Towards Statistical Comparison and Analysis of Models

Önder Babur[1], Loek Cleophas[1,2], Tom Verhoeff[1] and Mark van den Brand[1]
[1]*Eindhoven University of Technology, 5600 MB Eindhoven, The Netherlands*
[2]*Stellenbosch University, ZA-7602 Matieland, South Africa*

Keywords: Model-Driven Engineering, Model Comparison, Statistical Analysis, R, Vector Space Model, Clustering.

Abstract: Model comparison is an important challenge in model-driven engineering, with many application areas such as model versioning and domain model recovery. There are numerous techniques that address this challenge in the literature, ranging from graph-based to linguistic ones. Most of these involve pairwise comparison, which might work, e.g. for model versioning with a small number of models to consider. However, they mostly ignore the case where there is a large number of models to compare, such as in common domain model/metamodel recovery from multiple models. In this paper we present a generic approach for model comparison and analysis as an exploratory first step for model recovery. We propose representing models in vector space model, and applying clustering techniques to compare and analyse a large set of models. We demonstrate our approach on a synthetic dataset of models generated via genetic algorithms.

1 INTRODUCTION

Models are considered as central parts in Model-Driven Engineering (MDE), potentially orchestrating the whole software development process. As MDE gains popularity and widespread use, the complexity and variety of models increase. To cope with this problem, many approaches have been proposed: model comparison, matching, merging, etc. These approaches find many application areas in MDE, e.g. merging different model versions or detecting model clones. While the comparison techniques range from graph-based to linguistic ones, most involve pairwise or three-way (for model versioning from a common ancestor) comparison.

Domain model recovery, on the other hand, tackles the problem of extracting the common domain model out of various individual models; obtained either directly or indirectly by reverse engineering. Two examples of this can be drawn directly from our ongoing project for a flexible multiphysics engineering simulation framework: constructing (1) a standardized metadata schema that can support a number of input formats/schemas, and (2) a common metamodel/ontology to orchestrate the interoperability of a heterogeneous set of tools. The study in (Babur et al., 2015a) indicates the overwhelming number of tools in the domain, which makes it really difficult to extend manual model extraction efforts such as in (Babur et al., 2015b) to cover the whole domain.

As formulated above, domain model recovery qualifies as another potential application area for model comparison, but with a multiplicity of models to consider. This dimension has been diagnosed in (Klint et al., 2013) as an interesting aspect to explore. (Rubin and Chechik, 2013) further discusses the inadequacy of pairwise comparison techniques for multiple models and proposes an N-way model merging algorithm. To the best knowledge of the authors the statistical dimension of the generic model comparison problem has been largely overlooked.

In this paper, we present an initial attempt at model comparison and analysis for large datasets using Information Retrieval (IR) techniques and statistical analysis, as an exploratory first step in model recovery. In IR, a vector space model (VSM) is used to represent text documents, with vector elements corresponding to word occurrence (incidence) or frequency. We borrow this concept to represent models as vectors of the bigram combinations of related model elements. This model, in turn, is supplied with inverse document frequency and type-based weighting scheme to compute distances of models in the vector space. We then use the R statistical software (Maechler et al., 2013) to cluster, analyse and visualise the dataset. As an initial feasibility test of our approach, we use synthetically generated model pop-

The research leading to these results has been funded by EU programme FP7-NMP-2013-SMALL-7 under grant agreement number 604279 (MMP).

Babur, Ö., Cleophas, L., Verhoeff, T. and Brand, M.
Towards Statistical Comparison and Analysis of Models.
DOI: 10.5220/0005799103610367
In *Proceedings of the 4th International Conference on Model-Driven Engineering and Software Development (MODELSWARD 2016)*, pages 361-367
ISBN: 978-989-758-168-7

ulations obtained using the metamodel-driven model mutation framework in (van den Brand et al., 2011).

Positioning Model Comparison as a First Step for Model Recovery. Model comparison (Stephan and Cordy, 2013) is found in the literature as a common operation in a wide range of application areas, such as model versioning (Altmanninger et al., 2009), model clone detection (Deissenboeck et al., 2010), model merging (Brunet et al., 2006), model matching (Kolovos et al., 2009) and model recovery (Klint et al., 2013). It is typically defined as a binary operator (sometimes ternary in model versioning), mapping two model elements to a similarity measure. It can be categorized according to the output: boolean (exact matchers), similarity category (e.g. near-miss clones for model clone detection), or similarity degree (e.g. a real number in the range [0.0, 1.0]).

