On the Application of Model-Driven Optimization to Business Processes

Gabriele Taentzer[1], Jens Kosiol[1,2], and Leen Lambers[3]

[1] Philipps-Universität Marburg, Marburg, Germany
taentzer@mathematik.uni-marburg.de
kosiolje@mathematik.uni-marburg.de
[2] Universität Kassel, Kassel, Germany
[3] BTU Cottbus-Senftenberg, Cottbus, Germany leen.lambers@b-tu.de

Abstract. The optimization of business processes is an important task to increase the efficiency of the described workflows. Metaheuristic optimization, such as evolutionary search, has been used to optimize business process models, but it requires a high level of expertise that not all process designers have. *Model-driven optimization* (MDO) promises to make the use of metaheuristic optimization accessible to domain experts without in-depth technical expertise by allowing them to specify the optimization algorithm directly at the model level. Because this approach is less technical, the process designers can focus on the business process models and their properties. Using concrete business process optimization problems as a starting point, we discuss how MDO can be applied to these problems, what MDO would offer for business process optimization, and how the application to business processes could stimulate research on MDO.

Keywords: Business process models · Optimization · Model-driven engineering

1 Introduction

The optimization of business processes is a strategic activity in organizations because it can increase the efficiency of work. A number of metrics have been developed to analyze the quality of business process models [47]. Granularity, which is reflected in the size of activities, is crucial for the design of balanced processes; it can be measured with coupling and cohesion metrics [36]. To speed up work, workflows can be further optimized by increasing parallelism within tasks [12]. However, optimization is a difficult task when performed manually, especially when multiple objectives must be considered. It requires a well-suited optimization algorithm; the development of such an algorithm requires a high level of expertise that not all process designers have. For example, the implementation of an evolutionary algorithm usually requires (aspects of) business process models to be encoded in integer representations in order to perform an evolutionary search [46]. To make optimization accessible to domain experts, an

approach is needed that allows optimization tasks to be specified and executed without deep technical expertise.

Various software engineering problems, such as software modularization [6], process optimization, and release planning [2], have already been considered as optimization problems. They have often been solved by using evolutionary algorithms [20] which mimic the evolution in nature to solve optimization problems. Model-driven engineering (MDE) [38] aims at representing domain knowledge in models and solving problems through model transformations. MDE can be used in the context of evolutionary optimization to minimize the expertise required by users of optimization techniques. This combination of optimization and MDE is referred to in the literature as model-based or model-driven optimization (MDO) [1, 3, 9, 18, 22, 48]. It applies evolutionary optimization to models. MDO can simplify the application of evolutionary search to software engineering problems, because models are not encoded, but the search space consists directly of models that are evolved by model transformations. A conceptual framework that precisely defines all the main concepts of MDO based on graphs and graph transformation is presented in [23]. It is intended to assist the modeler in using MDO to solve such optimization problems.

Since we are focusing on evolutionary algorithms as the optimization technique in this paper, we will briefly recall them. With reference to, for example, [5, 16, 23, 48], an *evolutionary algorithm* is used to solve an optimization problem. Usually, such a problem is formally defined by means of an *objective* (or *fitness*) *function* that expresses the objective that is to be optimized. In practical applications, multiple objective functions often must be addressed simultaneously, leading to the concepts of *multi–objective problems* and *many-objective problems* and *Pareto optimization* [41]. For the optimization process, one needs a representation of possible solutions to the problem at hand; the solutions constitute the *search space*. Practical optimization problems usually come with additional *constraints* that clarify which of the represented solutions are *feasible* (i.e., constitute valid solutions to the optimization problem). Given a *constrained optimization problem* and a representation for solutions, the key ingredients of an evolutionary algorithm are a generator for an initial population of solutions, a mechanism for generating new solutions from existing ones (e.g., by *mutation* or *crossover*), a selection mechanism that typically implements the evolutionary concept of survival of the fittest, and a condition for stopping evolutionary computations.

MDO applies ordinary evolutionary algorithms, such as the well-known NSGA-II algorithm [10], or other metaheuristic search techniques to (constrained) optimization problems but uses models, e.g., [8, 23, 48], or model transformation sequences, e.g., [1, 3, 18], as representation for solutions. With MDEOptimiser [8] and MOMoT [3], tool support for both approaches is available so that optimizations can be performed. In this paper, we propose the use of MDO for business process optimization. We focus on the model-based approach to MDO (as it tends to perform better [22]) and propose to optimize models of workflow processes directly, e.g., in the standardized workflow modeling language BPMN [34].

