Generating Examples for Knowledge Abstraction in MDE: a Multi-Objective Framework

Edouard Batot

DIRO, Université de Montréal, Canada
Email: batotedo@iro.umontreal.ca

***Abstract*—Model-Driven Engineering (MDE) aims at raising the level of abstraction in software development and therefore relies on task automation. To foster automation, MDE promotes the use of specific domain languages (DSLs), essential to express ideas at the domain level. Furthermore, to ease communication between computer science and other fields, modelers employ model examples (*i.e.,* selected metamodel instances) to illustrate and refine their conceptual ideas. But, if the use of model examples has shown its efficiency, it is still an *ad hoc* process which requires automation.**

In this paper, we briefly depict the thorough example-to-knowledge learning process. Then, we present a framework that produces, from a metamodel, a representative model example set with regards to a given coverage definition. To find the best trade-off between coverage and a necessary minimality objectives, we use a non-dominated genetic algorithm (NSGA-II). We illustrated our method by generating a near-optimal set of models for the peculiar constraint learning task. We evaluated its efficiency comparing the resulting generated set with the best one issued from a raw random generation.

Our encouraging preliminary results let us envision a deep study of the relation between various types of coverage and their impact on our ability to abstract knowledge from examples.

I. Introduction

Model-Driven Engineering (MDE) aims at raising the level of abstraction in software development. It promotes the use specific domain languages (DSLs) that can help non computer scientists to model their applications and rely on automation for development and maintenance tasks, such as model transformation and code generation, testing, etc. However, in addition to the definition of DSLs, most of these tasks are themselves domain-dependent and require specific knowledge for their automation. In this respect and as acknowledged by Selic [1], technical issues, especially the lack of automation, are significant obstacles to the wide adoption of MDE.

Another kind of issue pointed out by Selic is communication among experts from different fields. To cope with the need of domain knowledge, many examples-based techniques were proposed to enhance automation in MDE. Examples are used either for knowledge elicitation from experts as in [2], or for learning this knowledge as in [3] and [4]. They are used as a vehicle to capture domain expertise when explicit knowledge is not available or difficult to communicate.

We view the learning of knowledge, to automate a MDE task, from examples as a 3-steps process as depicted in Fig. 1. The first step consists in generating/selecting automatically a set of model examples from a partial/complete metamodel. Then for the second step, depending on the task to automate, these examples have to be completed/corrected/annotated by the domain experts. For instance, for model transformation learning [5], source models are first automatically generated. Then, a domain expert can produce the corresponding target models. Similarly, for meta-model well-formedness rule (WFR) learning [4], the model examples, which are automatically generated, are labeled by the domain expert as valid or invalid. Model generation automation improvement tackles technical issues in model validation and model transformation. Finally, the third step consists in the actual knowledge learning using the examples defined in the two first step. Here, the task to automate defines the boundaries of the space of possible solutions.

In this paper, we focus on the first step of the example-to-knowledge learning process. Indeed, the quality of the learned knowledge depends heavily on the quality of the generated/-completed examples. Obviously, the examples should have a good coverage to make the learned knowledge more generalizable. In other words, the examples should be representative of the various situations that the automated task should process. Of course, the notion of coverage is not absolute and depends on the task to automate. Thus, different coverage criteria could be defined. The most obvious coverage is the one considering the entire modeling space defined by a metamodel. This coverage is needed for metamodel testing [6].

Fleurey *et al.* [7] describe possible strategies to cover such a modeling space. Then, the modeling space to explore can be reduced depending on the tasks to automate. For instance, in (WFR) learning, not all the metamodel constructs are likely to be concerned with well-formedness rules. Cadavid *et al.* [8] showed empirically that these rules follow a limited set of OCL templates. Thus, only the metamodel fragments that are involved in instance of these templates should be covered by the model generation. Similarly, for transformation learning, the metamodel fragments to cover are only those, potentially concerned by the transformation (see the work of Mottu *et al.* [9]). These are three examples of coverage that could be used separately or together to maximize the representativeness of the model examples.

