InTra: A Pragmatic Approach of Using Rule-Based Model Transformation to Reduce Complexity of UML and SysML Models

Philippe Barbie, Martin Becker and Andreas Schäfer
Fraunhofer IESE, Fraunhofer-Platz 1, 67663, Kaiserslautern, Germany

Keywords: Model Transformation, Interaction-Based Transformation, Rule-Based-Modelling, Interaction-Patterns, Complexity, MBSE, SysML, UML.

Abstract: While UML and SysML are important concepts for digitally representing complex systems, we are still confronted with the problems of comprehensibility and maintainability. Even models with comparatively few elements can quickly become confusing and hard to read, due to their relationship density. This leads to potential errors, both for the initial modelling and the later model evolution and evaluation phase. Driven by the needs of a large-scale modelling project in the field of communication systems, we have researched approaches to cope with the inherent model complexity by following a modelling approach, that is based on interaction patterns. In this paper, we will give an introduction to the challenge of the consistently increasing complexity of system models in the field of model-based systems engineering (MBSE). We will motivate our need for rule-based modelling in order to handle a real industry use case and outline the problems of system modeling, which could be solved by using a new rule-based modelling approach. InTra (*Interaction-based Transformation*) is an approach to significantly reduce the complexity of a system model, by reducing the number of connectors through the use of interaction rules. By doing so, we were able to create an abstracted variant of the same system model, but with a highly reduced number of connectors used and thus an overall reduced model complexity.

1 INTRODUCTION

The complexity of modern systems and the associated modelling effort is steadily increasing (Antinyan, 2020) (Warrilow, 2020) (Baduel, 2018) (Stützel, 2021). In a survey with 127 participants in 14 companies conducted in 2001, 84% of the participants answered the question *'What motivates your company to introduce systems engineering?'* with the reason of constantly increasing product complexity (Stützel, 2021). It is inspiring to observe that both David Long and Baduel et al. mention the particular need for improvement in modelling methodology to cope with the growing complexity of future system models (Warrilow, 2020) (Baduel, 2018). Since the early 2000's, a variety of different transformation approaches have been developed. While many of those approaches have their focus on transforming system or software models into executable code, there are also some languages that concentrate on model-to-model transformation. Some of the more popular approaches are, e.g. PROGRES (RWTH Aachen, 2021), AGG (Taentzer, 2003), AToM^3 (McGill, 2021), GReAT (Balasubramanian, 2006) (ISIS, 2021), GROOVE (Twente, 2021), BOTL (Braun, 2003) and Fujaba (Fujaba Development Group, 2021), even though some of them are not developed any further (Kahani, 2019) (Schürr, 2021).

In this paper we will investigate whether it is possible to use a model transformation approach to significantly reduce the complexity of system models, by transforming them into a rule-based representation. We developed a specific rule-based model transformation approach called InTra (*Interaction-based Transformation*), based on the problems introduced by a real industrial communication system, with over 5,000 user roles and over 9,000 communication relationships. This approach has the goal to decrease a system's complexity, by reducing it to its basic structure and combining it with applicable relationship patterns, thus decreasing the number of connectors in the resulting model by an order of magnitude. Having a large industry-related model, based on realistic data, was especially useful when developing the approach and testing its feasibility. To the best of our knowledge, this is the first approach that uses model-to-model transformation techniques to reduce a model's

Barbie, P., Becker, M. and Schäfer, A.
InTra: A Pragmatic Approach of Using Rule-Based Model Transformation to Reduce Complexity of UML and SysML Models.
DOI: 10.5220/0011615600003402
In *Proceedings of the 11th International Conference on Model-Based Software and Systems Engineering (MODELSWARD 2023)*, pages 97-104
ISBN: 978-989-758-633-0; ISSN: 2184-4348

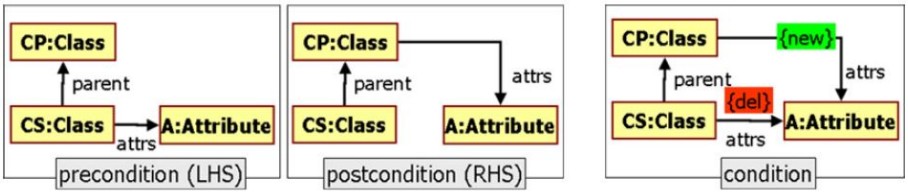

Figure 1: Notation of a transformation rule for graph models, according to (Varró, 2007).

information complexity. In this case, model transformation is not used, to transform the model into a different state of development, but to transform between different representations of the same model state.

