Towards Approximate Model Transformations

Javier Troya[1], Manuel Wimmer[1], Loli Burgueño[2], and Antonio Vallecillo[2]

[1] Business Informatics Group, Vienna University of Technology, Austria
{troya,wimmer}@big.tuwien.ac.at
[2] ETSI Informática, Universidad de Málaga, Spain
{loli,av}@lcc.uma.es

Abstract. As the size and complexity of models grow, there is a need to count on novel mechanisms and tools for transforming them. This is required, e.g., when model transformations need to provide target models without having access to the complete source models or in really short time—as it happens, e.g., with streaming models—or with very large models for which the transformation algorithms become too slow to be of practical use if the complete population of a model is investigated.
In this paper we introduce *Approximate Model Transformations*, which aim at producing target models that are accurate enough to provide meaningful and useful results in an efficient way, but without having to be fully correct. So to speak, this kind of transformations treats accuracy for execution performance. In particular, we redefine the traditional OCL operators used to query models (e.g., allInstances, select, collect, etc.) by adopting sampling techniques and analyse the accuracy of approximate model transformations results.

Keywords: Model Transformation, Approximation, Performance, Sampling

1 Introduction

Model transformations (MTs) are gaining acceptance as model-driven techniques are becoming commonplace. While models capture the views on systems for particular purposes and at given levels of abstraction, MTs are in charge of the manipulation, analysis, synthesis, and refinement of the models [3]. They do not only allow the generation of implementations from high-level models, but also to generate other views that can be properly analyzed or that provide users with the information they need, at the right level of abstraction, e.g., a synopsis of a larger data set.

So far the community has mainly focused on the *correct* implementation of a MT, according to its specification [4, 14–16, 30, 32], although there is an emergent need to consider other (non-functional) aspects such as performance, scalability, usability, maintainability and so forth [5]. In particular, the study of the performance of MTs is gaining interest as very large models living in the cloud have to be transformed as well [8, 28, 29]. The usual approach to improve performance has focused on the use of incremental execution [7, 22] and parallelization techniques [8, 28].

In this paper we explore a different path. Our aim is to weaken the need to produce *exact* target models but *approximate* ones. Such approximate target models should be accurate enough to provide meaningful and useful results to users, but alleviate the

need for the transformation to generate fully correct models—being able to produce such target models in much shorter time. We call *Approximate Model Transformations* (AMT) those model transformations that produce *approximate* target models staying inside a given error bound. For this, we investigate the adoption of statistical sampling techniques, such as they are employed in the database area [1, 10, 19] and in the general field of approximate computing [34].

This kind of MTs are needed in various circumstances. The most obvious situation is when very large models have to be synthesized into much smaller models for decision making, the synthesis process is not trivial, the decision should be made really fast, and near-optimal solutions or just accurate results are enough (without having to be fully correct). Another situation in which AMTs come also into play is when we cannot fully guarantee the correctness of the source models. For example, when dealing with infinite [9] or streaming models [11]: only a portion of them (e.g., the ones inside the *sliding window*) is available at any given moment in time. This means that some connections between model elements may be broken because not all the elements involved in these connections are available at the same time. By eliminating the need for the source model to be correct in this case (i.e., the portion we are considering is just a fragment [6], which does not need to conform to the source metamodel), we cannot guarantee that the transformation will produce a correct target model. As an example, think, e.g., of several wireless sensor networks sending information that needs to be semantically annotated, combined, and analyzed to make decisions about traffic routes [27]. Here the transformation rules in charge of distilling the information need to be somehow simplified in order to be fast—at the cost of sacrificing correctness.

There are several issues involved in these kinds of MTs. First, we need to provide a measure of the *accuracy* of the target model produced, what we do based on the confidence level and relative error, to be able to decide how good the results given by the approximate transformation are after analyzing them, and if they are good enough for the user's purposes. Second, we need to redefine in this context some of the traditional operators used to traverse or query the models: *allInstances()*, *select()*, *collect()*, etc. Third, we need to identify adequate methods for the design of approximate model transformations that provide accurate-enough results. Finally, we need to be able to identify specific scenarios where it makes sense to apply this kind of MTs.