In model generation, coverage is not the only criterion to consider. Due to the manual completion by the domain experts and the learning computation cost, the size of the model example set to generate should be limited. Thus a minimality criterion is also important to consider.

Contribution?

As mentioned earlier, the generation of model examples should be driven by many objectives: one or more types of coverage and minimality. Coverage and minimality are generally conflicting criteria and are difficult to combine in a single objective. In this paper, we propose a framework for model example set generation in the form of a multi-objective evolutionary algorithm. Our framework is generic as it can be applied to any metamodel and one or more coverage criteria can be selected.

The remainder of the paper is organized as follows. In Section II, we briefly describe research contributions on the edge between model generation and example-based automation in MDE. In Section III, we present our approach. Our preliminary results are outlined in Section IV and we conclude in Section V.

II. Related work

Recent studies have demonstrated the utility of using example models through the modeling process [3], [2]. Our work is related to three main areas: generation of test models, knowledge elicitation for modeling and knowledge learning from model examples.

Model generation, which is performed with the intent of providing tests cases, is the most target area. A plethora of papers has been published including new paradigms such as test-driven [10], behavior-driven [11] and story test-driven [12]. With the intent of providing examples to feed a testing task, automation here is circumscribed into generating models rather than defining them manually. Wu *et al.* [13] report on a systematic literature review about metamodel instance (*i.e.,* models) generation. They gather a total of 34 studies from 2002 to 2011, and present four main areas related to instance generation: compiler testing, model transformation testing, graph grammar and SAT-based approaches. Wu *et al.*found that there is no consensus on algorithms and theoretical frameworks used in each area to produce metamodel instances. Ways and means to generate models are problem-dependent but the criteria used for selecting model instances are, for the most part, based on Association-end multiplicities and Class attribute utilization and refer to Fleurey *et al.* [7] as the state-of-the-art in the area. [14] explores strategies based on multi-formalism knowledge to automate model generation for model transformation testing.

The second related area is dedicated to using examples for knowledge elicitation during modeling activities. Here, automation helps to emphasis interesting part of the task to automate and to exhibit unexpected corner cases. For instance, in [15], a metamodel is induced and refactored in an interactive way using examples to help in communication and incrementally enhance its accuracy. Fleurey *et al.*in [7] characterize models to test metamodels but do not derive knowledge from them. The focus is on the qualification of sets as in [16] where the authors aim at characterizing oracles in model transformation testing. Additionally, Mottu *et al.* [9] provide suitable test cases for model transformations based on the transformation footprint. They improve testing but do not automate the testing process, nor they derive knowledge from these cases. In the same fashion, Cadavid *et al.* in [6] explore modeling space boundaries using examples.

Finally, many contributions have been proposed recently to derive knowledge from examples in various Software Engineering fields. In this area, attention is paid to understand the various situations put to light by the examples, and to abstract knowledge from them. Examples of such contributions include WFR learning [4] and model transformation derivation [17] from examples. Another example is the detection and correction of design defects [18] using examples.

III. Approach

Fig. 1 shows how we envision the example-based knowledge abstraction to automate MDE tasks that are domain-dependent. Our process includes three steps driven by the task to automate. The task to automate defines the kind of coverage that will be considered in the model generation to ensure the representativeness of the generated example set. During the second step, the task to automate specifies which information has to be provided by the expert to complete the generated examples. Finally, the task to automate demarcates the solution space to be explored to learn the necessary knowledge in the last step. In this space, the quality of the candidate solutions is evaluated by their conformance to the examples, which are built in the two first steps.

This paper details our contribution to the first step of the process, i.e., generation of models guided by the selected coverage types. More concretely, we propose a model generation framework that allows to select the metamdodel to instantiate. Any metamodel expressed in Ecore can be used. The framework also offers an extensible set of coverage types from which one or more types have to be satisfied during the generation. The framework takes the form of a multi-objective evolutionary algorithm. In this context, adding a new coverage type consists in defining a new objective to evaluate.

