A Formal Framework for Incremental Model Slicing (Summary)

Gabriele Taentzer[1], Timo Kehrer[2], Christopher Pietsch[3], Udo Kelter[4]

Abstract: We report about a recently developed "Formal Framework for Incremental Model Slicing", published in [Ta18]. A model slice of a model is a submodel comprising a selected model part, called slicing criterion. In addition to classical use cases from the field of program understanding, model slicing is also motivated by specifying submodels of interest to be further processed more efficiently. Since slicing criteria are often modified during software development tasks, such slices often need to be updated. A slice update can be performed by creating the new slice from scratch or by incrementally updating the existing slice. We present a formal framework for defining model slicers that support incremental slice updates. This framework abstracts from the behavior of concrete slicers as well as from the concrete model modification approach. Incremental slice updates are shown to be equivalent to non-incremental ones. Furthermore, we present a framework instantiation based on the concept of edit scripts defining application sequences of model transformation rules, along with two two concrete model slicers implemented based on this instantiation.

Summary

Modeling frameworks such as the Eclipse Modeling Framework (EMF) do not scale beyond a few tens of thousands of model elements. Models of that size cannot even be edited in standard model editors. Thus, the *extraction of editable submodels from a larger model*, a specific form of model slicing, is the only viable solution to support an efficient yet independent editing of huge monolithic models [Pi17]. Adopting basic terminology from the field of program slicing as pioneered by Weiser [We81], a model slice is an interesting part of a model comprising a given slicing criterion, where it is up to a concrete slicing definition to specify which model parts are of interest.

Model slicing is faced with two challenging requirements which do not exist for traditional program slicers. First, the increasing importance and prevalence of domain-specific modeling languages (DSMLs) as well as a considerable number of different slicing specifications lead to a huge number of concrete slicers. Thus, methods for developing model slicers should abstract from a *slicer's concrete behavior*, and thus from a *concrete modeling language*, as far as possible. Second, slicing criteria are often modified during software development tasks since a slice created for an initial slicing criterion can turn out to be inappropriate.

[1] Philipps-Universität Marburg. taentzer@mathematik.uni-marburg.de
[2] Humboldt-Universität zu Berlin. timo.kehrer@informatik.hu-berlin.de
[3] Universität Siegen. cpietsch@informatik.uni-siegen.de
[4] Universität Siegen. kelter@informatik.uni-siegen.de

doi:10.18420/se2019-45

Rather than creating a new slice from scratch for a modified slicing criterion, slices must often be *updated incrementally*. This is indispensable for all use cases where slices are edited by developers since otherwise these slice edits would be blindly overwritten [Pi17]. In addition, incremental slice updating is a desirable feature when it is more efficient than creating the slice from scratch. To date, both requirements have been insufficiently addressed in the literature.

In this paper, we present a fundamental methodology for developing adaptable and incremental model slicers. To be independent of a concrete DSML and use cases, we restrict ourselves to static slicing in order to support both executable and non-executable models. Our framework is based on graph-based models and model modifications, and abstracts from the behavior of concrete slicers as well as from the concrete model modification approach. Within this framework, we show that incremental slice updates are equivalent to non-incremental ones. We present an instantiation of this formal framework where incremental model slicers are specified by model patches. Two concrete model slicers are implemented using EMF and the differencing and patching facilities of the model differencing framework SiLift [Si18].

The work presented in this paper has been conducted in terms of the research project MOCA [MO18], in which we develop fundamental methods, concepts and techniques supporting the version management of models. The extraction and updating of editable submodels supported through incremental model slicing is one of the key steps towards efficiently supporting the versioning of models based on a central repository and distributed workspaces. The repository hosts potentially huge models, while submodels thereof may be checked out into developer's workspaces in order to evolve certain parts of a model-based system.

Acknowledgment

This work was partially supported by the DFG (German Research Foundation) under the Priority Programme SPP1593: Design For Future - Managed Software Evolution.

References

[MO18] MOCA, http://www.dfg-spp1593.de/moca/index.html, 2018.

[Pi17] Pietsch, C.; Ohrndorf, M.; Kelter, U.; Kehrer, T.: Incrementally slicing editable submodels. In: Intl. Conf. on Automated Software Engineering. IEEE Press, 2017.

[Si18] SiLift, http://pi.informatik.uni-siegen.de/Projekte/SiLift, 2018.

[Ta18] Taentzer, G.; Kehrer, T.; Pietsch, C.; Kelter, U.: A Formal Framework for Incremental Model Slicing. In: International Conference on Fundamental Approaches to Software Engineering. Springer, pp. 3–20, 2018.

[We81] Weiser, M.: Program slicing. In: Intl. Conf. on Software Engineering. IEEE Press, 1981.