The structure of the paper is as follows. Section 2 sets the context by presenting a motivating example used throughout the paper to illustrate our ideas. Section 3 presents the concept of AMT and describes the ideas for our approach, which are applied in the case study as presented in Section 4. Then, in Section 5 we discuss related work, and in Section 6 we conclude the paper with an outlook on future work.

2 Motivating Example

Let us consider a real-world example of a Wireless Sensor Network (WSN) for measuring different climatological conditions. The metamodel that we show in Fig. 1 has been extracted from the data [2] obtained from a WSN deployed in the Elora George Conservation Area, Ontario, Canada, and also out of some ideas gathered from [27]. The station deployed gathers 19 different kinds of data, from which we have selected 6

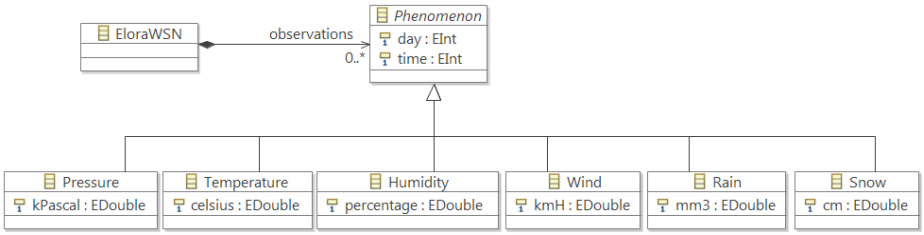

Fig. 1. Metamodel for observation phenomena in the WSN in Elora.

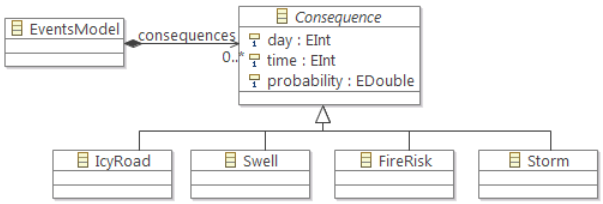

Fig. 2. Metamodel for defining consequences from observed phenomena.

of them. Thus, as we can see in the metamodel, the *EloraWSN* is composed of *observations*, of type *Phenomenon*. Each of them registers the *day* and *time*. The phenomena are specialized into different types: *Pressure*, *Temperature*, *Humidity*, *Wind*, *Rain* and *Snow*. Each of these specializations has an attribute that stores the value gathered from the environment. WSNs are normally a clear example of streaming models, since data is registered as time moves forward. Consequently, at any point in time, we cannot access the data that is to come in some future instant.

As explained in [27], to derive additional knowledge from semantically annotated sensor data, it is beneficial to use a rule-based reasoning engine. In this way, when a group of sensor nodes provides information regarding for instance temperature and precipitation, then, using such rules, we can specify possible road conditions. One of these rules could be the following: if the temperature is less than 1 degree Celsius and it is raining, then the roads are potentially icy. Also, the probability of the risk for having a fire or a storm could be obtained from the information about humidity, wind and temperature. To be able to express this kind of information in a model-based manner, we have created the metamodel shown in Fig. 2. It contains four different *Consequences* that are to be predicted according to the observed phenomena—there could be many more. Each consequence contains the *day* and *time* when it is predicted, as well as the *probability* that such consequence actually occurs.

In this context, we aim at reasoning over this kind of models not only considering the sliding window, i.e., the current information gathered from sensors that we have, but also taking into consideration that the information within such window can be potentially large. Thus, we focus on obtaining approximate predictions using AMTs.

3 Leveraging Sampling for AMTs

In statistics, a sample is a subset of individuals from within a statistical population that is known to be representative of the population as a whole. The process of using samples for addressing a specific problem is called *Sampling*. There exist several different sampling strategies. Which one to utilize depends on the context and the input data. For instance, in *Random Sampling*, each individual in the population is given the same probability to be in the sample. *Systematic Sampling*, in turn, involves a random start and then proceeds with the selection of every k^{th} individual from then onwards. There are other techniques to be used when the population embraces a number of distinct categories, such as *Stratified Sampling* or *Cluster Sampling*.