The generation process is viewed as a search-based combinatorial optimization to find the minimal set of models that best satisfy one or more coverage types for a selected metamodel. More concretely, we use a non-sorting genetic algorithm (NSGA-II [19]) to tackle the theoretical challenge of optimizing the representativeness of model example sets. After presenting the main process of evolutionary algorithm, we precise how we adjusted it to the context of coverage achievement. A last sub-section will present briefly the second and third step of the learning process.

A. Non-sorting genetic algorithm

Genetic algorithms (GA) are powerful tools to explore the search space of an optimization problem. More specifically, the basic idea of NSGA-II [19] is to make a population of candidate solutions evolve toward the near-optimal solution in order to solve a multi-objective optimization problem. NSGA-II is designed to find a set of optimal solutions, called

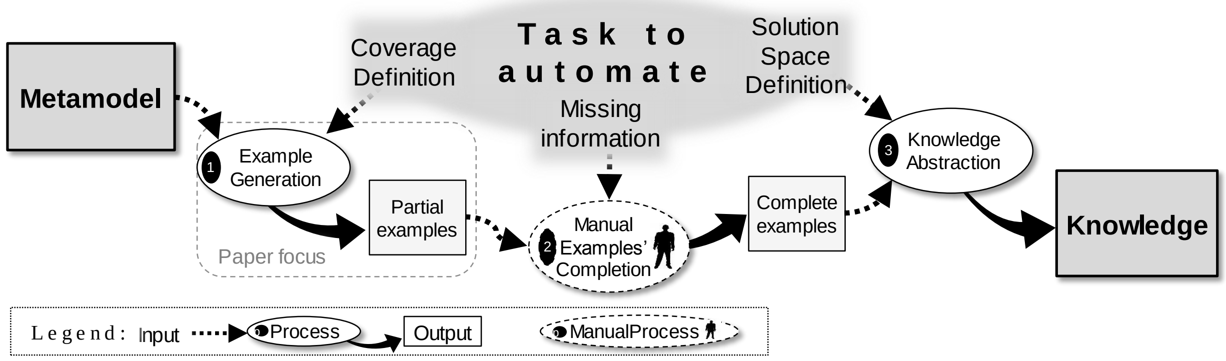

Fig. 1: Example-based knowledge abstraction in MDE

non-dominated solutions, also Pareto set. A non-dominated solution is the one which provides a suitable compromise between all objectives without degrading any of them. As described in Fig. 2, the first step in NSGA-II is to create randomly a population P_0 of $N/2$ individuals encoded using a specific representation. Then, a child population Q_0, of the same size, is generated from the population of parents P_0 using genetic operators such as crossover and mutation. Both populations are merged into an initial population R_0 of size N, which is sorted into dominance fronts according to the dominance principle. A solution s_1 dominates a solution s_2 for a set of objectives $\{O_i\}$ if $\forall i, O_i(s_1) \geqslant O_i(s_2)$ and $\exists j \mid O_j(s_1) > O_j(s_2)$. The first front includes the non-dominated solutions; the second front contains the solutions that are dominated only by the solutions of the first front, and so on and so forth. The fronts are included in the parent population P_1 of the next generation following the dominance order until the size of $N/2$ is reached. If this size coincides with part of a front, the solutions inside this front are sorted, to complete the the population, according to a crowding distance which favors diversity in the solutions [19]. This process will be repeated until a stop criterion is reached, e.g., a number of iterations.

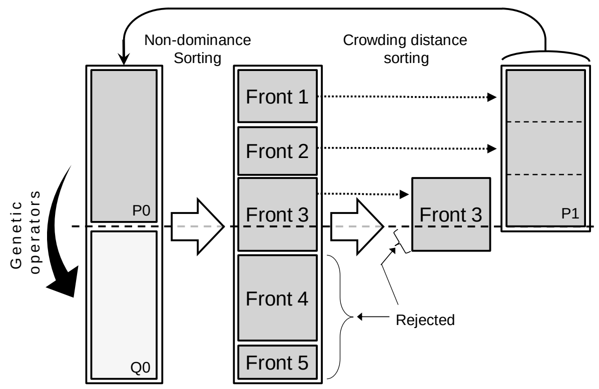

Fig. 2: NSGA-II [19]

B. Model example generation

When applying NSGA-II to a particular problem, solution representation, genetic operators, and objective functions must be carefully designed to address the problem requirements. The creation of the initial population has also to be defined. The next sections present our adaptation of NSGA-II to the context of model generation.