The paper is structured as follows. Section 2 offers a general overview of the field of graph-based model transformation. Section 3 describes the industrial setting in which we developed and tested the InTra approach. Section 4 elaborates the problems, which led us into the development of a rule-based model transformation approach. Section 5 explains the InTra approach in general and why model transformation is a necessary part of it. Section 6 will present the basic functionality and application of InTra, based on a simple family tree example model. Section 7 explains InTra's own model transformation language, which is completely based on model elements itself. Section 8 describes the experiences and results that were achieved in the given industrial setting from Section 3. Section 9 reviews relevant existing work in the field, e.g. work that helped us to derive necessary requirements for the development of a model transformation approach. Finally Section 10 concludes the paper.

2 TRANSFORMATION OF GRAPH-MODELS

While models exist in many different forms, such as relational databases or text-based structures, in this paper the scope will be centered around the transformation of graph models. Graphs are powerful data structures that are able to represent interactions between system components in an easily comprehensible way. Essentially, any system that consists of components with relations may be naturally described by a graph in which the vertices represent the entities and the edges represent the relations (Ghamarian, 2011). A transformation rule for graph models usually consists of two parts: *Pattern-Matching* and *Rewriting* (Wieber, 2015).

Pattern Matching describes the structure of the pattern, which is to be searched in the model. At all places where a given subgraph is found that matches the pattern, a transformation can take place. This first section of a transformation rule is also called '*Left-hand-side (LHS)*' (Wieber, 2015) (Czarnecki, 2006).

Rewriting, the second part of a transformation rule describes the final state, which has to be realized on all found pattern matching subgraphs. This part of the transformation rule is also called '*Right-hand-side (RHS)*' (Wieber, 2015) (Czarnecki, 2006).

In the example shown in Figure 1, taken from (Varró, 2007), the LHS defines a class *CS*, which is a child of a class *CP*, which has an attribute relationship to attribute *A*. The RHS of the interaction rule shows as a result of an applied model transformation, a deleted relationship between class *CS* and attribute *A* and the creation of a new relationship between the child class *CP* and attribute *A*.

3 INDUSTRIAL CONTEXT

In a previous industry project, we had to model an architecture representing a communication system with over 5,000 user roles to capture individual requirements of those user roles and their communication. The basic procedure was to model each user-role in each scenario and map the communication between those user roles via information exchange relationships. Each of those information exchange relationships contains at least one piece of information. Since this is not comprehensibly representable without any system structure, the user roles were grouped according to their communication hierarchies. This included further refinements in subgroups, as well as the mapping in superiors and subordinate user roles. Nevertheless, there was an average of three information exchanges per user-role in each communication scenario. One of the technical requirements for the system, which was set by the customer, was to use the well established modelling tool *Enterprise Architect* by Sparx Systems. As source of information, more than 50 domain experts were interviewed and asked what communication relationships exist in their specific working environment. The information collected had to be modelled into the system context, attributed, managed and delivered back for review. Even though

the information was available inside the model, it was difficult and time-consuming to comprehend, verify, and maintain the resulting system. As a result, even subject matter experts had major problems in understanding the communication relationships they should be already familiar with. Furthermore it was an essential requirement, that the resulting model diagrams could also be well understood by those who are not familiar with MBSE and system architectures. Unfortunately, for reasons of confidentiality, the corresponding system model and its contents cannot be made available to the readership within this paper.

4 PROBLEM STATEMENT

It can be observed that the technological barriers are not posing the greatest problem in here, but rather those of human cognitive perception. Mapping the structure and interrelationships of complex systems as a digital model is a major challenge. To be able to comprehend, understand and maintain what is represented beyond that, is an even greater task. Of course, UML and SysML offer a variety of methods to decompose the system to reach lower complexity. On the other hand, even using the best decomposition, you eventually reach the point where you need to represent the relations between the system components in the model. Normally this is done using elements and connectors, but even with just a few elements, this may mean a large number of information flows on a single diagram. Yet, it is the understanding of this connector-representation between system elements, that is a major cognitive challenge for the human brain (Koning, 2002).