A common issue when selecting the sample of a population is to decide its size so that it can be representative. This is influenced by a number of factors, including the purpose of the study, population size, the risk of selecting a "bad" sample, and the tolerable sampling error. In [19], several strategies for determining the sample size are presented when the data in the population follow a normal distribution.

With the sampling concept in mind, our proposal aims to redefine the common operators that current model transformation languages use to manage and operate with collections, such as *allInstances()*, *collect()*, *select()*, ..., i.e., the collections operators used by OCL. When transforming very large models, these operations become very expensive, performance-wise, because they have to traverse the whole model and deal with a large number of elements. If we reduce this number of elements, the transformation will be faster.

For this reason we introduce a set of new collection operators, each one corresponding to an OCL collection operator. The new ones are suffixed by "*Approx*" and incorporate additional arguments: one indicating the confidence level (CL) and another one indicating the relative error (RE). For example, *Temperature.allInstancesApprox(95, 10)* returns a set of instances of type *Temperature* whose size is determined by the formula proposed in [19] according to the CL and RE specified.

Fig. 3 illustrates this idea. In the upper part of the figure, there are ten elements of a certain type. Realizing the *Type.allInstances()* operation would return all the elements. However, with our new operator *Type.allInstancesApprox(CL, RE)* we obtain only a subset of them. The lower part of the figure shows ten elements of a certain type that are referenced by the first element in the upper part. A normal *collect(Condition)* operation, where all the elements satisfy the condition, would select the ten elements (along with

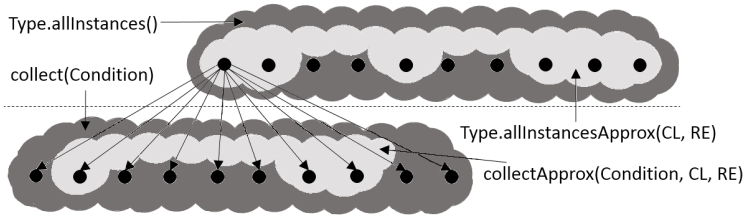

Fig. 3. Idea for Approximate Operators

their relationship to the object in the upper part). This means that all vertices and arcs are selected. We apply now the same concept as before with the *collectApprox(Condition, CL, RE)* operation, so that only a subset of the graph is actually retrieved.

Similarly, when we have a model transformation rule (think for instance of ATL [21]), declarative rules can also have now these arguments. In fact, matched rules in ATL apply an implicit *allInstances()* operation in the matching part, since they obtain all elements that satisfy this part (in the simplest case, where the matching part specifies only one type with no filtering, i.e., all the instances of such type are transformed). Finally, we also consider our approximate operators for the imperative part of transformation languages. Indeed, when a loop like *for (p in e.observations)* is used (consider that *e* is an element of type EloraWSN), then all phenomena are traversed. For this reason, applying an approximate operator would avoid the complete traversal.

The values to be chosen in the new arguments of these operations depend on how accurate and fast we want our transformation to be. There is effectively a tradeoff between these two metrics. Thus, the smaller the CL and the higher the RE, the smaller the size of the sample population is and the faster the transformation computes, being of course less accurate, and vice versa.

4 Implementation and Evaluation

We now describe how we have implemented and evaluated our case study.

4.1 Description and Setting

For our experiment, we use data gathered from the WSN in Elora during 2013 [2]. The data we have obtained from the website consists of 8760 points in time (365 days $\times$ 24 measurements per day). For each of these points, there is data gathered for each type. We have increased the quantity of data, by extrapolating the original, in 2^6 points. Thus, we have 6 different types of data (cf. Fig. 1), and for each data we have 560768 different measurements through time, what results in 3364608 objects in our input model, whose file has a size of 306 MB.

