Henshin: A Model Transformation Language and its Use for Search-Based Model Optimisation in MDEOptimiser

Daniel Strüber[1], Alexandru Burdusel[2], Stefan John[3], Steffen Zschaler[4]

Abstract: This tutorial presents Henshin, a versatile model transformation language increasingly used in academic and industrial applications. Henshin is based on the paradigm of graph transformation and provides a comprehensive tool set that supports largely declarative transformation specifications and various formal analyses. We present the application of Henshin in a *search-based model optimisation* task, where the goal is to find an optimal model regarding a given fitness function. Using Henshin, we specify evolutionary operators for MDEOptimiser, a novel search-based model optimisation tool.

Keywords: model transformation; graph transformation; model optimisation; evolutionary optimisation

1 Summary

Model transformation has been called the heart and soul of model-driven engineering, a paradigm in which models are continuously improved, refined, and translated. While transformations can, in principle, be developed using any general-purpose-language, these languages usually do not offer any support for challenges faced during transformation development, such as the need for verification, traceability, and optimisation. To better support developers, a variety of dedicated model transformation languages has emerged.

Henshin [Ar10, St17b] is a model transformation language based on the paradigm of algebraic graph transformations. Henshin's key benefits are: (i) a visual syntax, supporting a largely declarative specification of transformations, (ii) a mature formal foundation, enabling various formal analyses, and (iii) a comprehensive tool chain, comprising various editors, execution engines, and analysis tools. Supported analyses include model checking as well as conflict and dependency analysis. In academia, Henshin has been used in settings such as model versioning, model refactoring, and software product line transformations. In industry, Henshin has been used to verify the correctness of satellite control procedure translations.

Search-based model optimisation [ZM16] is a recent trend that combines the benefits of model-driven and search-based software engineering. Search-based software engineering

[1] Universität Koblenz-Landau, Universitätsstr. 1, 56070 Koblenz, Germany, strueber@uni-koblenz.de
[2] King's College London, London WC2R 2LS, United Kingdom, alexandru.burdusel@kcl.ac.uk
[3] Philipps-Universität Marburg, Hans-Meerwein-Str., 35032 Marburg, Germany, stefan.john@uni-marburg.de
[4] King's College London, London WC2R 2LS, United Kingdom, steffen.zschaler@kcl.ac.uk

provides software developers with technologies to solve optimisation tasks such as test case selection, component deployment, or release bundling. Yet, the encoding of these tasks as search problems is a complex and error-prone task that requires substantial expertise in meta-heuristic technologies, such as evolutionary optimisation algorithms.

To alleviate this issue, search-based model optimisation makes this expertise available by incorporating it into an optimisation framework. The user provides a solution-space specification, comprising a meta-model and some in-place model transformations for modifying the candidate solutions, i.e., the meta-model's instances. The model transformations can be specified manually, or even generated automatically [St17a]. In summary, search-based model optimisation supports the black-box use of search technologies, by enabling a problem specification based on domain concepts, rather a technology-specific encoding.

MDEOptimiser [BZ17] is a search-based model optimisation framework based on Henshin. Like other recent frameworks, in particular MOMoT [FTW16], MDEOptimiser involves Henshin to bridge model transformation with optimisation based on genetic algorithms. As its distinguishing feature, MDEOptimiser uses Henshin rules to specify the genetic operators, in particular the mutation operator. This set-up is particularly efficient in situations where the model itself, rather than the orchestration of rules, is to be optimised.

Goals. Participants will take three things from the tutorial: (i) How to specify and apply model transformation rules using Henshin, (ii) How to specify an optimisation problem using MDEOptimiser and Henshin, and (iii) How to design a high-quality mutation operator.

Prerequisites. Participants should be familiar with the Eclipse Modeling Framework (EMF), which provides the underlying modeling platform for Henshin. In particular, they should know how meta-models and model instances are specified using EMF.

References

[Ar10] Arendt, Thorsten; Biermann, Enrico; Jurack, Stefan; Krause, Christian; Taentzer, Gabriele: Henshin: advanced concepts and tools for in-place EMF model transformations. In: MODELS. Springer, pp. 121–135, 2010. https://www.eclipse.org/henshin/.

[BZ17] Burdusel, Alexandru; Zschaler, Steffen: , MDE Optimiser. https://mde-optimiser.github.io/, 2017.

[FTW16] Fleck, Martin; Troya, Javier; Wimmer, Manuel: Search-based model transformations with MOMoT. In: ICMT. Springer, pp. 79–87, 2016.

[St17a] Strüber, Daniel: Generating Efficient Mutation Operators for Search-Based Model-Driven Engineering. In: ICMT. pp. 121–137, 2017.

[St17b] Strüber, Daniel; Born, Kristopher; Gill, Kanwal Daud; Groner, Raffaela; Kehrer, Timo; Ohrndorf, Manuel; Tichy, Matthias: Henshin: A Usability-Focused Framework for EMF Model Transformation Development. In: ICGT. pp. 196–208, 2017.

[ZM16] Zschaler, Steffen; Mandow, Lawrence: Towards model-based optimisation: Using domain knowledge explicitly. In: MELO. pp. 317–329, 2016.