Empirical Software Engineering (2024) 29:52
https://doi.org/10.1007/s10664-023-10428-2

Traceability and reuse mechanisms, the most important properties of model transformation languages

Stefan Höppner[1] · **Matthias Tichy**[1]

Accepted: 16 November 2023 / Published online: 24 February 2024

Abstract

Context Dedicated model transformation languages are claimed to provide many benefits over the use of general purpose languages for developing model transformations. However, the actual advantages and disadvantages associated with the use of model transformation languages are poorly understood empirically. There is little knowledge and even less empirical assessment about what advantages and disadvantages hold in which cases and where they originate from. In a prior interview study, we elicited expert opinions on what advantages result from what factors surrounding model transformation languages as well as a number of moderating factors that moderate the influence.

Objective We aim to quantitatively asses the interview results to confirm or reject the influences and moderation effects posed by different factors. We further intend to gain insights into how valuable different factors are to the discussion so that future studies can draw on these data for designing targeted and relevant studies.

Method We gather data on the factors and quality attributes using an online survey. To analyse the data and examine the hypothesised influences and moderations, we use universal structure modelling based on a structural equation model. Universal structure modelling produces significance values and path coefficients for each hypothesised and modelled interdependence between factors and quality attributes that can be used to confirm or reject correlation and to weigh the strength of influence present.

Results We analyzed 113 responses. The results show that the MTL capabilities Tracing and Reuse Mechanisms are most important overall. Though the observed effects were generally 10 times lower than anticipated. Furthermore, we found that moderation effects need to be individually assessed for each influence on a quality attribute. The moderation effects of a single moderating variable vary significantly for each influence, with the strongest effects being 1000 times higher than the weakest.

Conclusion The empirical assessment of MTLs is a complex topic that cannot be solved by looking at a single stand-alone factor. Our results provide clear indication that evaluation

Communicated by: Maria Teresa Baldassarre and Tayana Conte

This article belongs to the Topical Collection: *Registered Reports*

✉ Stefan Höppner
stefan.hoeppner@uni-ulm.de

Extended author information available on the last page of the article

should consider transformations of different sizes and use-cases that go beyond mapping one elements attributes to another. Language development on the other hand should focus on providing practical, transformation specific reuse mechanisms that allow MTLs to excel in areas such as maintainability and productivity compared to GPLs.

Keywords Survey · Universal structure modeling · Model transformation language · DSL · Model transformation · MDSE · Advantages · Disadvantages · Quantitative analysis

1 Introduction

Model driven engineering (MDE) envisions the use of model transformations as a main activity during development (Sendall and Kozaczynski 2003). Studies show that MDE can help with analysis of complex systems during design (Evora et al. 2014) and even reduce development effort by a factor of 2 (Baker et al. 2005). This is especially true for industries where high degree of domain knowledge is required and many domain experts are involved during system design and development, e.g., embedded systems (Liebel et al. 2016). The recent resurgence of low/no-code development highlights the importance of models as central development artefacts for non-software developers. When practising MDE, model transformations are used for a wide array of tasks such as manipulating and evolving models (Metzger 2005), deriving artefacts like source code or documentation, simulating system behaviour or analysing system aspects (Schmidt 2006a).

Numerous dedicated model transformation languages (MTLs) of different form, aim and syntax (Kahani et al. 2019) have been developed to aid with model transformations. Using MTLs is associated with many benefits compared to using general purpose languages (GPLs), though little evidence for this has been brought forth (Götz et al. 2021). The number of claimed benefits is enormous and includes, but is not limited to, better *Comprehensibility*, *Productivity* and *Maintainability* as well as easier *development* in general (Götz et al. 2021). The existence of such claims can partially be attributed to the advantages that are ascribed to domain specific languages (DSLs) (Hermans et al. 2009; Johannes et al. 2009).

In a prior systematic literature review, we have shown that it is still uncertain whether these advantages exist and where they arise from (Götz et al. 2021). Due to this uncertainty it is hard to convincingly argue the use of MTLs over GPLs for transformation development. This problem is exacerbated when considering recent GPL advancements, like Java Streams, LINQ in C# or advanced pattern matching syntax, that help reduce boilerplate code (Höppner et al. 2021) and have put them back into the discussion for transformation development. Even a community discussion held at the 12th edition for the International Conference on Model Transformations (ICMT'19) acknowledges GPLs as suitable contenders (Cabot and Gerard 2019). Moreover, the few existing empirical studies on this topic provide mixed and limited results. Hebig et al. found no direct advantage for the development of transformations, but did find an advantage for the comprehensibility of transformation code in their limited setup (Hebig et al. 2018). A study conducted by us, found that certain use cases favour the use of MTLs, while in others the versatility of GPLs prevails (Höppner et al. 2021). Overall there exists a gap in knowledge in what the exact benefits of MTLs are, how strong their impact really is and what parts of the language they originate from.

To bridge this gap, we conducted an interview study with 56 experts from research and industry to discuss the topic of advantages and disadvantages of model transformation languages (Höppner et al. 2022). Participants were queried about their views on the advantages

and disadvantages of model transformation languages and the origins thereof. Responses were analysed using qualitative content analysis (Kuckartz 2014). The focus was on identifying factors influencing participants' beliefs regarding the existence of specific quality properties of MTLs. We also tried to elicit their reasons for having these beliefs. The results point towards three main-areas that are relevant to the discussion, namely *General Purpose Languages Capabilities*, *Model Transformation Languages Capabilities* and *Tooling*. From the responses of the interviewees we identified which claimed MTL properties are influenced by which sub-areas and why. They also provided us with insights on moderation effects on these interdependencies caused by different *Use-Cases*, *Skill & Experience levels* of users and *Choice of Transformation Language*.

All results of the interview study are qualitative. They do not provide indication on the strength of influence between the involved variables. Therefore they represent an initial data set that also requires quantitative analysis.

In this paper, we report on the results of a study to *confirm or deny* the interdependencies hypothesised from our interview results. We provide *quantification of the influence strengths* and *moderation effects*. To ensure a more complete theory of interactions, we also present the results of exploring interdependencies between factors and quality properties not hypothesised in the interviews.

Due to limited resources, this study focuses on the effects of *MTL capabilities* (namely Bidirectionality, Incrementality, Mappings, Model Management, Model Navigation, Model Traversal, Pattern Matching, Reuse Mechanisms and Traceability) on *MTL properties* (namely Comprehensibility, Ease of Writing, Expressiveness, Productivity, Maintainability and Reusability and Tool Support) in the context of their *uses-case* (namely bidirectional or unidirectional, incremental or non-incremental, meta-model sanity, meta-model, model and transformation size and semantic gap between input and output), the *skills & experience* of users and *language choice*. Further studies can follow the same approach and focus on different areas. Descriptions for all MTL capabilities and MTL properties can be found in Sect. 2 and thorough explanations can be found in the works of Götz et al. (2021); Höppner et al. (2022).

