Variability Representations in Class Models: An Empirical Assessment (Summary)

Daniel Strüber[1], Anthony Anjorin[2], Thorsten Berger[3]

Abstract: We present our paper originally published in the proceedings of the ACM/IEEE International Conference on Model Driven Engineering Languages and Systems 2020 (MODELS).
Owing to the ever-growing need for customization, software systems often exist in many different variants. To avoid the need to maintain many different copies of the same model, developers of modeling languages and tools have recently started to provide representations for such variant-rich systems, notably variability mechanisms that support the implementation of differences between model variants. Available mechanisms either follow the annotative or the compositional paradigm, each of them having unique benefits and drawbacks. Language and tool designers select the used variability mechanism often solely based on intuition. A better empirical understanding of the comprehension of variability mechanisms would help them in improving support for effective modeling. In this paper, we present an empirical assessment of annotative and compositional variability mechanisms for class models. We report and discuss findings from an experiment with 73 participants, in which we studied the impact of the chosen variability mechanisms during model comprehension tasks. We find that, compared to the baseline of listing all model variants separately, the annotative technique did not affect developer performance. Use of the compositional mechanism correlated with impaired performance. For a subset of our tasks the annotative mechanism is preferred to the compositional one and the baseline. We present actionable recommendations concerning support of flexible, tasks-specific solutions, and the transfer of best established best practices from the code domain to models.

Keywords: model-driven engineering; class models; variability; software product lines

1 Summary

Variant-rich systems can offer companies major strategic advantages, such as the ability to deliver tailor-made software products to their customers. Still, when developing a variant-rich system, severe challenges may arise during maintenance, evolution, and analysis, especially when variants are developed in the naive *clone-and-own* approach, that is, by copying and modifying them. As companies begin to streamline their development workflows for building variant-rich systems, they recognize a need for variability management in all key development artifacts, including models. The car industry is particularly outspoken on their need for model-level variability mechanisms.

Recognizing this need, researchers have started building variability mechanisms for models. Variability mechanisms are now available both for UML and DSMLs. Adoption in several

[1] Radboud University, Nijmegen, Netherlands d.strueber@cs.ru.nl
[2] IAV Automotive Engineering, Germany tony@anjorin.de
[3] Chalmers | University of Gothenburg, Sweden thorsten.berger@chalmers.se

doi:10.18420/SE2021_39

industrial DSMLs has demonstrated the general feasibility of model-level variability mechanisms in practice. Still, language and tool designers are offered little guidance on selecting the most effective variability mechanism for their purposes. In fact, there is a lack of evidence to support the preference of one mechanism over the other. Arguably, *comprehensibility* is a decisive factor for the efficiency of a variability mechanism—for any maintenance and evolution activity (e.g. bugfixing, feature implementations), the developers first need to understand the existing system. A better empirical understanding of the comprehension of variability mechanisms could support the development of more effective modeling languages and tools.

To this end, our paper [SAB20] presents an empirical study of variability mechanisms for class models, an ubiquitous modeling language. In a fully randomized experiment performed with 73 participants with relevant background, we studied how the choice of variability mechanism affects performance during model comprehension tasks. We consider two selected variability mechanisms that are representative for two main types: *Annotative* mechanisms maintain an integrated, annotated representation of all variants. They are conceptually simple, but can impair understandability since elements are cluttered with variability information. *Compositional* mechanisms allow to compose a set of smaller sub-models to form a larger model. They are appealing as they establish a clear separation of concerns, but involve a composition step which might be cognitively challenging. We aimed to shed light on the impact of these inherent trade-offs by using an annotative mechanism (model templates [CA05, St18]) and a compositional one (model refinement [An14]).

We present the following results: 1. Compared to working with an explicit enumeration of all variants, the annotative mechanism generally lead to a similar performance (completion times and correctness scores) and subjective difficulty ratings. 2. The compositional mechanism generally lead to worse performance and difficulty ratings. 3. The variability mechanism preferred by most participants depended on the considered task.

Bibliography

[An14] Anjorin, Anthony; Saller, Karsten; Lochau, Malte; Schürr, Andy: Modularizing triple graph grammars using rule refinement. In: FASE. Springer, pp. 340–354, 2014.

[CA05] Czarnecki, Krzysztof; Antkiewicz, Michał: Mapping features to models: A template approach based on superimposed variants. In: GPCE. Springer, pp. 422–437, 2005.

[SAB20] Strüber, Daniel; Anjorin, Anthony; Berger, Thorsten: Variability Representations in Class Models: An Empirical Assessment. In: MODELS. pp. 239–256, 2020.

[St18] Strüber, Daniel; Rubin, Julia; Arendt, Thorsten; Chechik, Marsha; Taentzer, Gabriele; Plöger, Jennifer: Variability-based model transformation: formal foundation and application. FAC, 30(1):133–162, 2018.