Business processes have been optimized in various ways [47]. In Section 2, we present a selection of typical optimization problems that have been considered for them. For each problem, we discuss the type of model being optimized, the objectives for which the models are being optimized, and the constraints that a feasible solution must satisfy. In Section 3, we describe for each of the selected problems how MDO can be applied to it and what the benefits and challenges are. Section 4 concludes the paper.

2 Optimization problems on business processes

The selected optimization problems we consider are the clustering of information elements (Sect. 2.1), the multi-objective optimization of non-functional properties of a business process (Sect. 2.2), and the parallelization of tasks of a business process (Sect. 2.3).

2.1 Clustering of information elements in workflow processes

The first optimization problem is formulated for workflow processes which can be seen as a conceptual basis for business processes [44]. Workflow processes have also been formalized as a special class of Petri nets, more precisely as *workflow nets*, to validate them.

A *workflow process* [36] contains a number of information elements that are used as input or output elements of operations. An operation is a basic processing step; it has one or more input information elements and one output information element. An activity in a workflow process consists of one or more operations. The output of one of the operations can be the input of another operation of that activity. A workflow process is *valid* if (1) all operations occur at least once in an activity, and (2) if the input of one operation depends on the output of another operation, then the respective activities of which they are part of are ordered so that they respect this dependency. While constraint (1) ensures the completeness of the activity design, constraint (2) ensures the correctness of their ordering.

The *optimization problem* to be solved is to find a good clustering of information elements and operations into activities. A clustering with low coupling between activities and high cohesion within each activity is considered the best, since in this case the clusters can be well identified. In general, *coupling* measures the number of connections between the elements of a model [45]. In workflow processes, two activities are coupled if they share one or more common information elements. The *cohesion* metric for workflow processes in [36] measures the coherence within the activities of the process model. Similar to the coupling metric, this cohesion metric also focuses on the information processing in the process. The *clustering problem* can be formulated as a *multi-objective problem*, since we aim for low coupling and high cohesion. The validity constraints presented above formulate the *feasibility constraints*.

2.2 Multi-objective optimization of non-functional properties

A business process has several non-functional properties of interest, such as cost, flow time, product quality, etc. Regularly, business processes are optimized with respect to these criteria. To be of practical importance, an optimization approach must consider multiple criteria simultaneously, e.g., minimizing cost and flow time while maximizing quality.

Vergidis et al. [46] propose a framework for the multi-objective evolutionary optimization of non-functional properties of business processes. They assume a set of *tasks* to be given, where for each task additional information is provided: A task comes with a set of *input resources* (it consumes), *output resources* (it produces), and values for attributes of interest (such as cost or duration). For a concrete optimization problem, a set of *input resources* and a set of *output resources* are given. The goal is to find a process (a subset of the given tasks) that produces the required output resources from the given input resources. That is, for a process to be *feasible*, the selected tasks must satisfy certain constraints: (1) each input resource of the optimization problem must be consumed by at least one selected task; (2) each output resource of the optimization problem must be produced by at least one selected task; and (3) the selected tasks must lead to a connected process diagram. The process should be optimal with respect to selected non-functional properties that are defined by objective functions. The proposed framework is generic in the sense that it is not restricted to specific non-functional properties. It is simply assumed that the objective functions can be computed by aggregating the attribute values of the selected tasks.

The framework in [46] uses multi-objective evolutionary algorithms (MOEAs) to search for optimal business processes. Different representations of a solution are used for different operators of such an algorithm. Basically, a business process is encoded as an array containing the tasks that make up the process. Standard variation operators (crossover and mutation) are applied to these arrays. The fitness of a solution with respect to the multiple objectives can be computed by aggregating the values of selected tasks for the respective attributes; the necessary information for this is stored in a matrix. To check the feasibility constraints, [46] develops a *Process Composition Algorithm* (PCA) that assembles the selected processes into a process diagram (repairing certain constraint violations on the way). The resulting process design can then be checked for feasibility; if constraint violations remain, their severity is computed in a *Degree of Infeasibility*.