Out of the four possible consequences that may happen in the output model (Fig. 2), we want to focus now on calculating the risk of having a fire. Noble et al. [25] present a formula for calculating it. More specifically, they describe the *MacArthur FFDI* index, which is a weather-based index derived empirically in south-eastern Australia. It indicates the probability of a fire starting, its rate of spread, intensity, and difficulty of suppression. This formula uses data from temperature, humidity, wind and rain. We will not go into what the values returned mean, since it is out of the scope of the paper, but we want to compare the values calculated by the exact model transformation (EMT) and an approximate model transformation (AMT).

$$FFDI = 2 * (0.987 * log(D) - 0.45 + 0.0338 * T + 0.0234 * V - 0.0345 * H)^e$$

In the formula, D stands for *Drought*, T for *Temperature*, V for *Wind* and H for *Humidity*. As for drought, it is calculated taking into account the data regarding rain.

To be able to create a transformation where the huge amount of data we have in our model is utilized, we want to calculate the probability for having fire risk every month. This is, for obtaining the probability in a certain month, we will use the data gathered in the previous month (by calculating the average value).

For selecting the samples, we have decided to use Systematic Sampling. In this technique, the sample is selected according to a random starting point and a fixed, periodic interval. This interval, called the sampling interval, is calculated by dividing the population size by the desired sample size. The reason for using this technique in this case study, as well as the concrete method for calculating the sample size [19], is the way data related to meteorological conditions evolve over time. For instance, if we approach summer, it is very likely that the temperature after ten days is higher than today's, and the intermediate values would normally range between these two values. The data is, consequently, normally distributed.

4.2 Implementation and Results

In both scenarios, the EMT and the AMT, we have to calculate the index 12 times – since we are considering all the months of the year. In our EMT, we have around 46730 (560768 divided by 12) points in time for each month. In our AMT, we want to make two experiments: calculate the index with a confidence level of 95% and a relative error of 3%, and with a confidence level of 99% and a relative error of 3%. According to the formula described in [19], the AMT only has to deal, each month, with 1043 and 1764 points in time in these situations, respectively.

As a proof-of-concept of our approach, we have implemented our experiment in Java/EMF. Thus, after obtaining the data from the Elora Station [2], we have converted it into a model conforming to the metamodel shown in Fig. 1 and generated the API for the metamodel in Java. Then, once we have loaded the model into memory, we have used the same implementation for obtaining the FFDI indexes for each month, where in the EMT we traverse the whole population of phenomena and in the AMT we use only the data returned by the systematic sampling, as explained above. Regarding the output metamodel, we create an element of type *FireRisk* for each month, where *day* and *time* indicate the month (first day of the year and time of the day of each month), and *probability* stores the computed FFDI index.

The results displayed in Table 1 show the differences between the three executions. As it can be read, the difference between the FFDI indexes calculated by the exact and the approximate model transformations are very similar. Furthermore, the gain in performance is huge: the first AMT is 49 times faster than the EMT, while the second AMT is 23 times faster. We have analyzed the results and calculated the relative error, obtaining a value of 0.06287% in the first AMT, what makes sense since we have used a confidence level of 95%. The second, and more accurate, AMT produces only an error of 0.0285%, at the expense of spending more time in the execution.

5 Related Work

To the best of our knowledge, this paper is the first work which deals with the development of AMTs for large models. In [3, 24], a model transformation intent catalog is

Table 1. Results from Exact MT (EMT) and Approximate MT (AMT).

		95% CL and 3% RE		99% CL and 3% RE	
Month	**FFDI in EMT**	**FFDI in AMT**	**Error**	**FFDI in AMT**	**Error**
January	0.36845	0.36778	0.00181	0.36445	0.01097
February	0.37482	0.40547	0.07559	0.36478	0.02751
March	0.36216	0.37434	0.03252	0.35931	0.00795
April	0.40994	0.43950	0.06727	0.38833	0.05565
May	0.59438	0.59420	0.00031	0.59081	0.00604
June	0.69891	0.76448	0.08577	0.69739	0.00217
July	0.73598	0.78531	0.06281	0.71403	0.03073
August	0.80324	0.83444	0.03739	0.80616	0.00362
September	0.76696	0.71483	0.07939	0.83423	0.08063
October	0.80842	0.73761	0.09599	0.79740	0.01381
November	0.77053	0.79436	0.02999	0.75324	0.02296
December	0.48258	0.49907	0.03304	0.47241	0.02154
Exec Time	0.25651	0.005154	–	0.011153	–

