# Optimisation Methods for Model-Driven Engineering *

Alexandru Burdusel

Department of Informatics, King's College London, London, UK, WC2R 2LS
alexandru.burdusel@kcl.ac.uk

**Abstract.** Recently there has been increased interest in combining the fields of Model-Driven Engineering (MDE) and Search-Based Software Engineering (SBSE). Currently, when solving MDE-SBSE problems, in addition to the problem description, the user is required to manually provide design space exploration (DSE) information, encoded as model transformation rules and an optimisation algorithm. Performance and solution quality strongly depend on the right choice of transformations and optimisation algorithm.
The aim of this research is to develop an approach to solving MDE optimisation problems, by removing the need to manually specify model transformations and an optimisation algorithm. The inputs required for the problem description are the metamodel, the initial model if one is available and a set of constraints and fitness functions. The remaining components needed for running the optimisation, are inferred from the given problem description. Solving this challenge allows domain experts to model optimisation problems using MDE and find good solutions without needing any knowledge about model transformations, constraint solving or optimisation.

**Keywords:** model driven engineering, domain specific language, search based optimisation

## 1 Introduction

MDE introduces models as a principal entity to describe complex engineering problems at a higher abstraction level than code [1]. The model is a representation consistent with a domain specified by the parent metamodel. A model is also referred to as an instance of a metamodel. A core MDE concept is model transformations, which enable model creation and alteration while maintaining metamodel consistency of the transformed model.

SBSE is a methodology for describing software engineering problems as optimisation problems [2]. A search based problem consists of a system to represent solutions, a process to generate new solutions from existing ones and a solution quality evaluation method.

* This research is in the first year stage.

Recently there has been an increasing amount of interest in combining the fields of MDE and SBSE [3,4,5,6]. For all of these approaches, the user describes the optimisation problem using MDE and SBSE concepts and the tool returns converging solution models resulting from running the optimisation. In [6] the authors propose MOMoT, a rule based optimisation tool that offers a Domain Specific Language (DSL), implemented as an Eclipse plugin, to enable users to specify optimisation problems in MDE. Viatra-DSE is another tool performing rule based model optimisation [7]. Both tools in [6] and [7] optimise a chain of rule applications to find the most suitable derivation chain which applied to an initial model results in a good solution model. A tool that runs optimisation directly on models is Crepe [3]. It has been extended in [4] to support multiple objectives. The tool uses a generic encoding for the models, as a set of integers, which is then used by genetic algorithms to apply mutations and crossover.

The research question this PhD aims to answer is: *Can we simplify the process of running optimisation on models?* We aim to contribute solutions to the following challenges in response to this research question: a) manual specification of transformation rules to be used as search operators requires knowledge about model transformations, optimisation and how the two fields can be combined; b) manual optimisation algorithm selection suitable for the problem being solved requires advanced knowledge about optimisation methods; c) choosing the most suitable model for the chosen optimisation strategy is not trivial. The proposed solution is to build a tool offering a DSL that allows users to specify an MDE optimisation problem by supplying only the metamodel, an optional initial model and a set of constraints and fitness functions. Using this information, the tool will then determine the best model to start the optimisation from, along with automatically generated transformation rules and a problem suited optimisation algorithm.

The contributions of this research will make the specification and solving of MDE optimisation problems more accessible, enabling domain experts to solve them without having knowledge about transformations and optimisation methods. The proposed solution will be evaluated by using the tool to solve industry case studies [8,4]. This type of evaluation will show if our proposed automated approach is better than the existing user driven alternatives.

## 2 Background

Search-Based MDE is the idea of combining the concepts of search based optimisation (SBO) and MDE, in order to solve optimisation problems specified using MDE [9]. An SBO problem specification requires the following elements: a) a candidate representation method; b) operators to generate new solutions from existing ones, by mutation or breeding; and c) a candidate quality evaluation method commonly referred to as fitness functions.

The MDE model and metamodel concepts are an ideal equivalent of the candidate and search space representations from SBO. In MDE, model transformations are the process to change the structure of models while ensuring

that their metamodel conformance is maintained. In SBO, transformations can be seen as operators, which are the process to mutate or combine individuals to explore the search space in order to find better solutions. The SBO search space is the set of all possible candidates that can be solutions to the problem being optimised. The process of generating new candidates using operators and evaluating their suitability is also known as DSE.

## 3 Problem statement

Existing tools solving MDE-SBSE problems require the users to specify a problem description consisting of a metamodel, model instances, model transformations and the optimisation algorithm. The problem is that in order to use these tools to find a solution to an optimisation problem, the user is required to know not only the domain of the problem but also how to specify model transformations and what optimisation algorithm is most suitable to solve it.