1) Solution representation and solution creation: As our goal is to maximize the coverage of a metamodel or some fragments of it, with a minimal overlap between the selected models, a solution for our problem refers simply to a set of models. We view the model example generation as the selection of the minimal set of models inside a large amount of randomly generated ones (call it the base of models). Our first step is to produce the base of models for the considered metamodel. To this end, we use AtlanMod instantiator [1]. This tool allows to generate models in XMI format from a meta-model described in ECORE. The expected number of models, number of objects per model, maximum number of attributes and references, and maximum depth of the references can be specified. The generator produces the required number of correct instances (invalid instances, w.r.t multiplicity constraints, identified by the tool are ignored). It is configured with a uniform distribution, i.e., when a maximum number is given for attributes/references/depth, any number below the maximum has the same probability to occur. We repeat the generation process with different model sizes to produce a large base of examples (10000 models of 10 to 200 objects in our evaluation study).

2) Objective functions: The objective functions assess the ability of a solution to solve the problem under consideration. To evaluate solutions (*i.e.,* model sets), we consider the coverage and minimality objectives. Coverage, as depicted in Fig. 1, has to be tailored to the task to automate. For illustration purpose, we consider the metamodel coverage for metamodel testing (*i.e., at large*) and the *OCL footprint* coverage for well-formedness rule learning. We present, in the following

[1]https://github.com/atlanmod/mondo-atlzoo-benchmark/tree/master/fr.inria.atlanmod.instantiator

paragraphs, the implementations of both coverage evaluation strategies as well as minimality.

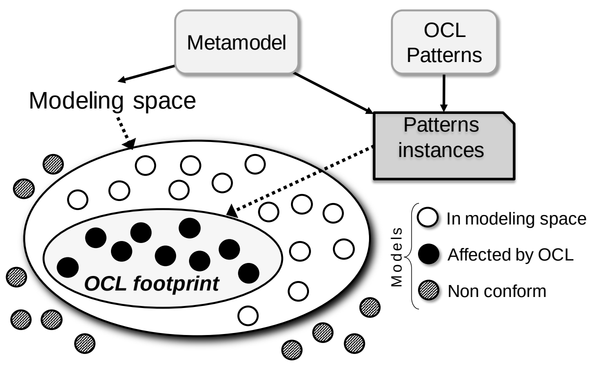

Fig. 3: Coverage tailored to WFR learning : *OCL footprint*

Coverage *at large*: Metamodel testing implies exploring the entire metamodel. Fleurey *et al.* [7] is considered as the state-of-the-art in partitioning metamodel in order to distillate interesting structures from its features and rate coverage upon these. Based on Ostrand *et al.* [20] category-partitioning method, the authors decompose the static structure of a metamodel in three hierarchical levels:

- an *object fragment* is the association of a possible value (or a range of values) to an attribute (or a reference) in the metamodel. To this end, each attribute or reference is partitioned, beforehand, into a set of (ranges of) values. For example, an integer-type attribute P of a class C is partitioned into three categories: {P=0, P=1, P>1}. For a given metamodel, an object fragment is defined for each category of the partition of each attribute/reference, e.g., $of(P, 0)$.
- a *model fragment* contains one or more object fragments. We considered two strategies to define the model fragments. For the *AllRanges* strategy, a model fragment is defined for each object fragment, e.g., $mf((P, 0))$. For the *AllPartitions* strategy, we define a model fragment for each attribute/reference as a set of object fragments of the corresponding partition, e.g., $mf((P, 0),(P, 1),(P, > 1))$.
- a *metamodel fragment partition MFP* is the set of all model fragments defined on the metamodel.