Model recovery can be considered as an umbrella term that encapsulates different approaches named domain model recovery (Klint et al., 2013), metamodel recovery (Javed et al., 2008), automated domain modeling (Reinhartz-Berger, 2010), etc. We can regard model recovery as reverse engineering a common model out of several entities, either directly from variant models, or from other sources such as design documents and source code - adding another layer of reverse engineering. For this paper, we choose to ignore this second layer; instead we lift the problem completely to the model level. So the starting point for our consideration is a set of homogeneous models, representing a common domain. A concrete example would be the class diagrams of similar software packages. While these models share similarities of the domain, no individual model is an instantiation of a metamodel or a configuration of a feature model. We would rather like to use model comparison in the sense of analysing the similarities among models, trying to find clusters and obtaining an overview of the whole dataset. The analysis information can be regarded as a first exploratory step in domain model recovery from multiple models.

Information Retrieval and Clustering. Information Retrieval (Manning et al., 2008) has a long history of developments in dealing with effectively indexing, analyzing and searching various forms of content including natural language text documents. As a first step for document retrieval in general, documents are collected and indexed via some unit of representation. Index construction can be implemented using models ranging from boolean indices to complex neural networks. One such model is the vector space model (VSM) with the following major steps:

- A vector representation of occurrence of the vocabulary in a document (binary), named *term incidence*;
- Optionally *zones* (e.g. 'author' or 'title' zones separate from the text bodies);
- Optionally weighting schemes to be used as multipliers such as:
  - *inverse document frequency* (see Section 2) to increase the discriminative effect of rare words;
  - zone weights, e.g. higher for important zones.

Once the VSM is constructed, the similarity of documents can be defined as the distance between these vectors. There exist several distance/similarity measures, such as Euclidian, Cosine or Manhattan. VSM with a selected distance measure is the prerequisite for identifing common groups of documents in the vector space. Among many different clustering methods (Jain and Dubes, 1988), K-means is a simple but effective one. It aims to identify cluster centers and minimises the residual sum of (squares of) distances of the points assigned in each cluster. Hierarchical clustering techniques assume an unknown number of clusters and rather build a nested tree structure (*dendrogram*) of the data points, representing proximities and potential clusters.

Objectives. The purpose of this study is to answer the following questions:

- **RQ1.** How can we represent models for N-way comparison and analysis?
- **RQ2.** How can we analyse and compare a large set of models?

2 METHOD FOR COMPARING A LARGE SET OF MODELS

In this section, we elaborate our approach on a small example. We describe the main steps as:

1. Obtaining a set of homogeneous models (of same type, e.g. UML class diagram) to be analyzed.
2. Generating the bigram vocabulary from the input models and the bigram types from the metamodel (that is, the generic metamodel e.g. UML metamodel, rather than lower level domain-specific metamodels); types being similar to zones in IR.
3. Calculating the term incidence matrix with an idf and type-based weighting scheme.
4. Analyse the dataset via simple K-means and/or hierarchical clustering with Manhattan distance.

A Small Exemplary Dataset. First we define a simple metamodel with basic building blocks, based on the Ecore kernel at (Budinsky, 2004), to base our

Table 1: Term incidence matrix of the input models. Note that some of the terms have been abbreviated due to space constraints, e.g. rootPackage-Bank as r-Bank and Bank-managers as Bank-mrs.

Model	r-Bank	r-Person	r-Account	...	Bank-mrs	mrs-Manager	Manager-Person	...
M1	1	1	0	...	1	1	1	...
M2	1	1	1	...	1	1	1	...
M3	1	1	0	...	1	1	1	...
M4	1	0	0	...	1	1	0	...

experiments on; since real metamodels such as Ecore are too large and complex for the purpose of this study. Figure 1 demonstrates the metamodel to drive the examples.