introduced which contains different transformation intents, where one of them is the *approximation* intent. According to the definition given in [24], a transformation "m_1 approximates another transformation m_2 if m_1 is equivalent to m_2 up to a certain error margin". As an example the Fast Fourier Transformation is given which is an approximation of the Fourier Transformation. This definition is in accordance with our idea of AMT. As future work, studying which other kinds of transformation intents may benefit from AMTs seems promising.

A related research area is concerned with model transformation by-example [23, 31, 33], which may be interpreted as an approximation of output models for new input models based on previously seen input/output model pairs. However, here the idea is not trading accuracy for response time, but reducing the development efforts of model transformations.

Concerning the reduction of the number of input elements to be read from input models to produce output models, two transformation strategies have been discussed in the past. First, if an output model already exists from a previous transformation run for a given input model, only the changes in the input model are propagated to the output model. Second, if only a part of the output model is needed by a consumer, only this part is produced while other elements are produced just-in-time. For the former scenario, *incremental* transformations [7, 20, 22, 26] have been introduced, while for the latter *lazy* transformations [29] have been proposed. In this paper, we proposed an orthogonal approach which does not rely on previous transformation runs and is able to produce an approximate answer independent of its consumption. However, combining AMTs with other query optimization techniques such as incremental execution, e.g., to store samples, seems a next logical step.

The approximation of computations has a long tradition in the database area in order to deal with very large data sets [10, 13]. Thus, several approximation techniques have emerged in the last decades. One technique in this respect is online query processing [17], which provides intermediate results already before the exact result is produced. Other techniques such as histograms, samples, and wavelets, are based on precomputed data synopses [13], i.e., synopses are constructed and stored for the complete data set prior to query time and used at query time to answer the queries. Most related to AMTs

as presented in this paper are approximate queries based on sampling. For approximate queries the user specifies in addition to the query a certain error bound or time constraint for the query [1, 18]. Based on this meta-information, the database selects a sampling strategy or a pre-cached sample if available to answer the query.

To sum up, several approximation techniques are available for databases, but their adoption to models is challenging because of the different meta-languages (graphs vs. relations) and query languages (OCL vs. SQL) employed. Furthermore, approximate queries are still intensively studied for databases to make them useable in different scenarios as well as to support a wide range of different query types. Currently, different techniques support different query types, but supporting approximate queries for general and flexible queries is still limited [1]. In order to deal with current scalability challenges in MDE, it seems promising to further explore which kinds of approximation techniques fit best for certain MDE scenarios.

6 Conclusions and Future Work

In this paper we have introduced *Approximate Model Transformations*, which aim at producing target models that are accurate enough to provide meaningful and useful results in an efficient way, but without having to be fully correct. We have explained its basic notions and applied it successfully in a case study. AMTs provide a mechanism for being able to balance performance and accuracy in the realm of model transformations design and execution.

Of course, this paper presents an exploratory study which requires deeper investigations at all levels. In particular, we were interested in exploring the possibility of defining AMTs, and our initial results show that there are enough reasons to keep working on them. The work presented here does not pretend to be conclusive, or comprehensive, but to open the path for the model transformation community to start working on it.

The ideas presented in the paper may also be challenged, as potential threats may impact the internal and external validity [12] of the results showed here. For example, a potential threat to external validity is that the case study that has been utilized, even if it is a real-world example, does not cover all cases. For instance, our input data follows a normal distribution, so the function we have used for calculating the sample size [19], as well as the sampling strategy used—systematic sampling, are optimal for this case study. We need to study the situations in which the data come in different forms. As for internal validity, the main threat can be due to the way our approach has been implemented so far. We have used Java/EMF in our implementation. In fact, in this paper we have *simulated* the behavior of the approximate operators in Java/EMF and therefore the performance gains we have obtained may not be comparable as if the operators had been implemented in a model transformation language and the data had been distributed on different computing nodes.