For a given model instance m_i, a model fragment mf_j is covered by m_i if all the object fragments in mf_j appear in m_i ($covering(mf_j, m_i) = true$). Starting from this, it is possible to derive the set of model fragments covered $MFC(ms)$ by a set of models ms (a solution in our problem). Then, our coverage objective function is defined as the proportion of model fragments covered by the candidate solution ms over the metamodel fragment partition MFP. Formally:

$$coverage(ms) = \frac{|MFC(ms)|}{|MFP|}$$

OCL *footprint*: the second coverage evaluation we implemented is based on the intent to focus on parts of the metamodel potentially affected by OCL constraints - *i.e.,* OCL footprint. Cadavid *et al.*in [21] perform an empirical study on the conjunct use of MOF and OCL. They collected from an important set of metamodels (from industry, academia and the Object Management Group[2]) a list of the most used OCL well-formedness constraints. They observed that all the metamodels tend to have a small core subset of concepts, which are constrained by most of the rules. However, the most interesting conclusion, from our perspective, is that there is a limited set of well-formedness rule patterns (22) that are the most used by metamodelers. Instances of these patterns are used separately or composed to define well-formedness rules in the form of OCL constraints. We use this as a base to evaluate the OCL footprint on a metamodel.

For that purpose, we identify all instances of each OCL pattern in the considered metamodel. For each instance, we identify the involved classes and features. The identified elements of the metamodel determine the parts that have to be covered by the model example sets. In other words, rather than defining the model fragments and the metamodel fragment partition from the whole metamodel, we consider only the elements involved in the constraint-patterns instances. Fig. 3 illustrates the idea of considering only the pattern instances. The coverage calculation of a set of models ms follows the formula as for the coverage at large.

Minimality: As our goal is to find the minimal set of models with the maximum coverage, we aims at having sets with dissimilar models. More concretely, to ensure minimality, we use the dissimilarity criterion as defined in [6]. The dissimilarity of a set ms is calculated as $1 - sim(ms)$ where $sim(ms)$ is the normalized similarity between models in ms.

3) Genetic operators: As illustrated in Fig. 4, *crossover operator* uses the single cut-point crossover. Each parent solution (set of models) is divided into two parts according to a randomly picked cut point. Then the parts of the parent solutions are exchanged to form two new model sets.

Mutation operator selects randomly a model in a model set and replaces it by a new model randomly selected in the model base.

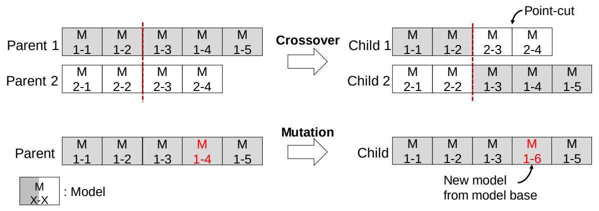

Fig. 4: Crossover and mutation adapted to coverage evaluation

C. Following step: Completion For illustrative purpose

Once the near-optimal example set constituted, experts may add missing information to complete the examples. As said before, in the case of WFR learning, example models have

[2]http://www.omg.org

to be discriminated by an expert who will annotate them as conform (or not conform) to his application domain. Labeled examples will allow the learning process of WFR.

For the case of model transformation learning, the source model examples, which are generated by our evolutionary algorithm, will be completed with their corresponding target models.

IV. Validation

Assessing our entire methodology would require a cross-comparison between metamodels from different ranges and different coverage kinds. To give a first look at the efficiency of our approach, we compared it to a random model set generation for two different metamodels using the OCL-footprint coverage. In this section, we present the setup of our validation procedure and then the first results obtained.

A. Setup

As we are using a fixed model base to select the representative model sets, we have to ensure the diversity of the models in our base. Therefore, we run the Instantiator to produce 10.000 models with different structural characteristics. During the generation, we varied the expected size of models from 10 to 200 objects (with steps of 10), and we picked randomly the numbers of attributes and references in a range of 0 to 14. The depth of reference chains is picked between 0 and 10. We took those numbers from common knowledge about the statistical structure of metamodels and its correlation with the practical use. These parameters can be changed in our framework. In the end, the 10.000 models generated are stored in the base from which the initial population of solutions is created and from which the models are randomly picked for the mutation operator.