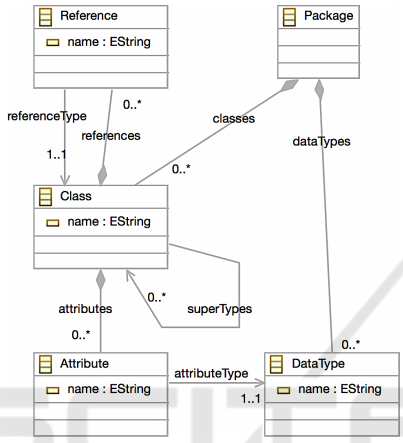
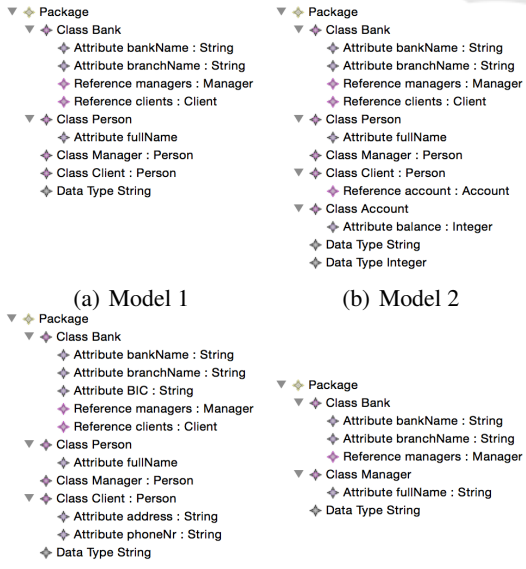


Figure 1: Simple metamodel for experiments.

Based on this metamodel, we construct four models with a few differences such as addition of new classes or removal of attributes, depicted in Figure 2.



(a) Model 1 (b) Model 2

(c) Model 3 (d) Model 4

Figure 2: A small set of models.

Generating the Bigram Vocabulary. From the input models and metamodel, we construct a typed bigram vocabulary. The type information comes from the metamodel: From Figure 1, we get the set {**classes, attributes, references, supertypes, referenceType, attributeType dataTypes**}. Next, we traverse all the models to extract the union of bigrams as our vocabulary: the examples in Figure 2 would yield **Model 1** = { **rootPackage-Bank, rootPackage-Person, ..., Bank-bankName, ..., bankName-String, ...** } The choice of bigrams vs. unigrams captures relational information between model elements.

Calculating Term Incidence Matrix. The bigram sets for each model allow us construct the term incidence matrix as in Table 1. We propose to apply a weighting scheme on the term incidence matrix, which includes two multipliers: an inverse document frequency (idf) and a type (zone) weight. The idf of a term t is used to assign greater weight to rare terms across models, and is defined as:

$$idf(t) = log_{10}\left(1 + \frac{\#\text{ total models}}{\#\text{ models with the term t}}\right)$$

Furthermore, a type weight is given to the bigrams representing their semantic importance. We claim, for instance, that classes are semantically more important than attributes, thus deserve a greater weight. We have used this experimental scheme for this paper:

$$zoneWeight(t,w) : \{\text{classes} \rightarrow 1.0, \text{attributes} \rightarrow 0.5, \text{dataTypes} \rightarrow 0.5, \text{references} \rightarrow 0.5, \text{supertypes} \rightarrow 0.2, \text{referenceType} \rightarrow 0.2, \text{attributeType} \rightarrow 0.2\}$$

The resulting matrix where term incidences are multiplied by idf and weights is given in Table 2.

Table 2: Idf and type weighted matrix.

Model	r-Bank	r-Person	r-Account	...
M1	0,30	0,37	0	...
M2	0,30	0,37	0,70	...
M3	0,30	0,37	0	...
M4	0,30	0	0	...

K-means Clustering. As the next step of our approach, we reduce the model similarity problem into a distance measurement of the corresponding vector

representations of models. Among various measures, Manhattan distance seems to be suitable since it corresponds to the *edit distance* of two models. p and q being two vectors of n dimensions, Manhattan distance is defined as:

$$manhattan(p,q) = \sum_{i=1}^{n} |p_i - q_i|.$$

While clustering makes little sense for our small example dataset we can still compute the center of just one cluster and the distances of models to the center. The center can be considered as a *rough average* of that cluster. Similarly, the model closest to the center can be considered as a *representative* of that cluster. The models' distances to center are given in Figure 3.

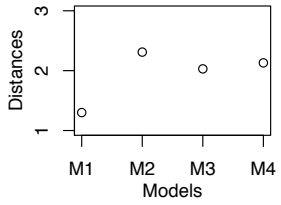


Figure 3: Distances of models to the cluster center.