2.3 Parallelization of tasks

The third optimization problem concerns business process optimization for processes described using the standardized workflow modeling language BPMN [34]. Durán and Salaün present an automated approach to optimizing *BPMN models that are enriched with a description of the execution time and resources associated with tasks* [12]. These enriched business process models take into account not only behavioral but also quantitative aspects.

The *optimization problem* aims at finding a reorganized enriched BPMN model with reduced execution time. Possible reorganizations of tasks within the BPMN model are described using *refactorings*. These reorganizations take into account the resources used by each task. The main idea for reducing the execution time is to increase parallelism between tasks as much as possible. The refactorings must take into account *specific constraints*. For example, tasks can only run in parallel if they do not compete for the same resources. Also, causal dependencies between tasks may need to be preserved when adding parallelism. These constraints represent *feasibility constraints* that must be satisfied for a solution to the optimization problem to be valid.

2.4 Summary of optimization problems

We briefly summarize the similarities and differences of the three selected optimization problems. All three optimization problems can be formulated for *models of business processes*. In the clustering problem, these models describe workflow processes; for the optimization of non-functional properties, the models are currently encoded as arrays of tasks in the business process. In the parallelization problem, the standard modeling language BPMN is already used. The *objective* of the clustering problem is of a structural kind, while the objectives of the other two problems are more behavioral, since they both focus on optimizing non-functional properties. In the parallelization problem, models even must be simulated in order to evaluate the objective function. All three problems share the fact that structural constraints must hold for a solution to be *feasible*. In particular, in the parallelization problem, it becomes clear that behavior preservation comes in addition to the preservation of structural constraints.

3 Applying model-driven optimization to business processes

In this section, we outline how MDO can be used to solve business process optimization problems and illustrate our ideas at the selected problems just recalled. We also discuss where MDO offers promising solutions to these optimization problems and where the problems present challenges that could trigger interesting research in MDO.

MDO for business processes As explained in the introduction, MDO denotes an approach to metaheuristic optimization in which models are crucial artefacts.[4] Thus, applying MDO to business processes amounts to developing model transformations in business process modelling languages such as BPMN. These

[4] Typically, models or model transformation sequences constitute the search space, and searching means modifying models or model transformation sequences. For feature model configuration [21, 30], search operators are designed and verified based on models; however, for the actual search, models and operators are translated into more machine-oriented representations to increase efficiency.

model transformation rules can then be used as search operators in optimization processes, where the search space consists of models (or model transformation sequences). While rules tailored to a specific optimization problem and a specific optimization algorithm promise the best results, MDO offers a certain degree of genericity. A set of rules that specifies basic modifications of business process models can constitute the search operators for different types of optimization algorithms (such as evolutionary algorithms, hill climbing, or simulated annealing) and for different optimization problems (i.e., different objective functions with respect to which models are evaluated).

For the three optimization problems presented, MDO would mean the following. For the clustering of information elements (in Section 2.1), (coupling and cohesion) metrics have been defined that can be used to estimate the quality of a given business process. However, no automated approach has been proposed to optimize business processes with respect to these quality criteria. MDO serves as such an approach. Coupling and cohesion, as defined in [36], are the objectives; they can be combined into a single objective function or be kept separate and multi-objective optimization is employed. For example, given a suitable set of transformation rules and a business process, evolutionary search can be used to find a clustering of the information elements that has low coupling and high cohesion.

For the optimization of non-functional properties (in Section 2.2), multi-objective optimization via evolutionary algorithms is already performed to find business processes of high quality. Applying MDO here means to not encode the models into arrays for the search but to use them directly. Below, we discuss the benefits we expect from this change.

For the parallelization of tasks (in Section 2.3), the authors already use models as crucial artefacts and the search is performed by model transformation. Thus, the work [12] can be considered as an instance of MDO. However, instead of using metaheuristic search, they explore their search space completely or via a hand-crafted heuristic, which is not feasible for larger search spaces or specific to their problem at hand. The framework of MDO here broadens the perspective to use the transformation rules suggested in [12] also for other optimization problems on business models, to try out other algorithms for the problem tackled in [12], or to complement the objective addressed in [12] by further ones, making the problem multi-objective.