We implemented two different coverage evaluations depending on the nature of the problem: partitioning the whole metamodel; and using the knowledge-based partitioning (*OCL footprint*) following the OCL patterns identified in [21].

Results of the comparison between a random generation and our approach are shown in Table I. Random generation builds random populations from models picked in the model base. To have a fair comparison, the random generation produces 50×800 model sets, which corresponds to the number of model sets explored by our approach (800 generations of 50 model sets each). The size of the set is randomly given for each iteration of the random generation and for the initial population in NSGA-II algorithm.

	ATL2.0		**Feature Diagram**	
	Random	NSGA-II	Random	NSGA-II
Dissimilarity	99%	99%	63%	95%
Coverage	78%	93%	100%	100%

TABLE I: Comparison between random generation and NSGA-II algorithm (in WFR learning)

To assess our process, we executed the two algorithms on two different metamodels: a small metamodel, Feature Diagram which contains 5 classes and 8 features; and ATL2.0, a metamodel of more considerable size with 84 classes and 146 features.

B. Results

Following our concern of having a fair comparison, we ran the random generation and our algorithm several times and recorded for each run the best solution for each algorithm. Then, we considered, for comparison, the best recorded solution for random and the worst recorded solution for our algorithm.

As described in Table I, our algorithm produces a solution that dominates the random solution for the two considered objectives. For ATL2.0, which is the largest metamodel, the dissimilarity quickly reaches a maximum of 99% in both algorithms whereas the coverage is by far better for our algorithm (93% compared to 78% for random execution). In the case of the Feature Diagram, the smallest metamodel, we observed the opposite. Both algorithms have a full coverage (100%). This is easy to explain considering the size of the metamodel and thus, the small size of metamodel fragment partition to cover. However, this good coverage was obtained in random at the cost of a low dissimilarity 63%, which was not the case for our algorithm with a dissimilarity of 95%.

We could conclude that these preliminary results are very encouraging for the effectiveness of our approach in generating minimal and representative sets of model examples. They strengthen our motivation in pursuing the investigation of the other steps of the example-to-knowledge learning process.

V. Conclusion

In this paper, we presented an approach to automatically generate example models for a given metamodel in order to abstract knowledge from them. We model the example generation as multi-objective optimization problem, and we solve it using an evolutionary algorithm NSGA-II. To evaluate our approach, we compared its results to those produced by a random generation. Our empirical observations show that we produce better solutions for the two considered metamodels.

An important perspective of this work consists in automating/characterizing the process of learning knowledge from examples in MDE. The framework we presented in this paper allows, from a metamodel, to produce a representative model example set with regards to a given coverage definition. Examples need then to be completed by an expert to have a fully-fledged set of examples that can be used to learn the knowledge necessary to automate a task in MDE. Future work includes first the study of the relation between various types of coverage and our ability to learn knowledge from examples. Then, we can explore the learning of various tasks in MDE.

References

[1] B. Selic, "What will it take? A view on adoption of model-based methods in practice," *Software and System Modeling*, vol. 11, no. 4, pp. 513–526, 2012.

[2] D. Zayan, M. Antkiewicz, and K. Czarnecki, “Effects of using examples on structural model comprehension: a controlled experiment,” in *36th International Conference on Software Engineering, ICSE ’14, Hyderabad, India - May 31 - June 07, 2014*, 2014, pp. 955–966.

[3] K. Bąk, D. Zayan, K. Czarnecki, M. Antkiewicz, Z. Diskin, A. Wąsowski, and D. Rayside, “Example-driven modeling: Model = abstractions + examples,” in *Proceedings of the 2013 International Conference on Software Engineering*, ser. ICSE ’13. IEEE Press, 2013, pp. 1273–1276.

[4] M. Faunes, J. Cadavid, B. Baudry, H. Sahraoui, and B. Combemale, “Automatically searching for metamodel well-formedness rules in examples and counter-examples,” in *Model-Driven Engineering Languages and Systems*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2013, vol. 8107, pp. 187–202.