In the study *User preference of graph layout aesthetics: a UML study* (Purchase, 2000) on the better comprehensibility of UML diagrams, 93% of the participants stated that crossing connectors in the diagram are an enormous obstacle to comprehensibility. 91% testified that bends in the connectors were a major barrier. These were the two most frequently mentioned comprehension barriers in the study.

In order to solve the problems mentioned above and to counteract the challenge of increasing complexity in system models, we will elaborate our approach named InTra (*Interaction-based Transformation*), which significantly reduces the number of connectors in the model, but at the same time preserves their information content in a comprehensible way.

5 InTra APPROACH

The approach introduced in this article is designed to significantly reduce the complexity of a system model with the help of individual *interaction rules* that work on the basic system structure. In this way it is possible to reduce the number of connectors even during the initial modelling phase, thus keeping the resulting model complexity small, which in turn improves the readability and maintainability. The term *'rule-based'* describes the idea of recognizing repeating interaction patterns in the basic structure of the model, and the definition of rules for those patterns, which describe the interaction concept behind those relationships. By including these rules, it is no longer necessary to store the said relations by connectors in the model. The creation of a rule is useful, if an interaction between certain element types or constellations always runs in the same way. Although in this rule-based form it is a lot harder to evaluate the model with simulations or programs, it is fully completed and understandable in its rule-based abstracted form. This means, that all information of the relations to be captured is already mapped in the model, partly abstracted as interaction rules to be interpreted. By expressing this information as rules, the representation of the model is changed, but its information content remains the same. Avoiding the redundancy of information in this way also reduces the risk of errors in the further processing of the model, since changes only need to be made to one rule instead of all the connectors that a single rule represents. Furthermore, the usage of interaction rules makes the model easier to understand for domain experts. This is the case because domain-specific interactions can be represented very efficiently and comprehensibly as interaction rules. Nevertheless, an experienced system architect should be responsible for identifying the basic structure of the system, recognizing repetitive patterns in the relationships between model elements, defining and configuring interaction rules for those patterns, and finally linking them to the relevant parts of the system's basic structure.

In an optional second phase of *model transformation*, the defined interaction rules are able to transform the model into a rule-independent version, suitable for data usage through e.g. simulations, by using the automatic model transformation algorithm of InTra. Therefore linked interaction rules are parsed by the algorithm, to define active filters for the pattern matching of the target model. Pattern matching will then find valid interaction paths inside the model and apply the defined rewriting effects between all start and endpoints of those found paths. Even though the rule-based version of the model is fully expressive, we