Expected benefits One of the promises of MDO is to make optimization accessible to domain experts without deep technical expertise. Domain experts need only interact with models they are already familiar with, and may even be able to design domain-specific search operators (i.e. transformation rules) that are well suited to the search. While this hope has yet to be empirically verified, testing it for business process optimization is appealing because process design is typically a domain in which domain experts without technical expertise are involved.

On the technical side, the main promise of MDO lies in the strong formal foundation that model transformation has in graph transformation [15]. It can be used to make the search for *feasible* solutions easier. Realistic optimization

problems often come with constraints that the solutions must satisfy. As seen above, this is also the case for all three optimization problems considered here. For the optimization of non-functional properties [46], the authors even explicitly mention that obtaining feasible process models during evolutionary search is a major challenge; most constructed solutions are infeasible. For graph transformation rules, there is ample experience with regard to the treatment of constraints (expressed in different kinds of logics). Such rules can be analyzed for preserving the validity of given constraints, e.g., [13, 27, 40], be equipped with application conditions that prevent rule applications that would introduce constraint violations, e.g., [19, 32], or, for certain types of constraints, even be adapted so that the constraints are preserved, e.g., [7, 21, 25]. In addition, there is a research focus on repairing graphs and models with respect to constraints, which is another way to make infeasible mutation and crossover results feasible, e.g., [29, 33, 37, 39].

MDO has begun to make use of these formal results. There is empirical evidence that evolutionary search on models benefits from transformation rules that preserve the given constraints [7, 21, 23]. For certain types of constraints (multiplicities), transformation rules that preserve them can be automatically derived from a meta-model [7], i.e., for a given modeling language. So we are convinced that MDO can be successfully used to optimize business processes for various purposes.

Raised research challenges Above, we argued that basing (evolutionary) search on model transformation provides a means to address the problem of *structural constraints* that are expected to hold for solutions. However, when optimizing business processes, one usually needs to consider *behavioral constraints* as well: Mostly, the optimized process should still exhibit the same behavior as the original one. For example, the optimization of non-functional properties requires that a feasible process uses the given resources and produces the required resources. This constraint can be expressed as a formula on the graph structure and be treated as described above. However, behavioral equivalence of the optimized process with the original one is often expressed as a simulation or bisimulation.

There are techniques that allow one to check that the input and the output of a graph (or model) transformation are behaviorally equivalent, e.g., are in a (bi)simulation relation (see, e.g., [4, 14, 17, 31, 35]). However, there seems to be less research on this topic than for structural constraints. Furthermore, we are not aware of any research in MDO on preserving behavioral equivalence during search. There is recent work on formalizing the BPMN execution semantics using graph transformation, which facilitates behavioral property checking [28], which could serve as a starting point for this line of research. All in all, we expect that business process optimization can stimulate new research on preserving behavioral equivalence during model transformation, thus enriching the set of techniques that are used in MDO.

Another challenge is to develop an appropriate crossover operator for business process models. A crossover operator typically mixes information from two parent solutions to compute (one or two) child solutions that resemble their parents. In general, evolutionary search can benefit from using both crossover and

mutation, rather than just mutation alone [11, 42]. We have started to develop a generic (i.e., domain-agnostic) crossover operator for use in MDO [24, 26]. It unifies many crossover operators that have been proposed for specific models or graphs. Initial experiments show some increase in search effectiveness compared to using mutation alone, but the experiments also seem to indicate that this generic operator suffers from producing too many infeasible solutions, i.e., from introducing constraint violations. We combined this operator with ad-hoc repair of the computed solutions [24]. In applying MDO to business processes, we expect this effect to occur as well, so the research on the development of constraint-preserving crossover operators [43] needs to be continued. In addition, research is needed on how to concretize the generic crossover operator for the business process domain.

4 Conclusion

Business process optimization is an important task for increasing the efficiency of workflows. To make optimization accessible to domain experts, an approach is needed that allows optimization tasks to be specified and executed without deep technical expertise. In this paper, we have argued that it would be promising to tailor model-driven optimization to business processes. We expect that this would make it easier for process designers to apply metaheuristic optimization such as evolutionary search to their optimization problems. We have recalled three selected business process optimization problems and sketched how they could be tackled and benefit from the use of MDO principles and techniques.