The goal of our study is to provide quantitative results on the influence strengths of interdependences between model transformation language *Capabilities* and claimed *Quality Properties* as perceived by users. Additionally we provide data on the strength of moderation expressed by *contextual properties*. The study is structured around the hypothesised interdependencies between these variables, and their more detailed breakdown, extracted from the previous interview study. Each presumed influence of a *MTL capability* on a *MTL property* forms one hypothesis which is to be examined in this study. All hypotheses are extended with an assumption of moderation by the context variables. The system of hypotheses that arises from these deliberations is visualised in a structure model, which forms the basis for our study. The structure model is depicted in Fig. 1. The model shows exogenous variables on the left and right and endogenous variables at the centre. Exogenous variables depicted in a ellipse with a dashed outline constitute the hypothesised moderating variables.

All hypotheses investigated in our study are of the form: *"<MTL Property> is (positively or negatively) influenced by <MTL Capability>"*. They are represented by arrows from exogenous variables on the left of Fig. 1 to endogenous variable at the centre. A moderation on the hypothesised influence is assumed from all exogenous variables on the right of the figure connected to the considered endogenous variable. In total we investigate 31 hypothesised influences, i.e. the number of outgoing arrows from the exogenous variables on the left of Fig. 1.

Our study is guided by the following research questions:

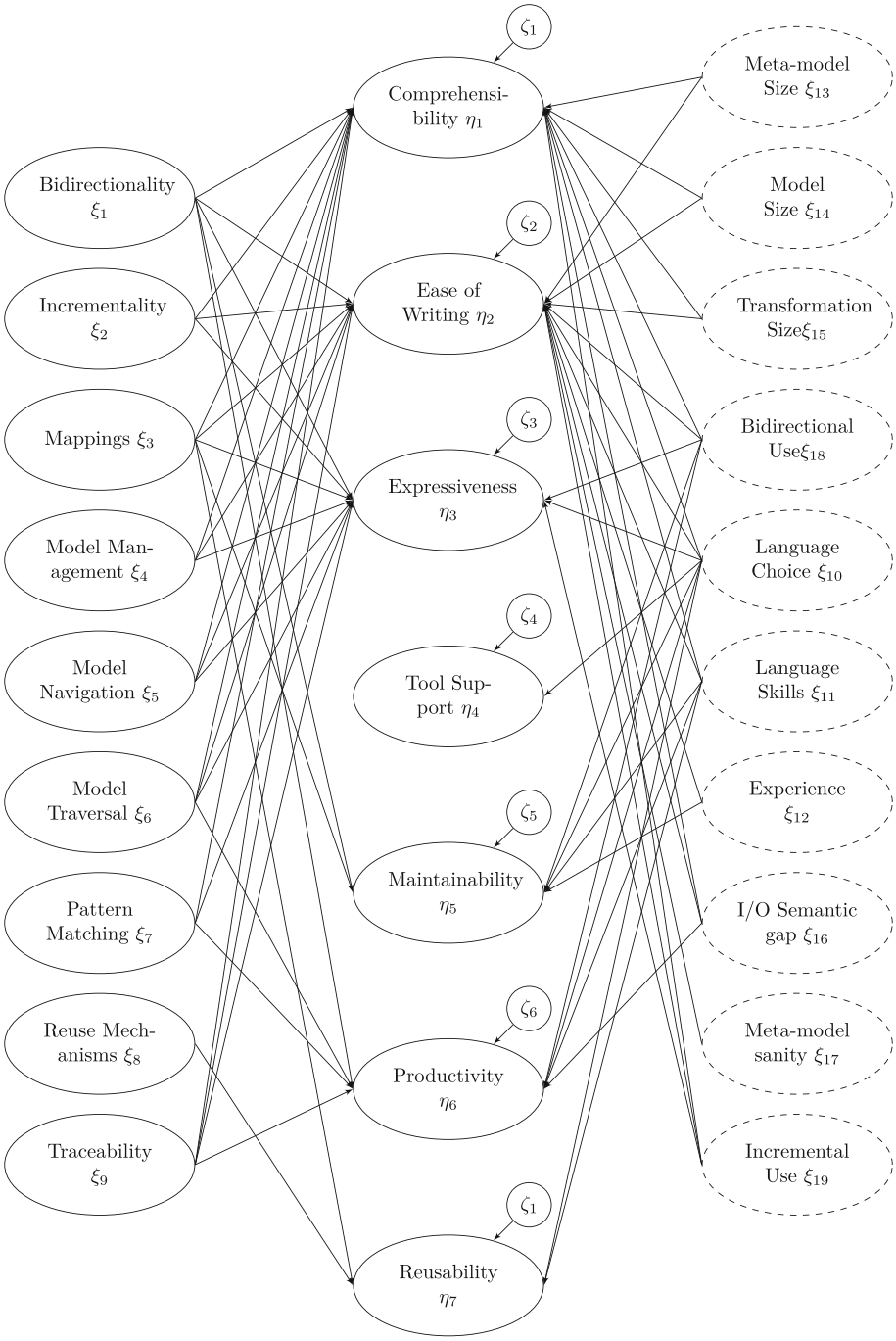

Fig. 1 Structure model depicting the hypothesised influence and moderation effects of factors on MTL properties

RQ1 Which of the hypothesised interdependencies withstands a test of significance?
RQ2 How strong are the influences of model transformation language capabilities on the properties thereof?
RQ3 How strong are moderation effects expressed by the contextual factors *use-case*, *skills & experience* and *MTL choice*?
RQ4 What additional interdependencies arise from the analysis that were not initially hypothesised?

As the first study on this subject it contains confirmatory and exploratory elements. We intend to confirm which of the interdependencies between *MTL capabilities*, *MTL properties* and *contextual properties* withstand quantitative scrutiny (**RQ1**). We explore how strong the influence and moderation effects between variables are (**RQ2 & RQ3**), to gain new insights and to confirm their significance and relevance (minor influence strengths might suggest irrelevance even if goodness of fit tests confirm a correlation that is not purely accidental). Lastly, we utilise the exploratory elements of USM to identify interdependencies not hypothesised by the experts in our interviews (**RQ4**).