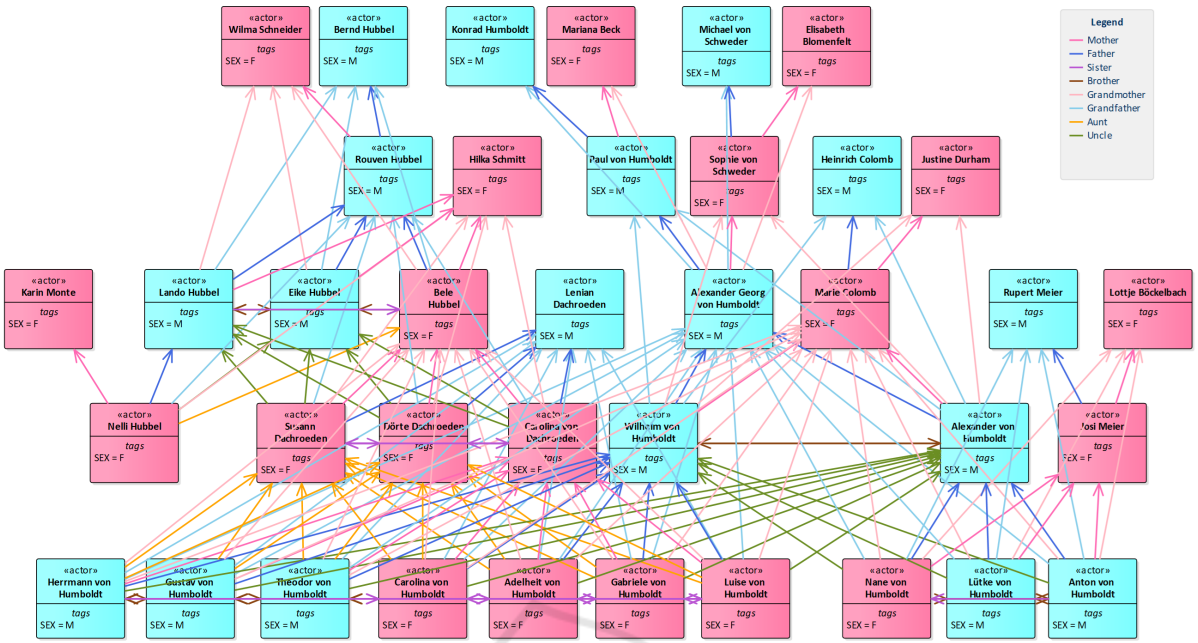

Figure 2: Extended family tree model with multiple incorporated relationship types.

need a way to (back-)convert the model to the traditional representation, as a backbone for our methodology. Furthermore, the transformation into independent relations can help modelers to understand the statement of a complex interaction rule and makes the model readable again for simulations and other external programs. For this reason we use model transformation techniques to be able to switch between rule-based and rule-independent representation of the model.

6 APPLICATION SHOWCASE

An illustrative example, although not productively usable, is a family tree as presented in Figure 2, which shall serve as an example to visualize the InTra approach. A family tree is a good way to explain the idea of InTra, because we already use the concepts of rule-based modelling subconsciously when thinking about family trees. The shown tree model consists of actors (*elements*), their gender (*attributes*), and the relationships between family members (*connectors*). As you can see, the model in Figure 2 shows several family relations in only one diagram. Normally, in a family tree we would expect exclusively `Child` connectors, and we would still be able to read other family relations from the model subconsciously. This cognitive concept is very similar to rule-based InTra approach. By using interaction rules to embed information into a model, that can be abstracted from existing information, we are able to reduce the model to its relevant basic system structure. In this case it is sufficient to only keep the `Child` relations, as we are used to from regular family trees, to be able to represent all other relationships with the help of rules.

For easier explanation we will concentrate on only a small part of the given family tree, which can be seen in Figure 3. Based on the here shown `Child` relations, which are part of the model's basic structure, we will build interaction rules that represent all other visible connectors. For example, each *male* or *female* source element of a `Child` relation is a *father* or a *mother* of the target element of that relation, respectively. To replace this kind of relation with an interaction-rule, such rule must be defined to search for incoming `Child` connectors for each possible element and follow that connector exactly one *hop* against its direction. Thereupon, the found parent element must be distinguished between male (`M`) or female (`W`), respecting the gender attribute (`SEX`), in order to then represent any `Father` or `Mother` relation in the model. Once these two rules are associated with the family tree model, all `Mother`, and `Father` connectors can be removed from the model without deleting their information content.

Another, only slightly more complex example are the *grandparent* relationships. For this purpose the above described rules need to be modified only minimally. Instead of following only one `Child` connector from target to source, the rule must follow exactly two `Child` connectors (two *hops*) in a row. The genders