There are several open issues that we plan to address next. In the first place, we would like to provide formal and precise specifications for our approximate operators, and integrate them in the ATL language. This would consist of extending the syntax of ATL so that these new operators are available, and also extending the ATL execution engine by implementing these operators. Secondly, we would like to add a new argu-

ment to the approximate operators, namely the time we want the operator to execute (such an approach is presented in [1] in the context of databases). The idea is to choose between this new operator or the two we have defined in this work (confidence interval and relative error) when calling a collection operator. Also, we need to study and develop methods for the appropriate design of AMTs. Although this will normally require a deep knowledge on the domain and the particular transformation scenarios, there is already a fair amount of work about the design of approximate and randomized algorithms [34] that could be applicable in this context. Last, but not least, in this work we have dealt with the approximation of values, i.e., numerical values stored in elements' attributes. However, we would also like to investigate how references can be approximated. For instance, if one element is referencing one million of different elements, how should we decide which ones to choose for the sample, and according to which criteria?

Acknowledgements. This work is partially funded by the European Commission under the ICT Policy Support Programme (grant no. 317859), by Research Project TIN2011-23795 and by Universidad de Málaga (Campus de Excelencía Internacional Andalucía Tech).

References

1. Agarwal, S., Mozafari, B., Panda, A., Milner, H., Madden, S., Stoica, I.: BlinkDB: Queries with Bounded Errors and Bounded Response Times on Very Large Data. In: Proc. of EuroSys. pp. 29–42 (2013)
2. Agricultural_Forest_Metereology_Group: Weather Records for the Elora Research Station, Elora, Ontario, Canada: Metereological data 2013 (2014), http://http://dataverse.scholarsportal.info/dvn/dv/ugardr
3. Amrani, M., Dingel, J., Lambers, L., Lúcio, L., Salay, R., Selim, G., Syriani, E., Wimmer, M.: Towards a Model Transformation Intent Catalog. In: Proc. of AMT. pp. 3–8 (2012)
4. Amrani, M., Lúcio, L., Selim, G., Combemale, B., Dingel, J., Vangheluwe, H., Traon, Y.L., Cordy, J.R.: A Tridimensional Approach for Studying the Formal Verification of Model Transformations. In: Proc. of VOLT. pp. 921–928 (2012)
5. van Amstel, M., Bosems, S., Kurtev, I., Pires, L.F.: Performance in Model Transformations: Experiments with ATL and QVT. In: Proc. of ICMT. pp. 198–212 (2011)
6. Azanza, M., Batory, D., Díaz, O., Trujillo, S.: Domain-specific Composition of Model Deltas. In: Proc. of ICMT. pp. 16–30 (2010)
7. Bergmann, G., Horváth, Á., Ráth, I., Varró, D., Balogh, A., Balogh, Z., Ökrös, A.: Incremental Evaluation of Model Queries over EMF Models. In: Proc. of MoDELS. pp. 76–90 (2010)
8. Burgueño, L., Troya, J., Wimmer, M., Vallecillo, A.: On the Concurrent Execution of Model Transformations with Linda. In: Proc. of BigMDE (2013)
9. Combemale, B., Thirioux, X., Baudry, B.: Formally Defining and Iterating Infinite Models. In: Proc. of MoDELS. pp. 119–133 (2012)
10. Cormode, G., Garofalakis, M.N., Haas, P.J., Jermaine, C.: Synopses for Massive Data: Samples, Histograms, Wavelets, Sketches. Foundations and Trends in Databases 4(1-3), 1–294 (2012)