We use an *online survey* to gather data on language use and perceived quality of researchers and practitioners. The responses are analysed using universal structure modelling (USM) (Buckler and Hennig-Thurau 2008) based on the structure model developed from the interview responses. This results in a quantified structure model with influence weights, significance values and effect strengths.

Based on the responses from 113 participants, the key contributions of this paper are:

- An adjusted structure model with newly discovered interdependencies;
- Quantitative data on the influence weight and effect strength of all factors as well as significant values for the influences;
- Quantitative data on the moderation strength of context factors;
- An analysis of the implications of the results for further empirical studies and language development;
- Reflections on the use of USM for investigating large hypotheses systems in software engineering research;

The method used in the reported study has been reviewed and published as part of the Registered Reports track at ESEM'22 (Höppner and Tichy 2022).

The structure of this paper is as follows: Sect. 2 provides an extensive overview of model-driven engineering, domain-specific languages, model transformation languages and structural equation modelling as well as universal structure modelling. Afterwards, in Sect. 3 the methodology is outlined. Demographic data of the responses is reported in Sect. 4 and the results of analysis is presented in Sect. 5. In Sect. 6 we discuss implications of the results and report our reflections on the use of USM. Section 7 discusses threats to validity of our study and how we met them. Lastly, in Sect. 8 we present related work before giving concluding remarks on our study in Sect. 9.

2 Background

In this section we provide the necessary background for our study.

2.1 Models & Model-driven Engineering

A multitude of model-driven approaches exists, each with a slightly different focus. In this paper, we focus on model-driven engineering (MDE), as it encompasses all the other approaches. We use the definition given by Brambilla et al. (2017).

According to Stachowiak (1973) models are "*a representation of entities and relationships in the real world with a certain correspondence for a certain purpose*". Such models are used as the central artefact in MDE (Brambilla et al. 2017). They are used for developing the intended solutions as well as to describe and reason about the problem domain (Brown et al. 2005). Using models in this way is claimed to be advantageous over regular development because the models can easily express domain-related concepts in an understandable fashion (Selic 2003).

The goal when applying MDE is to automatically generate artefacts from models. The generated artefacts can be models themselves or they can be other parts used in the running system that is being developed.

Brambilla et al. (2017) explain that the implementation of a system is spread over three levels. The **modelling level**, where models are defined. The **realisation level**, where the solutions are implemented through artefacts in use within the running system. And the **automation level**, where transformations from models to artefacts are defined. An overview of the relationship between the three levels can be found in Fig. 2.

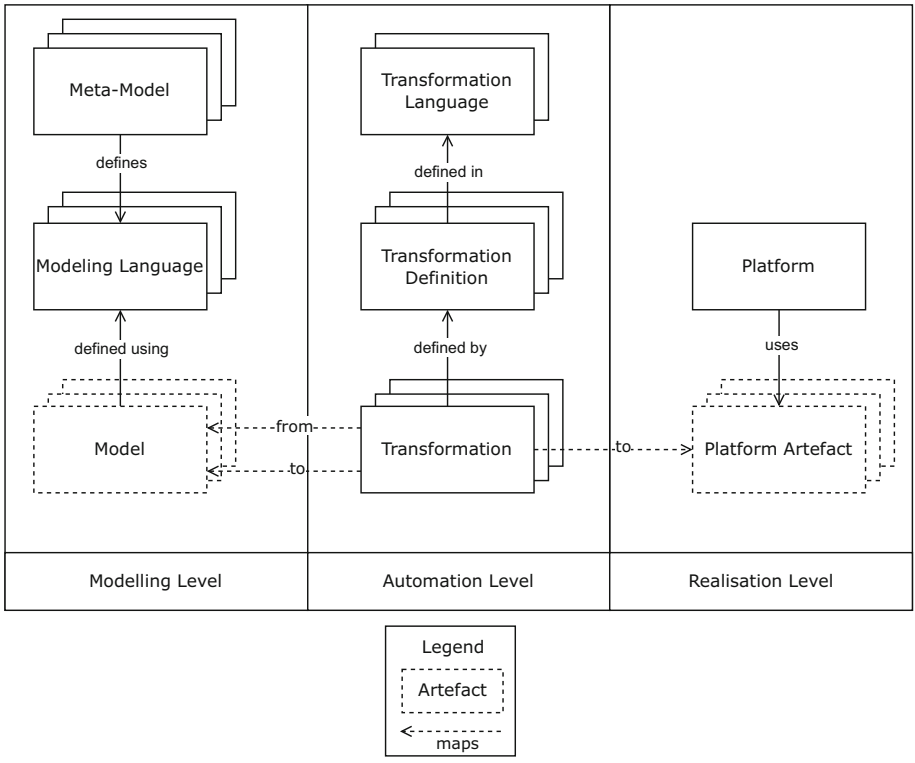

Fig. 2 Overview of MDE adapted from Brambilla et al. (2017)

Within this context, meta-models play an important role. Meta-models define how models that adhere to them are structured. They define an application domain for which models can be created. Each model is thus written in a modelling language defined through a meta-model (Bezivin 2004). Meta-models themselves also adhere to their own meta-models. To be able to define these higher-level meta-models the Object Management Group (OMG) released the modelling standard *Meta-object Facility* (MOF) (OMG 2002). Concrete implementations of MOF are e.g., the *Eclipse Modelling Framework* (EMF) (Steinberg et al. 2008) and the *.NET Modelling Framework* (Hinkel 2016).

In summary MDE describes a top-down approach to automatically generate executable solutions based on abstract models (Selic 2003; Schmidt 2006b). The (automatic) transformations from one model into other artefacts are called *model transformations* (MTs). They form the most essential part within MDE because they connect the modelling level with the realisation level. This is often stated in literature (Sendall and Kozaczynski 2003; Metzger 2005). Model transformations can be developed through the use of general-purpose programming languages (GPLs) or through the use of dedicated languages called model transformation languages (MTLs).

2.2 Domain-specific Languages

Fowler (2011) defines domain-specific languages (DSLs) as languages focused on one particular aspect. Their notation is designed by focusing on relevant features within the target domain (Van Deursen and Klint 2002). Thus, they provide domain specific language constructs allowing developers to directly express domain concepts. This, in turn, can increase ease of development and reduce the barrier of entry for non experts to understand what is written (Sprinkle et al. 2009; Fowler 2011). DSLs are intended to provide a suitable alternative to using general-purpose tools for solving problems for a specific domain.

Examples of domain specific languages are *shell scripts* in Unix operating systems (Kernighan and Pike 1984), HTML (Raggett et al. 1999) for designing web pages or AADL an architecture design language (SAEMobilus 2004).