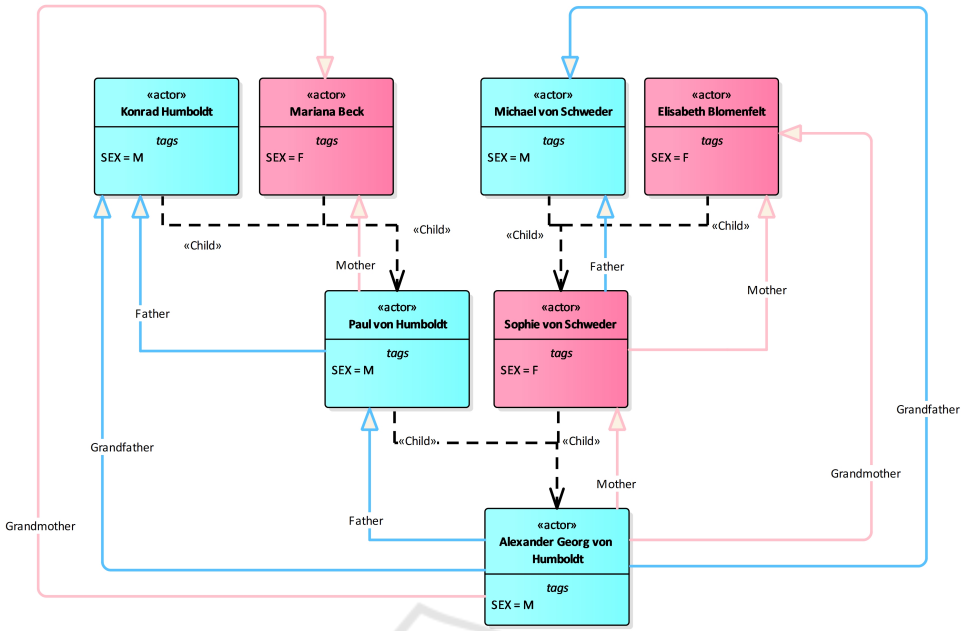

Figure 3: Small excerpt of the family tree shown in Figure 2.

of the intermediate steps are irrelevant for the evaluation of the current rules. Thereupon, the distinction between female and male works exactly as for the `Mother` and `Father` rule and replaces all relations `Grandmother` and `Grandfather` respectively.

Table 1: Overview of the number of connectors replaced by interaction rules in the family tree model.

Interaction-Rule	**Replaced connectors**
Daughter Rule	26
Son Rule	24
Mother Rule	25
Father Rule	25
Sister Rule	34
Brother Rule	28
Grandmother Rule	33
Grandfather Rule	33
Granddaughter Rule	30
Grandson Rule	36
Aunt Rule	15
Uncle Rule	17
Niece Rule	21
Nephew Rule	11
Sum	***358***

By using this approach in the example model from Figure 3, this model will result in only containing actors and `Child` connectors. Thus only 6 of the 16 connectors shown in Figure 3 are still part of the model, which is a reduction of 62.5%. However, the information content of the other 10 connectors is not lost, but merely abstracted in the form of interaction rules. With increasing number of elements and connectors in the model, interaction rules are a powerful tool to reduce the overall complexity, since a single rule can replace a large number of relations, and still remain comprehensible and maintainable.

Applying the InTra approach in the same way to the large family tree from Figure 2, using interaction rules for all family relationships `Daughter`, `Son`, `Mother`, `Father`, `Sister`, `Brother`, `Grandmother`, `Grandfather`, `Granddaughter`, `Grandson`, `Aunt`, `Uncle`, `Niece` and `Nephew` leads to even clearer results. Table 1 gives an overview about how many connectors could be replaced by each of the 14 interaction rules in the family tree example from Figure 2. A total of 358 connectors could be abstracted by rules. This leaves 50 child relations, from which all other connections can be derived through model transformation. This corresponds to a reduction of 87% of connectors in the family tree model from Figure 2. Nevertheless, this approach can be productively applied to other system models with greater practical relevance but similar reduction potential.

7 RULE DEFINITION

InTra was implemented as an addin for the modelling tool Enterprise Architect and was programmed in C#. With the help of this plug-in it is possible to use the