11. Cuadrado, J.S., de Lara, J.: Streaming Model Transformations: Scenarios, Challenges and Initial Solutions. In: Proc. of ICMT. pp. 1–16 (2013)
12. Easterbrook, S., Singer, J., Storey, M.A., Damian, D.: Selecting Empirical Methods for Software Engineering Research. In: Guide to Advanced Empirical Software Engineering, pp. 285–311. Springer (2008)
13. Garofalakis, M.N., Gibbons, P.B.: Approximate Query Processing: Taming the TeraBytes. In: Proc. of VLDB (2001)
14. Gogolla, M., Vallecillo, A.: *Tract*able Model Transformation Testing. In: Proc. of ECMFA. pp. 221–235 (2011)
15. Guerra, E.: Specification-Driven Test Generation for Model Transformations. In: Proc. of ICMT. pp. 40–55 (2012)
16. Guerra, E., de Lara, J., Wimmer, M., Kappel, G., Kusel, A., Retschitzegger, W., Schönböck, J., Schwinger, W.: Automated verification of model transformations based on visual contracts. Autom. Softw. Eng. 20(1), 5–46 (2013)
17. Hellerstein, J.M., Haas, P.J., Wang, H.J.: Online Aggregation. In: Proc. of SIGMOD. pp. 171–182 (1997)
18. Hu, Y., Sundara, S., Srinivasan, J.: Supporting Time-Constrained SQL Queries in Oracle. In: Proc. of VLDB. pp. 1207–1218 (2007)
19. Israel, G.D.: Determining Sample Size, University of Florida, Institute of Food and Agriculture Sciences (1992)
20. Johann, S., Egyed, A.: Instant and Incremental Transformation of Models. In: Proc. of ASE. pp. 362–365 (2004)
21. Jouault, F., Allilaire, F., Bézivin, J., Kurtev, I.: ATL: A model transformation tool. Science of Computer Programming 72(1-2), 31–39 (2008)
22. Jouault, F., Tisi, M.: Towards Incremental Execution of ATL Transformations. In: Proc. of ICMT. pp. 123–137 (2010)
23. Kessentini, M., Sahraoui, H.A., Boukadoum, M., Benomar, O.: Search-based model transformation by example. SoSyM 11(2), 209–226 (2012)
24. Lúcio, L., Amrani, M., Dingel, J., Lambers, L., Salay, R., Selim, G., Syriani, E., Wimmer, M.: Model Transformation Intents and Their Properties. SoSyM pp. 1–35 (2014)
25. Noble, I.R., Gill, A.M., Bary, G.A.V.: Mcarthur's fire-danger meters expressed as equations. Australian Journal of Ecology 5(2), 201–203 (1980)
26. Razavi, A., Kontogiannis, K.: Partial evaluation of model transformations. In: Proc. of ICSE. pp. 562–572 (2012)
27. Sheth, A., Henson, C., Sahoo, S.S.: Semantic Sensor Web. IEEE Internet Computing 12(4), 78–83 (2008)
28. Tisi, M., Perez, S.M., Choura, H.: Parallel Execution of ATL Transformation Rules. In: Proc. of MoDELS 2013. pp. 656–672 (2013)
29. Tisi, M., Perez, S.M., Jouault, F., Cabot, J.: Lazy execution of model-to-model transformations. In: Proc. of MoDELS. pp. 32–46 (2011)
30. Vallecillo, A., Gogolla, M., Burgueño, L., Wimmer, M., Hamann, L.: Formal Specification and Testing of Model Transformations. In: Proc. of SFM. pp. 399–437 (2012)
31. Varró, D.: Model Transformation by Example. In: Proc. of MoDELS. pp. 410–424 (2006)
32. Wimmer, M., Burgueño, L.: Testing M2T/T2M Transformations. In: Proc. of MoDELS. pp. 203–219 (2013)
33. Wimmer, M., Strommer, M., Kargl, H., Kramler, G.: Towards Model Transformation Generation By-Example. In: Proc. of HICSS (2007)
34. Zhu, Z.A., Misailovic, S., Kelner, J.A., Rinard, M.C.: Randomized accuracy-aware program transformations for efficient approximate computations. In: Proc. of POPL. pp. 441–454 (2012)