DSLs can be either *internal* or *external* languages Fowler (2011). External DSLs generally require a separate parser and execution environment. Examples of external DSLs are SQL (**Codd1970**) and CSS (**W3C2021**). Internal DSLs, on the other hand, do not require separate parsers or runtime environments. Instead, they are a form of API within a general-purpose language referred to as fluent interfaces (Fowler 2011). This means they allow an eloquent definition of DSL expressions that can be read much like a standard sentence in a natural language. LINQ (**Meijer2006**), a language integrated in.NET, and the declarative Java API to create mock objects JMock (**Freeman2004**) fall in this category of DSLs.

2.3 Model Transformation Concepts and Languages

As stated in Sect. 2.1, a model transformation (MT) is the process of (automatically) transforming one model into another artefact. Model transformations are the most integral part of MDE. To aid in developing them ample domain specific languages, so called model transformation languages (MTLs) have been developed (Arendt et al. 2010; Balogh and Varro 2006; Jouault et al. 2006; Kolovos et al. 2008; Horn 2013; George et al. 2012; Hinkel and Burger 2019). They provide explicit language constructs for many tasks involved in model transformations such as model matching.

There exist several works that classify the functionalities in model transformations and makeup of model transformation languages Czarnecki and Helsen (2006); Kahani et al. (2019); Mens and Gorp (2006). For the purpose of this paper, we will only be explaining those features that are relevant to our study and discussion in Sects. 2.3.1, 2.3.2, 2.3.3, 2.3.4, 2.3.5, 2.3.6, and 2.3.7. Table 1 provides an overview over the presented features.

Please refer to Czarnecki and Helsen (2006); Kahani et al. (2019); Mens and Gorp (2006) for complete classifications.

2.3.1 External and Internal Transformation Languages

Like domain-specific languages, MTLs can be distinguished based on whether they are embedded in another language or whether they are completely independent languages. Languages embedded in a host language are called *internal* languages. Examples for internal MTLs are *FunnyQT* (Horn 2013) a language embedded in Clojure, *NMF Synchronizations* and the *.NET transformation language* (Hinkel and Burger 2019) embedded in C#, and *RubyTL* (Cuadrado et al. 2006) embedded in Ruby.

Fully independent languages are called *external* languages. Prominent examples thereof are *Atlas transformation language* (ATL) (Jouault et al. 2006), one of the most widely known model transformation languages, the graphical transformation language Henshin (Arendt et al. 2010) as well as the transformation framework called VIATRA (Balogh and Varro 2006).

2.3.2 Transformation Rules

Czarnecki and Helsen (2006) describe rules as being *"understood as a broad term that describes the smallest units of [a] transformation [definition]"*. Transformation rules take

Table 1 MTL feature overview

Feature	Characteristic	Representative language
Embeddedness	Internal	FunnyQT (Clojure), RubyTL (Ruby), NMF Synchronizations (C#)
	External	ATL, Henshin, QVT
Rules	Explicit Syntax Construct	ATL, Henshin, QVT
	Repurposed Syntax Construct	NMF Synchronizations (Classes), FunnyQT (Macros)
Location determination	Automatic Traversal	ATL, QVT
	Pattern Matching	Henshin
Directionality	Unidirectional	ATL, QVT-O
	Bidirectional	QVT-R, NMF Synchronisations
Incrementality	Yes	NMF Synchronizations
	No	RubyTL
Tracing	Automatic	ATL, QVT
	Manual	NMF Synchronizations
Dedicated model navigation syntax	Yes	ATL (OCL), QVT (OCL), Henshin (implicit in rules)
	No	NMF Synchronizations, FunnyQT, RubyTL

```
public void methodExample(Wolf w) {
  System.out.println(w.getName());
}
public void methodExample2(Wolf w) {
  Male target = new Male();
  target.setFullName(w.getName() + " Wolf");
  REGISTRY.register(target);
}
```

List. 1 Example Java methods

on many different forms depending on the language. For example in ATL they are implemented via an explicit language construct called rule, while in GPLs they are often defined as functions, methods or procedures that implement a transformation from input elements to output elements.

The fundamental difference between model transformation languages and general-purpose languages that originates in this definition, lies in dedicated constructs that represent rules. When looking at GPLs, the difference between a transformation rule and any other function or procedure is not represented explicitly. It can only be made based on their contents. An example of this can be seen in List. 1, which contains exemplary Java methods.

In MTLs, on the other hand, rules tend to be dedicated language constructs that allow the explicit definition of a *mapping* between input and output elements. The example rules written in the model transformation language ATL in List. 2 make this apparent. They define mappings between model elements of type `Wolf` and model elements of type `Male` as well as between `Wolf` and `Female` using *rules*, a dedicated language construct for defining transformation mappings. The transformation is a modified version of the well known Families2Persons transformation case (Anjorin et al. 2017).

```
rule Wolf2Male {
  from
    s : Member (not s.isFemale())
  to
    t : Male (
      fullName <- s.name + ' Wolf'
    )
}

rule Wolf2Female {
  from
    s : Member (s.isFemale())
  to
    t : Female (
      fullName = s.name + ' Wolfess'
      partner = s.companion
    )
}
```

List. 2 Example ATL rules

2.3.3 Rule Application Control: Location Determination

Location determination describes the strategy that is applied to find those elements within a model that should be transformed by a transformation rule (Czarnecki and Helsen 2006). Most model transformation languages such as ATL, Henshin, VIATRA or QVT (OMG 2016), rely on some form of *automatic traversal* strategy for this.

We differentiate between two strategies of location determination, based on what matching that takes place during traversal. Languages such as ATL or QVT use the basic *automatic traversal*, where single elements are matched to which transformation rules are applied. The other type is *pattern matching*, which is used in languages like Henshin. In this type of location determination a model- or graph-*pattern* is matched to which rules are applied. This does allow the definition of sub-graphs consisting of several model elements and references between them which are then manipulated by a rule. The two location determination strategies cover two vastly different use-cases. *Automatic traversal* mainly aims at finding single elements to use for creating new elements. *Pattern-matching* mainly aims at finding structures and manipulating them or their elements directly.

General-purpose programming languages typically do not incorporate either location determination strategy. Instead, they depend on developers to manually write code for identifying the appropriate elements to which rules should be applied.

The *automatic traversal* of ATL applied to the example from List. 2 will result in the transformation engine automatically executing the `Wolf2Male` on all individual model elements of type `Wolf` where the function `isFemale()` returns `false` and the `Wolf2Female` on all other model elements of type `Member`.