[5] M. Faunes, H. Sahraoui, and M. Boukadoum, “Genetic-programming approach to learn model transformation rules from examples,” in *Theory and Practice of Model Transformations*, ser. Lecture Notes in Computer Science, K. Duddy and G. Kappel, Eds. Springer Berlin Heidelberg, 2013, vol. 7909, pp. 17–32.

[6] J. J. Cadavid and H. A. Baudry, Benoit Sahraoui, “Searching the boundaries of a modeling space to test metamodels,” in *2012 IEEE Fifth International Conference on Software Testing, Verification and Validation, Montreal, QC, Canada, April 17-21, 2012*, 2012, pp. 131–140.

[7] F. Fleurey, B. Baudry, P.-A. Muller, and Y. Le Traon, “Towards dependable model transformations: Qualifying input test data,” *Journal of Software and Systems Modeling (SoSyM)*, vol. 8, no. 2, pp. 185–203, 2009.

[8] J. J. Cadavid, B. Combemale, and B. Baudry, “An analysis of metamodeling practices for MOF and OCL,” *Computer Languages, Systems and Structures*, vol. 41, no. 0, pp. 42 – 65, 2015.

[9] J. Mottu, S. Sen, M. Tisi, and J. Cabot, “Static analysis of model transformations for effective test generation,” in *23rd IEEE International Symposium on Software Reliability Engineering, ISSRE 2012, Dallas, TX, USA, November 27-30, 2012*, 2012, pp. 291–300.

[10] D. Janzen and H. Saiedian, “Test-driven development: Concepts, taxonomy, and future direction,” *Computer*, vol. 38, no. 9, pp. 43–50, 2005.

[11] D. North, “Introducing bdd,” *Better Software*, vol. 12, 2006.

[12] S. S. Park, “Communicating domain knowledge through example-driven story testing,” Ph.D. dissertation, Citeseer, 2011.

[13] H. Wu, R. Monahan, and J. F. Power, “Metamodel instance generation: A systematic literature review,” *CoRR*, vol. abs/1211.6322, 2012.

[14] S. Sen, B. Baudry, and J. Mottu, “Automatic model generation strategies for model transformation testing,” in *Theory and Practice of Model Transformations*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2009, vol. 5563, pp. 148–164.

[15] J. J. Lopez-Fernandez, J. S. Cuadrado, E. Guerra, and J. de Lara, “Example-driven meta-model development,” *Software & Systems Modeling*, pp. 1–25, 2013.

[16] O. Finot, J. Mottu, G. Sunyé, and T. Degueule, “Using meta-model coverage to qualify test oracles,” in *Proceedings of the Second Workshop on the Analysis of Model Transformations (AMT) 2013, Miami, FL, USA, September 29, 2013*, 2013.

[17] I. Baki, H. Sahraoui, Q. Cobbaert, P. Masson, and M. Faunes, “Learning implicit and explicit control in model transformations by example,” in *Model-Driven Engineering Languages and Systems*, ser. Lecture Notes in Computer Science. Springer International Publishing, 2014, vol. 8767, pp. 636–652.

[18] M. Kessentini, W. Kessentini, H. Sahraoui, M. Boukadoum, and A. Ouni, “Design defects detection and correction by example,” in *Program Comprehension (ICPC), 2011 IEEE 19th International Conference on*, June 2011, pp. 81–90.

[19] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, “A fast elitist non-dominated sorting genetic algorithm for multi-objective optimisation: NSGA-II,” in *Parallel Problem Solving from Nature - PPSN VI, 6th International Conference, Paris, France, September 18-20, 2000, Proceedings*, 2000, pp. 849–858.

[20] T. J. Ostrand and M. J. Balcer, “The category-partition method for specifying and generating fuctional tests,” *Commun. ACM*, vol. 31, no. 6, pp. 676–686, 1988.

[21] J. Cadavid, B. Combemale, and B. Baudry, “Ten years of Meta-Object Facility: an Analysis of Metamodeling Practices,” AtlanMod, Research Report RR-7882, 2012.