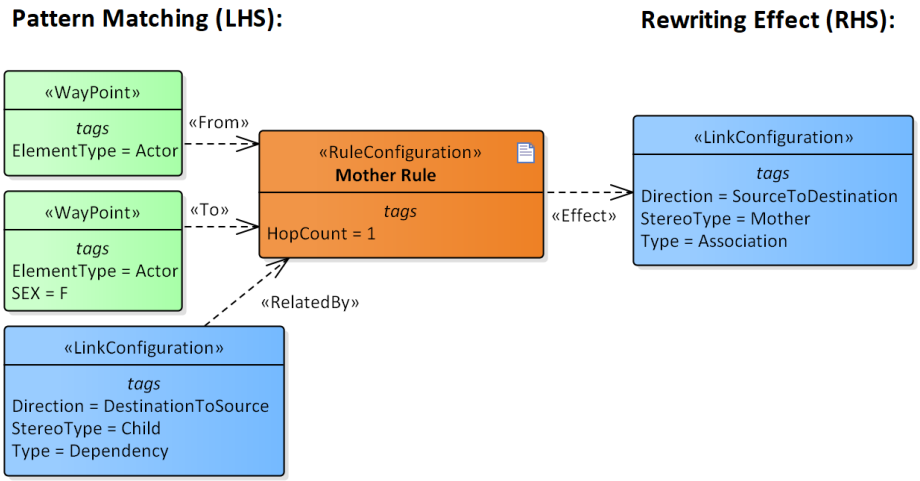

Figure 4: InTra `RuleConfiguration`: `Mother`.

proposed approach directly inside Enterprise Architect, without the need of additional tools or programming knowledge.

As presented in Figure 4, interaction rules in InTra are defined as model elements themselves, and thus become part of the model in which they are used. The shown rule consists of a `RuleConfiguration`, which holds the rule's `Name` and `HopCount`, a *Pattern Matching* (Left-Hand-Side) area and a *Rewriting* (Right-Hand-Side) area. The names chosen for these parts of a rule follow the common practice of several authors, including (Varró, 2007), (Wieber, 2015) and (Kahani, 2019), among others. In contrast to the representation of LHS and RHS in Figure 1 from (Varró, 2007), InTra does not use a direct subgraph representation for rule definition, but rather defines LHS and RHS with the help of *interaction rules*, by using diagram-based interaction constraints for both vertices and edges of the target model. The *Pattern Matching* area defines conditions which need to be fulfilled by elements (defined as `WayPoint` conditions) and connectors (defined as `LinkConfiguration` conditions), respectively, to define a certain pattern of interaction and thus find a set of valid interaction paths in the linked model segment. The *Rewriting* area describes the modifications that are made to the model if a valid interaction path is found and model transformation shall be executed. Rewriting effects can be the *creation* or the *removal* of certain kind of connectors between a start and end `WayPoint` of a found interaction path. For the definition of filter conditions the element's and connector's attributes `Name`, `Alias`, `EA-Type`, `Stereotype`, `minimal Multiplicity`, `maximal Multiplicity`, `Embedment` and any kind of `Tagged Value` can be analysed. Therefore the values of those attributes can be compared by direct *text comparison*, *regular expressions* and *numeric comparison* operators like <, <=, >, >=, == and ><, formulated as `Tagged Value` of a `WayPoint` or a `LinkConfiguration`. To define a *Pattern Matching* for possible interaction paths in the model, the mentioned conditions are connected with the core of the rule by the connectors `From`, `Over`, `To` and `RelatedBy`. A valid interaction path must begin with an element conforming to a `From WayPoint`, be linked to other elements by connectors conforming to a `RelatedBy LinkConfiguration`, end in a element conforming to a `To WayPoint` and must be connected to exactly `(HopCount-1)` elements in between, which are conforming to a `Over WayPoint`. Multiple `WayPoints` or `LinkConfigurations` of the same type will result in a logical `OR` for that specific condition pattern. While the rule shown in Figure 4 represents the most basic form of an interaction rule, a so called `RuleConfiguration`, there is also the possibility to combine several of these rather simple rules in a more complex rule, a so called `RuleChain`. Thus, `RuleConfigurations` can be thought of as atomic rule pieces that can be assembled into a more complex `RuleChain`. Rules can therefore be decomposed and offer high reusability. An example `RuleChain` for the family model from Figure 2 is the `Sister` rule shown in Figure 5. The rule evaluation of a `RuleChain` always starts with a `RuleChainStart` element. From there on, the contained `RuleConfigurations` are executed one after another (`AppendRule`), until a `RuleChainEnd` element is reached. Each interaction path found by a `RuleConfiguration` is thereby extended by the next `RuleConfiguration` in the chain, by passing all `To WayPoints` of one rule to the next one as possible `From WayPoints`. The resulting interaction path of a