The aim of our research is to propose an approach to automatically infer from the *problem specification* consisting of a *metamodel* and a *set of constraints and fitness functions*, the most suitable starting model for the optimisation process required for the current problem, the best operators for efficient design space exploration and the most suitable optimisation algorithm. To make this possible, we have identified the following challenges that would have to be solved as part of this research:

1. *Model evolutions.* In order to efficiently explore the search space, it may not be enough to have transformation rules that ensure model consistency, but which also avoid local optima and provide good design space exploration during the search, through mutations and breeding. Automatically generating such transformation rules remains a challenging problem;
2. *Optimisation algorithms.* Automatically selecting the most suitable optimisation algorithm from a deterministic or stochastic repository of algorithms, using only high-level problem description and no user input is not trivial. Using the right algorithm for a problem can result in considerable performance and quality improvements. The selected algorithm must also be suitable for the generated model evolutions. Making this selection requires in-depth knowledge about optimisation algorithms;
3. *Initial model provision.* Performance can be improved if the model size can be reduced. This can be achieved if the start model contains parts that remain static during the optimisation. These parts can be removed from the model while running the optimisation, then added back at the end. This step can be done manually by the user, however, it can be time consuming and error prone. Doing this step automatically remains a challenge.

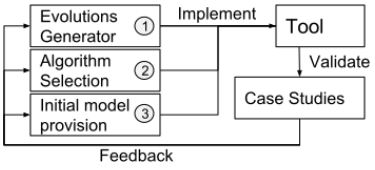
## 4 Approach

The research has been started by identifying case studies that can be represented as MDE-SBSE problems [8,4]. The number of case studies will be continuously increased for the duration of the project.

In Fig. 1 we identify the three steps to solve the research challenges discussed in Sect. 3. Step 1, consists of using the constraints, objectives and the metamodel to generate efficient DSE operators. In Step 2, the most suitable search algorithm to solve the problem will be identified using the generated operators, the tool input and knowledge about algorithm requirements. Step 3 is finding the best model to start the optimisation from. This step consists of reducing the model size if possible, by automatically removing static model parts before the optimisation and gluing them back to the resulting solution models. For each challenge we will propose an algorithm. This will then be implemented in our tool and validated using the identified case studies.

The tool is built as an Eclipse plugin. The optimisation algorithms supported by the tool are implemented using the MOEA framework [1]. A high level architecture of the tool can be seen in Fig. 2. The grayed box for the initial model denotes that in some instances this model may be given only in some cases and therefore, when not available, it would have to be generated automatically using a tool like Cartier [10].

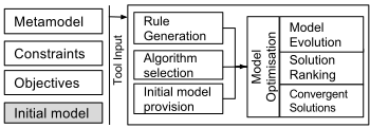
Fig. 1: Research steps.




## 5 Current status

The research has been started by working on the *model evolutions* challenge. We have found promising results by proposing an automated way of generating transformation rules from a metamodel and a set of additional multiplicity constraints. Our approach has been tested using the Class Responsibility Assignment(CRA) case study proposed as a challenge at TTC 16 [8]. The CRA scores found by the automatically generated rules have been close or better in some cases than the scores obtained with the manually defined rules. A detailed overview of our results for this implementation can be found in [11]. We are now working on generalising the approach and validating it with more case studies. A high level overview of the expected timeline for completion has been included in Fig. 3.

Fig. 2: Tool architecture.




[1] http://moeaframework.org/


Fig. 3: Project Timeline.

## 6 Related work

In [12] the authors introduce an automated way for generating consistency preserving edit operations(CPERs) from a metamodel. The rules are generated so that they can generate a valid model upon multiple applications. Another approach to generating transformation rules is proposed in [13]. The author presents a framework to generate DSE exploration rules using a genetic algorithm which is used to run a set of higher-order transformations from a training model. The obtained rules can then be used to transform models conforming to the same metamodel as the training model. This approach is one possible solution to the identified evolutions generator challenge, however, we aim to generate transformation rules using a deterministic algorithm, eliminating the performance overhead of a genetic algorithm.

In [14] the author introduces the model transformations by example(MTBE) approach to generating model transformation rules using an iterative, semi-automated method of generating model transformation rules from a set of example mappings between the source and the target models of the transformations, provided by the user at the start of the process. At each iteration, the user can refine the generated rules by validating them with more test models. Using this approach, the quality of the generated rules depends on the intuition of the user and also on increased availability of test models. In [15] the authors propose the MOdel Transformation as Optimisation by Examples(MOTOE) approach to transform a source model into a target model using particle swarm optimisation(PSO) [16] and without specifying transformation rules.

Kessentini et al. propose an approach to generating transformation rules as an optimisation problem in [17]. The solution starts by randomly generating transformation rules, which are used to find target models. The solution uses PSO to find the best rules that generate good quality target models and once these rules are identified a local heuristic is used for further improvement.