3 A DATASET FOR MODEL COMPARISON

Model Mutation Framework. We use the model mutation framework in (van den Brand et al., 2011) to synthetically generate populations of models to be used for comparison. The framework is capable of generating mutations on a model based on its metamodel, i.e. the mutated instances will still be metamodel-compliant. It can be configured to generate random distribution of different types of mutations such as addition, removal or modification of model elements. For more information on the features and operation, please refer to (van den Brand et al., 2011).

Synthetic Generation of Model Populations. Using the mutation framework, we have generated a simple setting for creating model populations based on the metamodel in Figure 1 and a slightly modified version of the bank model example in Figure 2. The goal is to emulate a model population evolved from a common model (e.g. a hypothetical domain model). The genetic algorithm starts with the initial generation consisting of only M_0, i.e. the root model. At every iteration, the set of models at that level (called a *generation*) is mutated with one atomic addition (with 70% probability) or removal (with 30% probability) operator. The number of offsprings generated at each iteration is set by a branching factor b, and the total number of iterations is set by a generation count n. For our experiments, we did not consider any fitness function (i.e. every generated model survives) or crossover operations. One further simplification is that modification of model elements (e.g. renaming) is also not included as a mutation operator.

The algorithm basically creates tree structures representing model generations with atomic mutations at each branch. Following this setting, we have generated one dataset to test our approach: with $b = 2$ and $n = 5$, where we keep the older generations and thus have the whole population tree.

4 EXPERIMENTS & DISCUSSION

Experiment on the Whole Tree. The first analysis on the dataset was determining the number of clusters, k. A simple method for this is running the clustering for a range of 1 to a suitably large N and visualising the total sum of squares within clusters. Figure 5 shows these numbers for $N = 15$. A visual heuristic was used for locating *'elbows'* in the plot, potentially good candidates for k. Inspecting the figure, we picked $k = 7$ by its significant drop after $k = 6$, and the relative saturation after $k > 7$.

We in turn cluster the dataset with Manhattan distance and $k = 7$. The tree-like structure of model population, with each node as one model, branch as a mutation, and depth as a generation, allows a good visualisation of the clustering algorithm. The clustering was done with the *Kmeans* function of *amap* package and the result is depicted in Figure 4. Each cluster was coded with a unique colour/pattern. The visualisation agrees with the initial intention; populations evolve in different directions and start to form clusters.

Experiment on the Leaves. We would also like to analyse the set of models, finding representative models and outliers among them. To emulate this scenario, we used only the youngest generation (i.e. the leaves of the tree) to cluster. First we clustered the set with just $k = 1$ to find a center and plot the distances of each model to the center, depicted in Figure 6. Two implications are that there are no representative models close enough to the center, and no visible outliers within the single cluster. This is expected as models have accumulated mutations over generations and form separate clusters (Figure 4).

This scenario is suitable for applying hierarchical clustering just on the leaves, and try to reconstruct a hierarchical similarity scheme resembling the one in Figure 4. Indeed, we obtained a dendrogram of the models using the *hclust* function in *stats* package with Manhattan distance, given in Figure 7. The nodes on

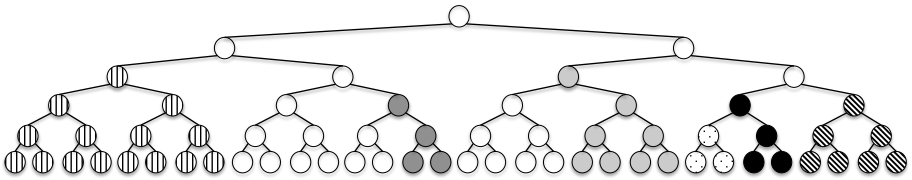


Figure 4: Population tree with 7 clusters.

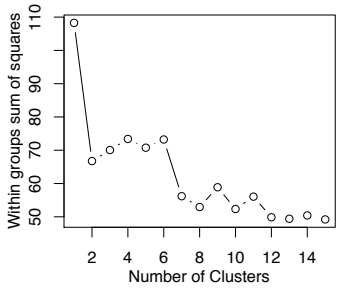


Figure 5: Total sum of squares within clusters vs # clusters.