The *pattern matching* of Henshin can be demonstrated using Fig. 3. It describes a transformation that creates a couple connection between two persons that like each other. When the transformation is executed the transformation engine will try and find instances of the defined graph pattern and apply the changes on the found matches. This example also demonstrates the main difference between *automatic traversal* and *pattern matching*. The engine will search for a sub graph within the model instead of finding a single model element.

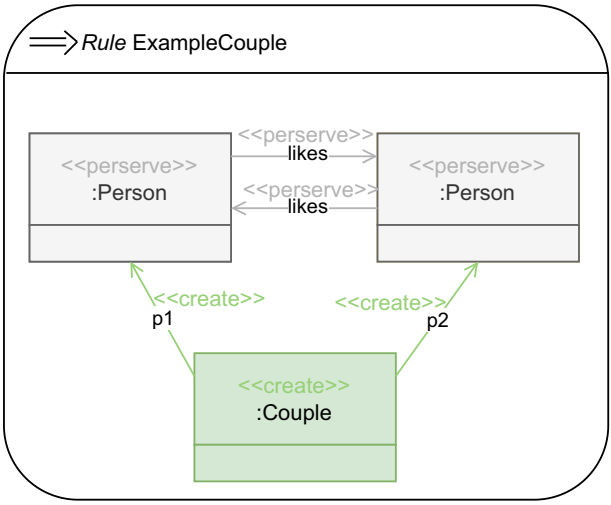

Fig. 3 Example Henshin transformation

```
top relation Wolf2Male {
  n, name : String;
  domain Wolfs s:Wolf {
    name = n };
  domain Wolfssex t:Male {
    name = fullName};
  where {
    fullName = n + ' Wolf'; };
}
```

List. 3 Example QVT-R relation

2.3.4 Directionality

The directionality of a model transformation describes whether it can be executed in one direction, called a unidirectional transformation or in multiple directions, called a multidirectional transformation (Czarnecki and Helsen 2006).

Distinguishing between unidirectional and bidirectional transformation languages is relevant to this study. Many bidirectional languages allow for executing transformations in both ways based solely on one transformation rule definition. Other bidirectional languages require rules for both directions to be defined explicitly. Such languages distinguish themselves from unidirectional languages solely through the fact, that the definitions for both directions can be made next to each other in the same module. General-purpose languages can not provide bidirectional support and also require both directions to be implemented explicitly in two separate transformation definitions.

The transformation definition in List. 2 defines a unidirectional transformation. The input and output of the transformation are fix and the transformation can only be executed in that direction.

In contrast, the QVT-R relation in List. 3 is a bidirectional transformation definition (For simplicity reasons the transformation omits the condition that males are only created from wolfs that are not female). The transformation does not define input and output. Instead it defines how two elements of the respective domains relate to one another. As a result given a `Wolf` element its corresponding `Male` elements can be inferred, and vice versa.

2.3.5 Incrementality

Incrementality of a transformation describes whether existing models can be updated based on changes in the source models without rerunning the complete transformation (Czarnecki and Helsen 2006). Some refer to this as model synchronisation.

Incrementality implementations require active monitoring of the input and output models. They also need information on which rules affect what parts of the models. If a change in the input model occurs, this information is used to determine which rules should be executed on the changed elements to produce the synchronized target elements. This sometimes requires additional management tasks to be executed to keep synchronized models valid and consistent.

2.3.6 Tracing

Tracing *"is concerned with the mechanisms for recording different aspects of transformation execution, such as creating and maintaining trace links between source and target model elements"* Czarnecki and Helsen (2006).

Many dedicated transformation languages, such as ATL and QVT have automated mechanisms to manage trace information. In these languages, the traces are automatically created during transformation execution. The trace information is used to provide seamless access to the target elements based on their sources. Additionally, some languages allow the information to be accessed through special syntax constructs.

An example of tracing in action can be seen in `line 16` of List. 2. Here the `partner` attribute of a `Female` element that is being created, is assigned to `s.companion`. The `s.companion` reference points towards a element of type `Wolf` within the input model. When creating a `Female` or `Male` element from a `Wolf` element, the ATL engine will resolve this reference into the corresponding element, that was created from the referred `Wolf` element via either the `Wolf2Male` or `Wolf2Female` rule.

2.3.7 Dedicated Model Navigation Syntax

Dedicated query languages or syntax constructs for navigating models is not part of any feature classification for model transformation languages. However, it was an often discussed topic in the preceding interview study (Höppner et al. 2022) and therefore an explanation is required for this study too.

Languages such as OCL (OMG 2014), which are often embedded in transformation languages like ATL, provide dedicated syntax for querying and navigating models. They provide syntactical constructs that aid users in navigation tasks. Their aim is to ease access to models by not requiring users to implement queries using loops or other general-purpose constructs. To do so, OCL provides a functional approach for accumulating and querying data based on collections. In contrast, Henshin combines model navigation with its pattern matching enabling users to define graph patterns to find the sought-after model elements.

2.4 MTL Quality Properties

There exists a large body of quality properties that get associated with model transformation languages. In literature many claims are made about advantages or disadvantages of MTLs in these different properties. These properties were previously categories by Götz et al. (2021). This study focuses on a subset of all the identified quality properties of MTLs which requires them to be properly explained. In this section, we give a brief description of our definitions of each of the quality properties of MTLs relevant to the study.

Comprehensibility describes the ease of understanding the purpose and functionality of a transformation based on reading code.

Ease of Writing describes the ease at which a developer can produce a transformation for a specific purpose.

Expressiveness describes the amount of useful dedicated transformation concepts in a language.

Productivity describes the degree of effectiveness and efficiency with which transformations can be developed and used.

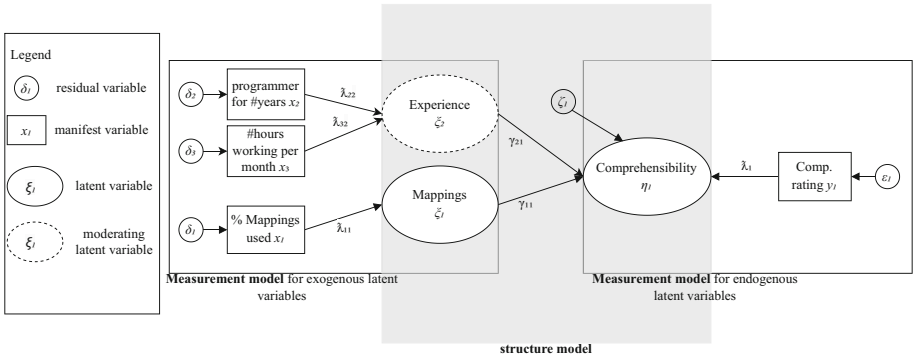

Fig. 4 The makeup of a structural equation model. (Höppner and Tichy 2022)

Maintainability describes the degree of effectiveness and efficiency with which a transformation can be modified.