The application of MDO to business process optimization also poses new challenges for MDO. MDO has mostly been considered to address problems in software engineering with objectives of a structural nature such as modularization. Since business processes describe behavior, it must also be shown that the optimized processes comply with behavioral constraints. Finally, the development of domain-specific crossover operators seems to be another relevant research goal. We have discussed how existing research results on model and graph transformation can support these lines of research in order to successfully apply MDO to business processes.

References

1. Abdeen, H., Varró, D., Sahraoui, H.A., 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.) ACM/IEEE International Conference on Automated Software Engineering, ASE '14, Vasteras, Sweden – September 15–19, 2014. pp. 289–300. ACM (2014). https://doi.org/10.1145/2642937.2643005
2. Bagnall, A.J., Rayward-Smith, V.J., Whittley, I.M.: The next release problem. Inf. Softw. Technol. **43**(14), 883–890 (2001). https://doi.org/10.1016/S0950-5849(01)00194-X

3. Bill, R., Fleck, M., Troya, J., Mayerhofer, T., Wimmer, M.: A local and global tour on MOMoT. Softw. Syst. Model. **18**(2), 1017–1046 (2019). https://doi.org/10.1007/s10270-017-0644-3
4. Bisztray, D., Heckel, R., Ehrig, H.: Compositional verification of architectural refactorings. In: de Lemos, R., Fabre, J., Gacek, C., Gadducci, F., ter Beek, M.H. (eds.) Architecting Dependable Systems VI. Lecture Notes in Computer Science, vol. 5835, pp. 308–333. Springer (2008). https://doi.org/10.1007/978-3-642-10248-6_13, https://doi.org/10.1007/978-3-642-10248-6_13
5. Blum, C., Chiong, R., Clerc, M., De Jong, K., Michalewicz, Z., Neri, F., Weise, T.: Evolutionary optimization. Variants of evolutionary algorithms for real-world applications pp. 1–29 (2012)
6. Bowman, M., Briand, L.C., Labiche, Y.: Solving the class responsibility assignment problem in object-oriented analysis with multi-objective genetic algorithms. IEEE Trans. Software Eng. **36**(6), 817–837 (2010). https://doi.org/10.1109/TSE.2010.70
7. Burdusel, A., Zschaler, S., John, S.: Automatic generation of atomic multiplicity-preserving search operators for search-based model engineering. Softw. Syst. Model. **20**(6), 1857–1887 (2021). https://doi.org/10.1007/s10270-021-00914-w
8. Burdusel, A., Zschaler, S., Strüber, D.: MDEOptimiser: A Search Based Model Engineering Tool. In: Babur, Ö., Strüber, D., Abrahão, S., Burgueño, L., Gogolla, M., Greenyer, J., Kokaly, S., Kolovos, D.S., Mayerhofer, T., Zahedi, M. (eds.) Proceedings of the 21st ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings, MODELS 2018, Copenhagen, Denmark, October 14–19, 2018. pp. 12–16. ACM (2018). https://doi.org/10.1145/3270112.3270130
9. Burton, F.R., Poulding, S.M.: Complementing metaheuristic search with higher abstraction techniques. In: Paige, R.F., Harman, M., Williams, J.R. (eds.) 1st International Workshop on Combining Modelling and Search-Based Software Engineering, CMSBSE@ICSE 2013, San Francisco, CA, USA, May 20, 2013. pp. 45–48. IEEE Computer Society (2013). https://doi.org/10.1109/CMSBSE.2013.6604436
10. Deb, K., Agrawal, S., Pratap, A., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. **6**(2), 182–197 (2002). https://doi.org/10.1109/4235.996017
11. Doerr, B., Happ, E., Klein, C.: Crossover can provably be useful in evolutionary computation. Theor. Comput. Sci. **425**, 17–33 (2012). https://doi.org/10.1016/j.tcs.2010.10.035, https://doi.org/10.1016/j.tcs.2010.10.035
12. Durán, F., Salaün, G.: Optimization of BPMN Processes via Automated Refactoring. In: International Conference on Service-Oriented Computing. pp. 3–18. Springer (2022)
13. Dyck, J., Giese, H.: k-inductive invariant checking for graph transformation systems. In: de Lara, J., Plump, D. (eds.) Graph Transformation – 10th International Conference, ICGT 2017, Held as Part of STAF 2017, Marburg, Germany, July 18–19, 2017, Proceedings. Lecture Notes in Computer Science, vol. 10373, pp. 142–158. Springer (2017). https://doi.org/10.1007/978-3-319-61470-0_9, https://doi.org/10.1007/978-3-319-61470-0_9
14. Dyck, J., Giese, H., Lambers, L.: Automatic verification of behavior preservation at the transformation level for relational model transformation. Softw. Syst. Model. **18**(5), 2937–2972 (2019). https://doi.org/10.1007/S10270-018-00706-9, https://doi.org/10.1007/s10270-018-00706-9