`RuleChain` thus consists of start elements of the first, and end elements of the last rule of the chain. Thus, it is certainly not a successive execution of transformation rules, but rather a narrowing down of the solution space, based on conditions that sequentially build upon each other. Each `RuleChain` defines its own effects and reductions as `LinkConfigurations`, which are linked to the endpoint of the chain analogous to a `RuleConfiguration`, using an `Effect` or `Reduction` connector. In our Example from Figure 5 initially the `Parent RuleConfiguration` is evaluated, which identifies the parents of a person. Then, these elements are passed as potential `From WayPoints` of the `Daughter RuleConfiguration`. Evaluating this rule finds all daughters of a person, and in this particular case all daughters of a person who is parent of a different person. So, in summary, the two rules in succession find all *daughters* of all *parents* of a person, which corresponds to all *sisters* of that person. Some attentive readers may wonder at this point what happens if the source element itself is female. If the source element itself is female, it is correctly not identified as a sister of herself, because connectors that have already been visited by one of the `RuleConfigurations` are not visited again during *Pattern Matching* of a `RuleChain`. For this reason, it is also impossible to get caught in cycles during the model transformation. The fact that interaction rules themselves are also defined as model elements has the additional advantage that interaction rules themselves can also be transformed via other InTra rules. Due to this property InTra rules can be seen as rules of higher order (Mens, 2006).

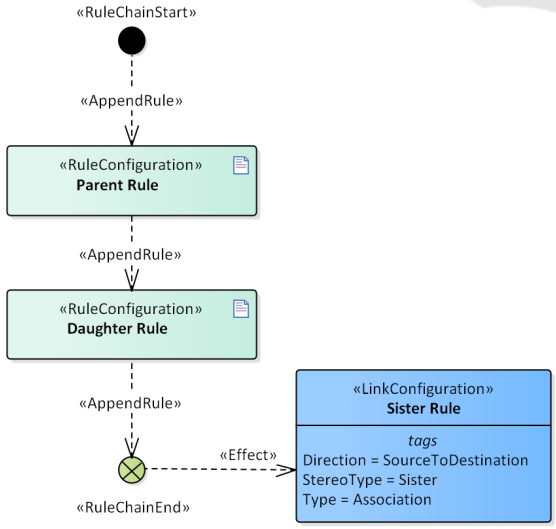

Figure 5: InTra `RuleChain`: `Sister`.

8 INDUSTRIAL APPLICATION RESULTS

In the communication model introduced in Section 3, the InTra approach could be successfully applied to abstract the communication relationships in multiple views of the model, thus highly reducing the complexity of the system. This has proven to be extremely helpful, especially with regard to the readability and maintainability of the model. By building 128 interaction rules and creating 255 links between rules and parts of the model, the number of information exchanges in the model could be reduced from 9,355 to 2,615. This is a reduction of 72 %. Therefore, it could be proven, that the InTra approach is able to be applied lucratively in realistic, productive systems and model environments. Furthermore, it was possible to explain the contents of the model to the domain experts more easily with the help of interaction rules. Subsequently, we received feedback from the future model users, that they perceived the new approach to be comprehensible and helpful.

9 RELATED WORK

In preparation for the development of the InTra approach, the current state of the art on the topic was examined, whereby it was necessary to find out, which generic requirements a transformation approach must fulfill in order to be considered suitable. These requirements were identified based on the taxonomies and comparative characteristics of the contributions of (Czarnecki, 2006), (Ghamarian, 2011), (Kahani, 2019) and (Mens, 2006). All listed works offered the possibility of identifying important criteria regarding the requirements for rule-creation, model transformation and general usability of the approach, but especially (Czarnecki, 2006) and (Czarnecki, 2002) provided a majority of useful properties and were able to give a good view on existing model transformation approaches. They also provided their findings based on a feature diagram according to (Kang, 1990) in a structured and comprehensible way. (Kahani, 2019) and (Varró, 2007) give an excellent introduction into the field of graph-based model transformation. Both (Valiente, 1997) and (McCreesh, 2020) are dealing with the challenge of the NP-complete complexity of the subgraph-isomorphism problem, which describes the usual run-time limit for the *Pattern Matching* in traditional model transformation approaches.