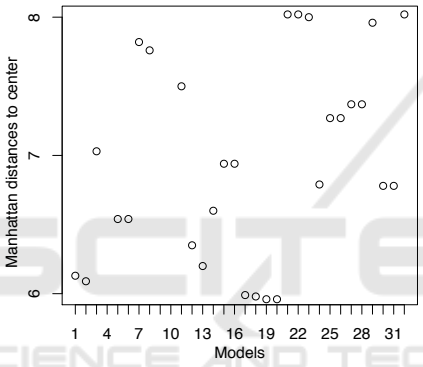


Figure 6: Distances of the leaf models to the center.

the dendrogram are labelled 1 to 32 and correspond to the consecutive leaves from left to right in Figure 4. There is an obvious resemblance between the two structures; the dendrogram can be interpreted as an approximate reconstruction of the population tree.

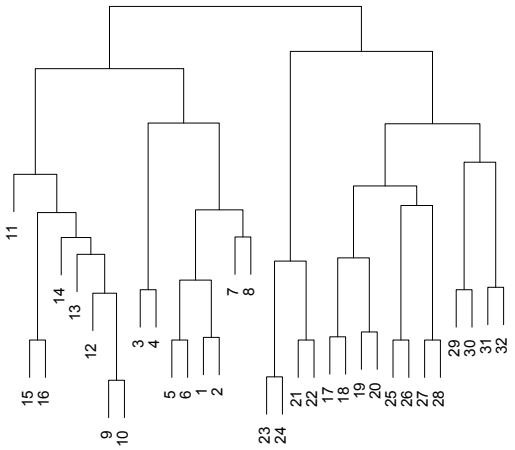


Figure 7: Hierarchical clustering - dendrogram.

An Initial Assessment of our Approach. Based on our experiments, a statistical perspective on the comparison and analysis of large datasets seems promising. While the dataset was synthetic and relatively small, we were able to obtain intuitive results. Using VSM allows a uniform representation of models for statistical analysis, while the accompanying idf and type-based weighting scheme yields a suitable scaling in the vector space (**RQ1**). Using a distance measure and clustering over VSM, many characteristics and relations among the models, such as representatives, clusters and outliers, can be analysed (**RQ2**).

An advantage of our approach is the scalability and tool support. The algorithm complexity range from linear (e.g. VSM construction) to polynomial (e.g. K-means as observed complexity and hierarchical clustering) with respect to the number and size of models. Moreover, R provides a plethora of efficient and flexible statistical libraries for analysis. Metamodel-based construction of the bigram vocabulary provides a good amount of reduction in vector space, improving over basic IR indexing.

Threats to Validity. While our initial experiments indicate the potential of our approach for large-scale model comparison and analysis, there are a number of threats to validity. The biggest one is that we have used a synthetic, simplified and homogeneous dataset. The variations in the synthetic dataset are simpler than in a real world scenario where larger, more complex and possibly heterogeneous models are reverse engineered from different tools. For instance, we currently handle neither Natural Language Processing (NLP) issues such as synonyms/renamings, nor semantical equivalence of model elements. Furthermore, while the general architecture of our approach seems plausible, we have not evaluated other options for individual steps; but chosen them intuitively as a first attempt.

5 RELATED WORK

Only a few model comparison techniques consider the multiplicity of input models without doing pairwise comparisons, such as N-way merging based on weighted set packing in (Rubin and Chechik, 2013). Feature model extraction (She et al., 2011) and con-

cept mining (Abebe and Tonella, 2010) use NLP to cluster features/concepts. (Ratiu et al., 2008) builds domain ontologies as the intersection of graphs of APIs, but does not focus on the statistical dimension of problem. Metamodel recovery (Javed et al., 2008) is another approach which assumes a once existing (but somehow lost) metamodel, and does not hold for our scenario. (Dijkman et al., 2011) applies a technique similar to ours, specifically for business process models using process footprints and thus lacks the genericness of our approach. Note that a thorough literature study beyond the technological space of MDE, for instance regarding data schema matching and ontology matching/alignment, is out of scope for this paper and is therefore omitted.

6 CONCLUSIONS AND FUTURE WORK

In this paper, we have presented a new perspective on the N-way comparison and analysis of models as a first step in model recovery. We have proposed a generic approach using the IR techniques VSM and tf-idf to uniformly represent multiple models, and apply statistical analysis with K-means and hierarchical clustering. Using a model mutation framework, we have synthetically generated a dataset to apply our method and demonstrate its potential uses. The results indicate that our approach is a promising first step for analysing large datasets, being generic and scalable/efficient using R.