In [18], the author identifies some of the current approaches to automatically selecting an algorithm to solve an optimisation problem. The survey also highlights the need for more contributions to this field as the problem of algorithm selection is non-trivial.

## 7 Conclusions

This research will contribute to the field by providing an automated way to solve complex MDE optimisation problems by only requiring the user to input their metamodel, an optional initial model, and a set of constraints and fitness

functions. The solution will automatically determine: a) the most suitable model to start the optimisation from, b) the most efficient transformation rules to use for exploring the search space and c) the appropriate optimisation algorithm to find the best solution. These contributions will consist of proposed algorithmic approaches to solve them and a tool implementing the algorithms to validate their generality using industry case studies.

## References


1. Atkinson, C., Kuhne, T.: Model-driven development: a metamodeling foundation. IEEE software **20**(5) (2003) 36–41
2. Harman, M., Jones, B.F.: Search-based software engineering. Information and Software Technology **43**(14) (2001) 833–839
3. Williams, J.R.: A novel representation for search-based model-driven engineering. PhD thesis, University of York, UK (2013)
4. Efstathiou, D., Williams, J.R., Zschaler, S.: Crepe complete: Multi-objective optimisation for your models. In: Proc. 1st Int'l Workshop on Combining Modelling with Search- and Example-Based Approaches (CMSEBA'14). (2014)
5. Abdeen, H., Varró, D., Sahraoui, H., Nagy, A.S., Debreceni, C., Hegedüs, Á., Horváth, Á.: Multi-objective optimization in rule-based design space exploration. In Crnkovic, I., Chechik, M., Grünbacher, P., eds.: Proc. 29th ACM/IEEE Int'l Conf. Automated Software Engineering (ASE'14), ACM (2014) 289–300
6. Fleck, M., Troya, J., Wimmer, M.: Marrying search-based optimization and model transformation technology. Proceedings of the First North American Search Based Software Engineering Symposium (2015) 1–16
7. Hegedüs, Á., Horváth, Á., Ráth, I., Varró, D.: A model-driven framework for guided design space exploration. In: Proc 26th IEEE/ACM Int'l Conf. Automated Software Engineering (ASE'11). (November 2011) 173–182
8. Fleck, M., Troya, J., Wimmer, M.: The Class Responsibility Assignment Case. In: Proceedings of the 9th Transformation Tool Contest @STAF. Volume 1758 of CEUR Workshop Proceedings. (2016) 1–8
9. Burton, F.R., Poulding, S.: Complementing metaheuristic search with higher abstraction techniques. In: Proceedings of the 1st International Workshop on Combining Modelling and Search-Based Software Engineering, IEEE Press (2013) 45–48
10. Sen, S., Baudry, B., Mottu, J.M.: Automatic model generation strategies for model transformation testing. In: International Conference on Theory and Practice of Model Transformations, Springer (2009) 148–164
11. Burdusel, A., Zschaler, S.: Automatic generation of evolution rules for model-driven optimisation. In: 8th International Workshop on Graph Computation Models (GCM'17). (2017) accepted.
12. Kehrer, T., Taentzer, G., Rindt, M., Kelter, U.: Automatically deriving the specification of model editing operations from meta-models. In: International Conference on Theory and Practice of Model Transformations, Springer (2016) 173–188
13. Strüber, D.: Generating efficient mutation operators for search-based model-driven engineering. In: International Conference on Theory and Practice of Model Transformations (ICMT). (2017) accepted.
14. Varró, D.: Model transformation by example. In: Proc. Model Driven Engineering Languages and Systems (MODELS 2006). Volume 4199 of LNCS., Springer (2006) 410–424

15. Kessentini, M., Sahraoui, H., Boukadoum, M.: Model transformation as an optimization problem. In Czarnecki, K., Ober, I., Bruel, J.M., Uhl, A., Völter, M., eds.: Model Driven Engineering Languages and Systems: 11th International Conference, MoDELS 2008, Toulouse, France, September 28 - October 3, 2008. Proceedings, Berlin, Heidelberg, Springer Berlin Heidelberg (2008) 159–173
16. Eberchart, R., Kennedy, J.: Particle swarm optimization. In: IEEE International Conference on Neural Networks, Perth, Australia. (1995)
17. Kessentini, M., Wimmer, M., Sahraoui, H., Boukadoum, M.: Generating transformation rules from examples for behavioral models. In: Proceedings of the Second International Workshop on Behaviour Modelling: Foundation and Applications, ACM (2010) 2
18. Kotthoff, L.: Algorithm selection for combinatorial search problems: A survey. AI Magazine **35**(3) (2014) 48–60