10 CONCLUSION

In this paper we investigated the usage of model transformation techniques to reduce the complexity of large system models, to improve the usability and comprehensibility. We elaborated on the common problems of system modeling, gave an overview about the field of graph-based model transformation, introduced our own rule-based model transformation approach InTra, described how rules can be defined and applied and gave an example of the results that can be expected by using the approach in large system models. We also highlighted the advantages of the approach and how it can be used to tackle the problems that may arise in large models. We were able to successfully use the approach in the industrial setting described in Section 3, and presented our results and experiences in Section 8. As application showcase we presented a family tree model in which arbitrary family relations could be derived using the existing child relations and the given basic structure of the model. We were able to show that the approach in this example was well suited to reduce the system complexity without affecting the information content of the model. The InTra rule-based modelling approach has the potential to make the growing complexity of today's system models more manageable, comprehensible and maintainable through the use of rules.

However, it is primarily necessary to identify further real industry scenarios, similar to the one from Section 3, in order to further test the approach for its abilities in real world scenarios. Potential cases should primarily have a clear basic system structure and reoccurring relation patterns. Likewise, the models should be quite large, in order to justify a rule-based reduction of connectors. Case studies based on further practical application could be used to investigate and enhance the InTra approach. In addition, it is necessary to investigate with further empirical studies, if a reduction in the number of connectors actually leads to a reduction in complexity in terms of human perception, and whether interaction rules are truly more comprehensible for the viewer than the sum of connectors they replace. Nevertheless, it should be considered that domain experts and system architects will perceive models in different ways, which could lead to diverging results.

REFERENCES

Antinyan, V. (2020). Revealing the Complexity of Automotive Software. Volvo Car Group.

Baduel, R. (2018). SysML Models Verification and Validation in an Industrial Context: Challenges and Experimentation.

Balasubramanian, D. (2006). The Graph Rewriting and Transformation Language: GReAT.

Braun, P. (2003). BOTL The Bidirectional Object Oriented Transformation Language.

Czarnecki, K. (2002). Classification of Model Transformation Approaches.

Czarnecki, K. (2006). Feature-Based Survey of Model Transformation Approaches.

Fujaba Development Group (2021). Fujaba Tool Website. https://web.cs.upb.de/archive/fujaba/. Website University of Paderborn.

Ghamarian, A. H. (2011). Modelling and analysis using GROOVE.

ISIS (2021). GReAT Tool Website. https://www.isis.vanderbilt.edu/tools/GReAT.

Kahani, N. (2019). Survey and classification of model transformation tools.

Kang, K. C. (1990). Feature-Oriented Domain Analysis (FODA) Feasibility Study.

Koning, H. (2002). Practical Guidelines for the Readability of IT-architecture Diagrams.

McCreesh, C. (2020). The Glasgow Subgraph Solver: Using Constraint Programming to Tackle Hard Subgraph Isomorphism Problem Variants.

McGill (2021). AToM3 Introduction Website. McGill University.

Mens, T. (2006). A Taxonomy of Model Transformation.

Purchase, H. C. (2000). User Preference of Graph Layout Aesthetics: A UML Study.

RWTH Aachen (2021). PROGRES Tool Website. http://www-i3.informatik.rwth-aachen.de.

Schürr, A. (2021). PROGRES - A Graph Transformation Programming Environment. https://www.es.tu-darmstadt.de/.

Stützel, B. (2021). Systems Engineering in Deutschland - Die deutsche Unternehmenslandschaft im Vergleich. Studie 2021 — Publisher: Prozesswerk GmbH.

Taentzer, G. (2003). AGG: A Graph Transformation Environment for Modeling and Validation of Software.

Twente (2021). GROOVE Tool Website. https://groove.ewi.utwente.nl/about. Twente University.

Valiente, G. (1997). An algorithm for graph pattern-matching.

Varró, D. (2007). The model transformation language of the VIATRA2 framework.

Warrilow, R. (2020). Lost in Translation? – How MBSE is Evolving to address Today's Complexity Challenges.

Wieber, M. S. (2015). Qualitätssicherung von Modelltransformationen - Über das dynamische Testen programmierter Graphersetzungssysteme.