Reusability describes the ease of reusing transformations or parts of transformations to create new transformations (with different purposes).

Tool Support describes the amount of quality tools that exist to support developers in their efforts.

2.5 Structural Equation Modelling and (Universal) Structural Equation Modelling

Structural Equation Modelling (SEM) is an approach used for confirmatory factor analysis (Graziotin et al. 2021). It defines a set of methods used to *"investigate complex relationship structures between variables and allows for quantitative estimates of interdependencies thereof. Its goal is to map the a-priori formulated cause-effect relationships into a linear system of equations and to estimate the model parameters in such a way that the initial data, collected for the variables, are reproduced as well as possible"* (Weiber and Muhlhaus 2021).

In structural equation modelling a distinction between two sets of variables (*manifest* & *latent*) is made. *Manifest* variables are empirically measured. *Latent* variables are not measured directly and describe theoretical constructs that are hypothesised to interact with each other. They are further divided into *exogenous* or independent and endogenous or *dependent* variables.

At the heart of SEM are so called structural equation models, a sample of which can be seen in Fig. 4. structural equation models are made up of three connected sub-models. The *structure model*, which defines all hypothesised interactions between exogenous (ξ_{exID}) and endogenous (η_{endID}) latent variables. The *measurement model* of the exogenous latent variables, which reflects the relationships between all exogenous latent variables and their associated manifest variables. The *measurement model* of the endogenous latent variables, which reflects the relationships between all endogenous latent variables and their associated manifest variables.

In the *structure model* each exogenous variable is linked, by arrow, to all endogenous variables that are presumed to be influenced by it. Each connection is given a variable (γ_{exID_endID}) that measures the influence strength. If an exogenous variable *moderates* the influences on another endogenous variable, the exogenous variable is depicted with a dashed

outline and connected to all endogenous variables that are moderated by it[1] For each moderated influence a variable of the form $\gamma_{exID\_endID\_modEndID}$ is assigned. Additionally, an error variable is appended to each endogenous latent variable. The error variable represents the influence of variables not represented in the model.

Figure 4 shows an example structure equation model model for the hypothesis that *"Mappings help with the comprehensibility of transformations, depending on the developers experience."* Höppner and Tichy (2022). The structure model at the centre of the figure, is comprised of the exogenous latent variable ξ_1 (*Mappings*), the moderating exogenous variable ξ_2 (*Experience*), the endogenous latent variable η_1 (*Comprehensibility*), a presumed influence of *Mappings* on *Comprehensibility* via γ_{11} and the error variable ζ_1. Lastly, the model contains a moderation effect of Experience on all influences of Comprehensibility. This moderation effect is assigned by the variable γ_{11_2}. The moderation variables are not depicted in our graphical representation of the structure model because of their high number and associated visibility issues.

In the *measurement model* each manifest variable is linked, by arrow, to all exogenous latent variables that are measured through it. For each connection a variable is assigned that measures the indication strength of the manifest variable for the latent variable. An error variable for each manifest variable is also introduced. They represent measurement errors.

In Fig. 4, the *measurement model for exogenous latent variables* is seen at the left of the figure. It is comprised of the exogenous latent variables ξ_1 (*Mappings*) and ξ_2 (*Experience*), the manifest variables x_1 (*% of code using Mappings*), x_2 (*number of years a person has been a programmer*) and x_3 (*number of hours per month spent developing transformations*) their measurement accuracy for Mapping usage λ_{11} and their measurement accuracy for Experience λ_{22} & λ_{32} and the associated measurement error δ_1, δ_2 & δ_3.

The *measurement model for endogenous latent variables*, seen at the right side of Fig. 4, is structured the same way as the measurement model for exogenous latent variables.

Structural equation modelling calls for estimation of influence weights and latent variables within a given structural equation model. Traditional methods (covariance-based structural equation modeling & partial least squares) use different mathematical approaches such as maximum-likelihood estimation or least squares (Weiber and Muhlhaus 2021).

Universal Structure Modeling (USM) is an exploratory approach that complements the traditional confirmatory SEM methods (Buckler and Hennig-Thurau 2008). It combines and enhances the iterative methodology of partial least squares with a Bayesian neural network approach using multilayer perceptron architecture. USM derives a starting value for latent variables in the model via principal component analysis. The Bayesian neural network is then used to discover a system of linear and non-linear interaction paths between the variables within the structure model. This enables USM to identify complex relationships that may not be detected using traditional SEM approaches including hidden structures within the data and highlights unproposed model paths, nonlinear relations among model variables, and moderation effects.

The primary measures calculated in USM are the 'Average Simulated Effect' (ASE), 'Overall Explained Absolute Deviation' (OEAD), 'interaction effect' (IE) and 'parameter significance'. ASE measures the average change in the endogenous variable resulting from a one-unit change in the exogenous variable across all simulations. OEAD assesses the degree

[1] Usually moderation is illustrated via arrows from the moderating exogenous variable to the arrow representing the moderated influence, i.e., an arrow between an exogenous variable and an endogenous variable. However our illustration deviates from this due to the size and makeup of our hypothesis system. Standard representations can be found in the basic literature such as Weiber and Muhlhaus (2021).

of fit between the observed and simulated values of the endogenous variable, capturing the overall explanatory power of the model. IE evaluates the extent to which the effect of one exogenous variable on the endogenous variable depends on the level of another variable. Parameter Significance determines whether the estimated coefficients for each exogenous variable in the model are statistically significant at a predetermined level of confidence which indicated if the exogenous variable has a meaningful impact on the endogenous variable and is calculated through a bootstrapping routine (Mooney et al. 1993). These metrics together provide a comprehensive assessment of the performance and explanatory power of a USM model.

USM is recommended for use in situations where traditional SEM approaches may not be sufficient to fully explore the relationships between variables. Using USM instead of traditional structural equation modelling approaches is suggested for studies where there are still uncertainties about the completeness of the underlying hypotheses system and for exploring non-linearity in the influences (Weiber and Muhlhaus 2021; Buckler and Hennig-Thurau 2008). Furthermore, the requirements for the scale levels of data is reduced. This enables the use of categorical variables in addition to metric variables (Weiber and Muhlhaus 2021).

At present, the tool NEUSREL[2] is the only tool available for conducting USM.