15. Ehrig, H., Ehrig, K., Prange, U., Taentzer, G.: Fundamentals of Algebraic Graph Transformation. Monographs in Theoretical Computer Science, Springer (2006). https://doi.org/10.1007/3-540-31188-2
16. Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing. Natural Computing Series, Springer, 2nd edn. (2015). https://doi.org/10.1007/978-3-662-44874-8
17. Engels, G., Kleppe, A., Rensink, A., Semenyak, M., Soltenborn, C., Wehrheim, H.: From UML activities to TAAL – towards behaviour-preserving model transformations. In: Schieferdecker, I., Hartman, A. (eds.) Model Driven Architecture - Foundations and Applications, 4th European Conference, ECMDA-FA 2008, Berlin, Germany, June 9–13, 2008. Proceedings. Lecture Notes in Computer Science, vol. 5095, pp. 94–109. Springer (2008). https://doi.org/10.1007/978-3-540-69100-6_7, https://doi.org/10.1007/978-3-540-69100-6_7
18. Fleck, M., Troya, J., Wimmer, M.: Marrying search-based optimization and model transformation technology. In: Proceedings of the First North American Search Based Software Engineering Symposium. Elsevier (2015), http://publik.tuwien.ac.at/files/PubDat_237899.pdf, accessed: 2022-12-07
19. Habel, A., Pennemann, K.: Correctness of high-level transformation systems relative to nested conditions. Math. Struct. Comput. Sci. **19**(2), 245–296 (2009). https://doi.org/10.1017/S0960129508007202
20. Harman, M., Mansouri, S.A., Zhang, Y.: Search-based software engineering: Trends, techniques and applications. ACM Comput. Surv. **45**(1), 11:1–11:61 (2012). https://doi.org/10.1145/2379776.2379787
21. Horcas, J.M., Strüber, D., Burdusel, A., Martinez, J., Zschaler, S.: We're Not Gonna Break It! Consistency-Preserving Operators for Efficient Product Line Configuration. IEEE Trans. Software Eng. **49**(3), 1102–1117 (2023). https://doi.org/10.1109/TSE.2022.3171404, https://doi.org/10.1109/TSE.2022.3171404
22. John, S., Burdusel, A., Bill, R., Strüber, D., Taentzer, G., Zschaler, S., Wimmer, M.: Searching for optimal models: Comparing two encoding approaches. J. Object Technol. **18**(3), 6:1–22 (2019). https://doi.org/10.5381/jot.2019.18.3.a6
23. John, S., Kosiol, J., Lambers, L., Taentzer, G.: A graph-based framework for model-driven optimization facilitating impact analysis of mutation operator properties. Softw. Syst. Model. **22**(4), 1281–1318 (2023). https://doi.org/10.1007/S10270-022-01078-X, https://doi.org/10.1007/s10270-022-01078-x
24. John, S., Kosiol, J., Taentzer, G.: Towards a configurable crossover operator for model-driven optimization. In: Kühn, T., Sousa, V. (eds.) Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings, MODELS 2022, Montreal, Quebec, Canada, October 23–28, 2022. pp. 388–395. ACM (2022). https://doi.org/10.1145/3550356.3561603
25. Kosiol, J., Fritsche, L., Nassar, N., Schürr, A., Taentzer, G.: Constructing constraint-preserving interaction schemes in adhesive categories. In: Fiadeiro, J.L., Tutu, I. (eds.) Recent Trends in Algebraic Development Techniques – 24th IFIP WG 1.3 International Workshop, WADT 2018, Egham, UK, July 2–5, 2018, Revised Selected Papers. Lecture Notes in Computer Science, vol. 11563, pp. 139–153. Springer (2019). https://doi.org/10.1007/978-3-030-23220-7_8, https://doi.org/10.1007/978-3-030-23220-7_8
26. Kosiol, J., John, S., Taentzer, G.: A generic construction for crossovers of graph-like structures and its realization in the eclipse modeling framework. J. Log. Algebraic Methods Program. **136**, 100909 (2024).