As future work, the most important goal is to work with real datasets of possibly heterogeneous models, rather than the synthetic one. A real dataset (e.g. class diagrams of multiple domain tools) can be acquired through reverse engineering, which we have omitted for this paper. The NLP or semantic issues pose the next set of challenges to tackle. A careful assessment of different and more advanced options for model representation, distance measures, and clustering techniques needs to be done in order to increase the accuracy and efficiency of our approach. Although this is presented as an exploratory step, it can also be investigated how VSM and clustering information can be used for model merging and domain model recovery.

REFERENCES

Abebe, S. L. and Tonella, P. (2010). Natural language parsing of program element names for concept extraction. In *Program Comprehension (ICPC), 2010 IEEE 18th International Conference on*, pages 156–159. IEEE.

Altmanninger, K., Seidl, M., and Wimmer, M. (2009). A survey on model versioning approaches. *International Journal of Web Information Systems*, 5(3):271–304.

Babur, Ö., Smilauer, V., Verhoeff, T., and van den Brand, M. (2015a). Multiphysics and multiscale software frameworks: An annotated bibliography. Technical Report 15-01, Dept. of Mathematics and Computer Science, Technische Universiteit Eindhoven, Eindhoven.

Babur, Ö., Smilauer, V., Verhoeff, T., and van den Brand, M. (2015b). A survey of open source multiphysics frameworks in engineering. *Procedia Computer Science*, 51:1088–1097.

Brunet, G., Chechik, M., Easterbrook, S., Nejati, S., Niu, N., and Sabetzadeh, M. (2006). A manifesto for model merging. In *Proc. of the 2006 Int. Workshop on Global Integrated Model Management*, pages 5–12. ACM.

Budinsky, F. (2004). *Eclipse modeling framework: a developer's guide*. Addison-Wesley Professional.

Deissenboeck, F., Hummel, B., Juergens, E., Pfaehler, M., and Schaetz, B. (2010). Model clone detection in practice. In *Proc. of the 4th Int. Workshop on Software Clones*, pages 57–64. ACM.

Dijkman, R., Dumas, M., Van Dongen, B., Käärik, R., and Mendling, J. (2011). Similarity of business process models: Metrics and evaluation. *Inf. Systems*, 36(2):498–516.

Jain, A. K. and Dubes, R. C. (1988). *Algorithms for clustering data*. Prentice-Hall, Inc.

Javed, F., Mernik, M., Gray, J., and Bryant, B. R. (2008). Mars: A metamodel recovery system using grammar inference. *Inf. and Software Tech.*, 50(9):948–968.

Klint, P., Landman, D., and Vinju, J. (2013). Exploring the limits of domain model recovery. In *Software Maintenance (ICSM), 2013 29th IEEE International Conference on*, pages 120–129. IEEE.

Kolovos, D. S., Ruscio, D. D., Pierantonio, A., and Paige, R. F. (2009). Different models for model matching: An analysis of approaches to support model differencing. In *Comparison and Versioning of Software Models, 2009. ICSE Workshop on*, pages 1–6. IEEE.

Maechler, M., Rousseeuw, P., Struyf, A., Hubert, M., and Hornik, K. (2013). *cluster: Cluster Analysis Basics and Extensions*. R package version 1.14.4.

Manning, C. D., Raghavan, P., Schütze, H., et al. (2008). *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge.

Ratiu, D., Feilkas, M., and Jürjens, J. (2008). Extracting domain ontologies from domain specific apis. In *Software Maintenance and Reengineering, 2008. CSMR 2008. 12th European Conf. on*, pages 203–212. IEEE.

Reinhartz-Berger, I. (2010). Towards automatization of domain modeling. *Data & Knowledge Engineering*, 69(5):491–515.

Rubin, J. and Chechik, M. (2013). N-way model merging. In *Proc. of the 2013 9th Joint Meeting on Foundations of Software Engineering*, pages 301–311. ACM.

She, S., Lotufo, R., Berger, T., Wąsowski, A., and Czarnecki, K. (2011). Reverse engineering feature models. In *Software Engineering (ICSE), 2011 33rd International Conference on*, pages 461–470. IEEE.

Stephan, M. and Cordy, J. R. (2013). A survey of model comparison approaches and applications. In *Modelsward*, pages 265–277.

van den Brand, M., Hofkamp, A., Verhoeff, T., and Protić, Z. (2011). Assessing the quality of model-comparison tools: a method and a benchmark data set. In *Proc. of the 2nd Int. Workshop on Model Comparison in Practice*, pages 2–11. ACM.