3 Methodology

The methodology used in this study has been reviewed and published as part of the Registered Reports track at ESEM'22 (Höppner and Tichy 2022). In the following, we provide a more detailed description and highlight all deviations from the reported method as well as justification for the changes.

The study itself is comprised of the following steps which were executed sequentially and are reported on in this section.

1. Development of survey methodology.
2. Submission to the Registered Reports track at EMSE'22.
3. Methodology revision based on feedback.
4. Development of online survey using an on premise version of the survey tool LimeSurvey.[3]
5. Survey review and pilot test by co-authors.
6. Reworking survey based on pilot test.
7. Opening online survey to public.
8. Reaching out to potential survey subjects per mail and social media.
9. Closing of online survey (9 weeks after opening).
10. Data extraction.
11. Data analysis using the USM tool NEUSREL.

The steps executed differ in two ways from those reported in the registered report. First, we do not contact potential participants for a second time after two weeks. This was deemed unnecessary based on the number of participants at that point in time. Moreover, we did not want to bother those that participated already and had no way of knowing their identity.

[2] https://www.neusrel.com

[3] https://www.limesurvey.org/

Second, we kept the survey open 3 weeks longer than intended due to receiving several requests to do so.

3.1 Survey Design

In this section we detail the design of the used questionnaire and methodology used to develop and distribute it.

3.1.1 Questionnaire

The questions in the questionnaire are designed to query data for measuring the latent variables from the structure model in Fig. 1. The complete questionnaire can be found in the open access repository of Ulm University (Höppner and Tichy 2023). In the following, we describe each latent variable and explain how we measure it through questions in the questionnaire.

There are 26 latent variables relevant to our study. Variables $\xi_{1..19}$ describe exogenous variables and $\eta_{1..7}$ describe endogenous variables. Each latent variable is measured through one or more manifest variables. Extending the structure model from Fig. 1 with the manifest variables produces the complete structural equation model evaluated in this study. Note that USM reduces the requirements for the scale levels of data thus allowing the use of categorical variables in addition to metric variables (Weiber and Muhlhaus 2021).

All latent variables related to *MTL capabilities* ($\xi_{1..9}$) are associated with a single manifest variable $x_{1..9}$, which measures how frequently the participants utilized the MTL capabilities in their transformations. This measurement is represented as a ratio ranging from 0% to 100%. The higher the value of $x_{1..9}$, the more frequently the participants used the MTL capabilities in their transformations. Similarly, latent variables related to *MTL properties* ($\eta_{1..7}$) are associated with a single manifest variable $y_{1..7}$ which measures the perceived quality of the property on a 5-point likert scale (e.g., very good, good, neither good nor bad, bad, very bad). For USM analysis the likert scale answers are mapped to numbers from 1 to 5 for each manifest variable.

The use of single-item scales is a debated topic. We justify their usage for the described latent variables on multiple grounds. First, the latent variables are of high complexity due to the abstract concepts they represent. Second, our study aims to produce first results that need to be investigated in more detail in follow up studies, more focused on single aspects of the model. And third, due to the size of our structural equation model multi-item scales for all latent variables would increase the size of the survey, potentially putting off many subjects. The validity of these deliberations for using single-item scales is supported by Fuchs and Diamantopoulos (2009).

The latent variable *language choice* (ξ_{10}) is measured by means of querying participants to list their 5 most recently used transformation languages. In our registered report we planned to also request participants to give an estimate on the percentage of their respective use % (x_{10}). This was discarded during pilot testing as it was seen as unnecessarily prolonging the questionnaire. Pilot testers had difficulties providing accurate data and questioned whether this data was actually used in analysis.

Language skills (ξ_{11}) is measured through x_{11} and x_{12} for which participants are asked to give the amount of years they have been using each language (x_{11}) and the amount of hours they use the language per month (x_{12}). The relation between the latent variable and the manifest variables is grounded in the assumption that using a language for a longer time and more frequently increases one's skills in it.

Similarly, *experience* (ξ_{12}) is associated with the amount of years subjects have been involved in defining model transformations (x_{13}) and the amount of hours they spend on developing transformations each month (x_{14}).

Meta-model size (ξ_{13}) and *model size* (ξ_{14}) both require participants to state the range between which their (meta-) models vary (x_{15}, x_{16}). This is measured by offering participants a number of ranges of (meta-) model objects. For each range participants should give an estimate on how much percent of the (meta-) models they work fall within that size range. For models the ranges are: #objects $\leq$ 10, 10 $\leq$ #objects $\leq$ 100, 100 $\leq$ #objects $\leq$ 1000, 1000 $\leq$ #objects $\leq$ 10000, 10000 $\leq$ #objects $\leq$ 100000, 100000 $\leq$ #objects. For meta-model the ranges are: #objects $\leq$ 10, 10 $\leq$ #objects $\leq$ 20, 20 $\leq$ #objects $\leq$ 50, 50 $\leq$ #objects $\leq$ 100, 100 $\leq$ #objects $\leq$ 1000, 1000 $\leq$ #objects. Similarly, *Transformation size* (ξ_{15}) is measured on a range of lines of code (x_{17}). The options being: LOC $\leq$ 100, 100 $\leq$ LOC $\leq$ 500, 500 $\leq$ LOC $\leq$ 1000, 1000 $\leq$ LOC $\leq$ 5000, 5000 $\leq$ LOC $\leq$ 10000, 10000 $\leq$ LOC. Querying size data in this manner and the associated ranges have been successfully applied in a prior work the authors were involved in Groner et al. (2021). For USM analysis each range is assigned a number from 1 to 6 to represent them.

To formulate the *semantic gap between input and output* (ξ_{16}) we elicit the similarity of the structure (x_{18}) and data types (x_{19}) on a 5-point likert scale (very similar, similar, neither similar nor dissimilar, dissimilar, very dissimilar). We chose to measure the semantic gap this way, because from our experience the structure and involved data types are the main factors that make models look similar. Participants are asked to give the percentage of all their meta-models that fall within each of the five assessments.

The *meta-model sanity* (ξ_{17}) is measured through means of how well participants perceive their structure (x_{20}) and their documentation (x_{21}) to be on a 5-point scale (very well, well, neither well nor bad, bad, very bad). Based on our experience, these are the two main contributing factors to how easily people can understand a meta-model and work with it effectively. Participants are asked to give the percentage of all their meta-models that fall within each of the five assessments.

Lastly, for both *bidirectional uses* (ξ_{18}) and *incremental uses* (ξ_{19}) we query participants on the ratio of bidirectional (x_{22}) and incremental (x_{23}) transformations compared to simple uni-directional transformations they have written.