https://doi.org/10.1016/J.JLAMP.2023.100909, https://doi.org/10.1016/j.jlamp.2023.100909

27. Kosiol, J., Strüber, D., Taentzer, G., Zschaler, S.: Sustaining and improving graduated graph consistency: A static analysis of graph transformations. Sci. Comput. Program. **214**, 102729 (2022). https://doi.org/10.1016/j.scico.2021.102729
28. Kräuter, T., Rutle, A., König, H., Lamo, Y.: Formalization and analysis of BPMN using graph transformation systems. In: Fernández, M., Poskitt, C.M. (eds.) Graph Transformation - 16th International Conference, ICGT 2023, Held as Part of STAF 2023, Leicester, UK, July 19-20, 2023, Proceedings. Lecture Notes in Computer Science, vol. 13961, pp. 204–222. Springer (2023). https://doi.org/10.1007/978-3-031-36709-0_11, https://doi.org/10.1007/978-3-031-36709-0_11
29. Lauer, A., Kosiol, J., Taentzer, G.: Empowering model repair: A rule-based approach to graph repair without side effects. In: ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, MODELS 2023 Companion, Västerås, Sweden, October 1–6, 2023. pp. 831–840. IEEE (2023). https://doi.org/10.1109/MODELS-C59198.2023.00132, https://doi.org/10.1109/MODELS-C59198.2023.00132
30. Martinez, J., Strüber, D., Horcas, J.M., Burdusel, A., Zschaler, S.: Acapulco: an extensible tool for identifying optimal and consistent feature model configurations. In: Felfernig, A., Fuentes, L., Cleland-Huang, J., Assunção, W.K.G., Quinton, C., Guo, J., Schmid, K., Huchard, M., Ayala, I., Rojas, J.M., Le, V., Horcas, J.M. (eds.) SPLC '22: 26th ACM International Systems and Software Product Line Conference, Graz, Austria, September 12 – 16, 2022, Volume B. pp. 50–53. ACM (2022). https://doi.org/10.1145/3503229.3547067, https://doi.org/10.1145/3503229.3547067
31. Narayanan, A., Karsai, G.: Towards verifying model transformations. In: Bruni, R., Varró, D. (eds.) Proceedings of the Fifth International Workshop on Graph Transformation and Visual Modeling Techniques, GT-VMT@ETAPS 2006, Vienna, Austria, April 1–2, 2006. Electronic Notes in Theoretical Computer Science, vol. 211, pp. 191–200. Elsevier (2006). https://doi.org/10.1016/J.ENTCS.2008.04.041, https://doi.org/10.1016/j.entcs.2008.04.041
32. Nassar, N., Kosiol, J., Arendt, T., Taentzer, G.: Constructing optimized constraint-preserving application conditions for model transformation rules. J. Log. Algebraic Methods Program. **114**, 100564 (2020). https://doi.org/10.1016/j.jlamp.2020.100564
33. Nassar, N., Kosiol, J., Radke, H.: Rule-based Repair of EMF Models: Formalization and Correctness Proof. In: Graph Computation Models (GCM 2017), Electronic Pre-Proceedings (2017), pages.di.unipi.it/corradini/Workshops/GCM2017/papers/Nassar-Kosiol-Radke-GCM2017.pdf
34. OMG: Business process model and notation. version 2.0 (2011), http://www.omg.org/spec/BPMN/2.0/
35. Rangel, G., Lambers, L., König, B., Ehrig, H., Baldan, P.: Behavior preservation in model refactoring using DPO transformations with borrowed contexts. In: Ehrig, H., Heckel, R., Rozenberg, G., Taentzer, G. (eds.) Graph Transformations, 4th International Conference, ICGT 2008, Leicester, United Kingdom, September 7–13, 2008. Proceedings. Lecture Notes in Computer Science, vol. 5214, pp. 242–256. Springer (2008). https://doi.org/10.1007/978-3-540-87405-8_17, https://doi.org/10.1007/978-3-540-87405-8_17

