with these opinions [2].

From a distance, I have also watched various attempts to generalize Feature Models (FMs) to address next generation Software Product Line (SPL) concerns such as admitting replicated features, features with attributes, numerical features, and expressing constraints. I recoiled at the complexity of these attempts, and the use of OCL as the language to express constraints.

I do not profess to know what future SPL Variability Modeling Languages will be and how constraints in such languages will be expressed. But I do believe the answer will be guided by:

- **Simplicity!** FL was chosen for classical FM constraints because it was a simple mathematical standard. I’m not sure there is a formal grammar (language) to which all classical FM tools agree, but it is hard to screw-up writing FL constraints. Next-generation FM constraints should be equally straightforward to write.

- **Don’t Invent, Reuse!** Do we really need a new constraint language for future FMs? Clearly we need more than FL. But are we good enough as language engineers to create a new constraint language without making a complete mess of it? Shouldn’t we reuse existing languages or sub-languages of existing languages? Our expertise is in SPLs, not in language engineering. If you want an example of (IMO) a failed custom constraint language, it is OCL. Re-read the 2nd sentence of this Introduction.

- **Circularity Avoidance!** Generalizing beyond hierarchical relationships of classical FMs, we’re not far away from UML class diagrams [4] and Model Driven Engineering metamodels [3] – which are class diagrams + constraints. And constraints for class diagrams beg the use of OCL.

In this talk, I offer and demonstrate a way out of this circular conundrum. My solution does not eliminate all problems, but it does diminish key problems about OCL standards, OCL tooling, reducing the need for yet-another-language, minimizing long-term tool maintenance, and keeping constraint languages both familiar and simple to SPL programmers, practitioners, and researchers.

Acknowledgments. Batory is supported by NSF grant CCF1212683.

REFERENCES

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

SPLC ’19, September 9–13, 2019, Paris, France
© 2019 Association for Computing Machinery.
ACM ISBN 978-1-4503-6668-7/19/09... $15.00
https://doi.org/10.1145/3307630.3342406