36. Reijers, H.A., Vanderfeesten, I.T.: Cohesion and coupling metrics for workflow process design. In: International Conference on Business Process Management. pp. 290–305. Springer (2004)
37. Sandmann, C., Habel, A.: Rule-based graph repair. In: Echahed, R., Plump, D. (eds.) Proceedings Tenth International Workshop on Graph Computation Models, GCM@STAF 2019, Eindhoven, The Netherlands, 17th July 2019. EPTCS, vol. 309, pp. 87–104 (2019). https://doi.org/10.4204/EPTCS.309.5, https://doi.org/10.4204/EPTCS.309.5
38. Schmidt, D.C.: Guest editor's introduction: Model-driven engineering. Computer **39**(2), 25–31 (2006). https://doi.org/10.1109/MC.2006.58
39. Schneider, S., Lambers, L., Orejas, F.: A logic-based incremental approach to graph repair featuring delta preservation. Int. J. Softw. Tools Technol. Transf. **23**(3), 369–410 (2021). https://doi.org/10.1007/S10009-020-00584-X, https://doi.org/10.1007/s10009-020-00584-x
40. Schneider, S., Maximova, M., Giese, H.: Invariant analysis for multi-agent graph transformation systems using k-induction. In: Behr, N., Strüber, D. (eds.) Graph Transformation – 15th International Conference, ICGT 2022, Held as Part of STAF 2022, Nantes, France, July 7–8, 2022, Proceedings. Lecture Notes in Computer Science, vol. 13349, pp. 173–192. Springer (2022). https://doi.org/10.1007/978-3-031-09843-7_10, https://doi.org/10.1007/978-3-031-09843-7_10
41. Seada, H., Deb, K.: A unified evolutionary optimization procedure for single, multiple, and many objectives. IEEE Trans. Evol. Comput. **20**(3), 358–369 (2016). https://doi.org/10.1109/TEVC.2015.2459718, https://doi.org/10.1109/TEVC.2015.2459718
42. Sudholt, D.: How crossover speeds up building block assembly in genetic algorithms. Evolutionary Computation **25**(2), 237–274 (2017). https://doi.org/10.1162/EVCO_a_00171
43. Thölke, H., Kosiol, J.: A multiplicity-preserving crossover operator on graphs. In: Kühn, T., Sousa, V. (eds.) Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings, MODELS 2022, Montreal, Quebec, Canada, October 23–28, 2022. pp. 588–597. ACM (2022). https://doi.org/10.1145/3550356.3561587, https://doi.org/10.1145/3550356.3561587
44. Van Der Aalst, W.M.: Business process management demystified: A tutorial on models, systems and standards for workflow management. Springer (2004)
45. Vanderfeesten, I., Cardoso, J., Mendling, J., Reijers, H.A., Van der Aalst, W.: Quality metrics for business process models. BPM and Workflow handbook **144**(2007), 179–190 (2007)
46. Vergidis, K., Saxena, D.K., Tiwari, A.: An evolutionary multi-objective framework for business process optimisation. Appl. Soft Comput. **12**(8), 2638–2653 (2012). https://doi.org/10.1016/J.ASOC.2012.04.009, https://doi.org/10.1016/j.asoc.2012.04.009
47. Vergidis, K., Tiwari, A., Majeed, B.: Business process analysis and optimization: Beyond reengineering. IEEE Trans. Syst. Man Cybern. Part C **38**(1), 69–82 (2008). https://doi.org/10.1109/TSMCC.2007.905812, https://doi.org/10.1109/TSMCC.2007.905812
48. Zschaler, S., Mandow, L.: Towards model-based optimisation: Using domain knowledge explicitly. In: Milazzo, P., Varró, D., Wimmer, M. (eds.) Software Technologies: Applications and Foundations – STAF 2016 Collocated Workshops: DataMod,

GCM, HOFM, MELO, SEMS, VeryComp, Vienna, Austria, July 4–8, 2016, Revised Selected Papers. Lecture Notes in Computer Science, vol. 9946, pp. 317–329. Springer (2016). https://doi.org/10.1007/978-3-319-50230-4_24