3.1.2 Pilot Study

We pilot tested the study with three researchers from the institute. All pilot testers are researchers in the field of model driven engineering with more than 5 years of experience. Based on their feedback, we reworded some questions, removed the usage percentage part of the question for *language choice* and added more precise descriptions of the queried concepts. We then made the questionnaire publicly available and distributed a link to it via emails.

3.1.3 Target Subjects & Distribution

The target subjects are both researchers and professionals from industry that have used dedicated model transformation languages to develop model transformations in the last five years. We use voluntary, judgment and convenience sampling to select our study participants. Both authors reached out to researchers and professionals they knew personally via mail and request them to fill out the online survey. We further reach out, via mail, to all authors of publications listed in *ACM Digital Library*, *IEEE Xplore*, *Springer Link* and *Web of Science*

that contain the key word *model transformation* from the last five years. One author used the search engines provided by each of the publication repositories to find such publications. The abstracts of all resulting publications were then used to decide whether authors should be contacted. A third source of subjects is drawn from social media. The authors use their available social media channels to recruit further subjects by posting about the online-survey on the platforms. The social media platform used for distribution was MDE-Net,[4] a community platform dedicated to model driven engineering.

The sampling method differs from the intended method by not including snowballing sampling as a secondary sampling method. We decided on this to have more control over the subjects receiving a link to the study. There is no way to control how closely secondary and tertiary contacts are related to the subject matter of model transformations and MTLs. This could introduce an additional threat to the validity of the answers regarding the representativeness of survey participants.

Participation was voluntary, and we did not incentivise participation through offering rewards. This decision is rooted in our experience in previous studies, one other survey with 83 subjects (Groner et al. 2021), and the interview study we are basing this study on with 56 subjects (Höppner et al. 2022). Both studies reached a satisfactory number of participants without resorting to participation incentives.

It is suggested in literature to have between 5 to 10 times as many participants as the largest number of parameters to be estimated in each structural equation (i.e., the largest number of incoming paths for a latent model variable) (Buckler and Hennig-Thurau 2008). Thus, the minimal number of subjects for our study to achieve stable results is 80. To gain any meaningful results a sample size of 30 must not be undercut (Buckler and Hennig-Thurau 2008).

In total we directly contacted 2383 potential participants. Of these, 99 were contacted because of personal contacts of the authors, while the remaining 2284 were contacted as a result of the literature search. How many additional potential participants were reached through the social media posts can not be determined. The survey got a total of 113[5] responses. It surpasses the minimum threshold of 80, ensuring stable results, suggested in literature.

3.2 Data Analysis

We use USM to examine the hypotheses system modelled by the structure model shown in Fig. 1. USM is chosen over its structural equation modelling alternatives due to it being able to better handle uncertainty about the completeness of the hypothesis system under investigation, it having more capabilities to analyse moderation effects and the ability to investigate non linear correlations (Weiber and Muhlhaus 2021).

USM requires a declaration of an initial likelihood of an interdependence between two variables. This is used as a starting point for calculating influence weights but can change over the course of calculation. For this, Buckler and Hennig-Thurau (2008) suggest to only assign a value of 0 to those relationships that are known to be wrong. We use the results of our interview study (Höppner et al. 2022), shown in the structure model, to assign these values. For each path that is present in the model, we assume a likelihood of 100%. To check for interdependencies that might have been missed by interview participants, we also use a likelihood of 100% for all missing paths between $\xi_{1..19}$ and $\eta_{1..7}$. Our plan was to use a

[4] https://mde-network.com/

[5] This constitutes a response rate of max. 4.8%. We do not know how many responses are a result of our social media posting.

likelihood of 50% for these interdependencies, to provide the tool with some hints to the possibility of these paths not existing, but the tool available to us only allowed for either 100% or 0% to be put as input.

The tool NEUSREL is used on the extracted empirical data and the described additional input to estimate path weights and moderation weights within the extended structure model, i.e., the structure model where each exogenous latent variable is connected to all endogenous latent variables. It also runs significance tests via a bootstrapping routine (Buckler and Hennig-Thurau 2008; Mooney et al. 1993) and produces the significance value estimates for each influence. The following procedures are then followed to answer the research questions from Sect. 1.

RQ1. We reject all hypothesised influences, i.e., those present in our structure model in Fig. 1, that do not pass the statistical significance test. The threshold we set for this is 0.01, deviating from the standard practice of 0.05 because we want greater confidence in the results. Moreover, we discard hypothesised influences with minimal effects strengths that are several (two or more) magnitudes lower than the median influence of all coefficients. If, for example, the median of all path coefficients is 0.03 all influences with a coefficient lower or equal to 0.0009 are discarded. We do so because such low influences suggest that the influence is negligible.

RQ2 & RQ3. All path coefficients produced that were not rejected in **RQ1** will then provide direct values for the influence and moderation strengths to answer **RQ2** & **RQ3**.

RQ4. The same significance criteria we applied to all hypothesised influences for **RQ1**, we also apply to the extended influences, i.e., those not present in the structure model from Fig. 1. Those influences that pass the significance test are added to the initial structural model as newly discovered influences.

3.3 Privacy and Ethical Concerns

All participants were informed of the data collection procedure, handling of the data and their rights, prior to filling out the questionnaire. Participation was completely voluntary and not incentivised through rewards.

During selection of potential participants the following data was collected and processed.

- First & last name.
- E-Mail address.

This data was deleted once the potential participants were contacted.

The questionnaire did not collect any sensitive or identifiable data.

All data collected during the study was not shared with any person outside of the group of authors.

The complete information and consent form can be found in Appendix C.

The study design was not presented to an ethical board. The basis for this decision are the rules of the German Research Foundation (DFG) on when to use a ethical board in humanities and social sciences.[6] We refer to these guidelines because there are none specifically for software engineering research and humanities and social sciences are the closest related branch of science for our research.

[6] https://www.dfg.de/foerderung/faq/geistes_sozialwissenschaften/

4 Demographics

We detail the background and experience of the participants in our study in the following sections.

To illustrate distributions of participants answers to questions where they were asked to estimate how many of their use cases fall into which category, we use Ridgeline Plots (Wilke 2019) (Figs. 7, 8, 9, 10, 11, and 12). Each line in the plot illustrates the distribution of responses from an individual participant for a specific category, and the peaks in each line highlight areas where they provided higher estimates.

For example, the first ridge line at the bottom of Fig. 7 shows the answers of a participant who has stated that 100% of their transformations revolve around meta-models with 10 or less meta-model elements.

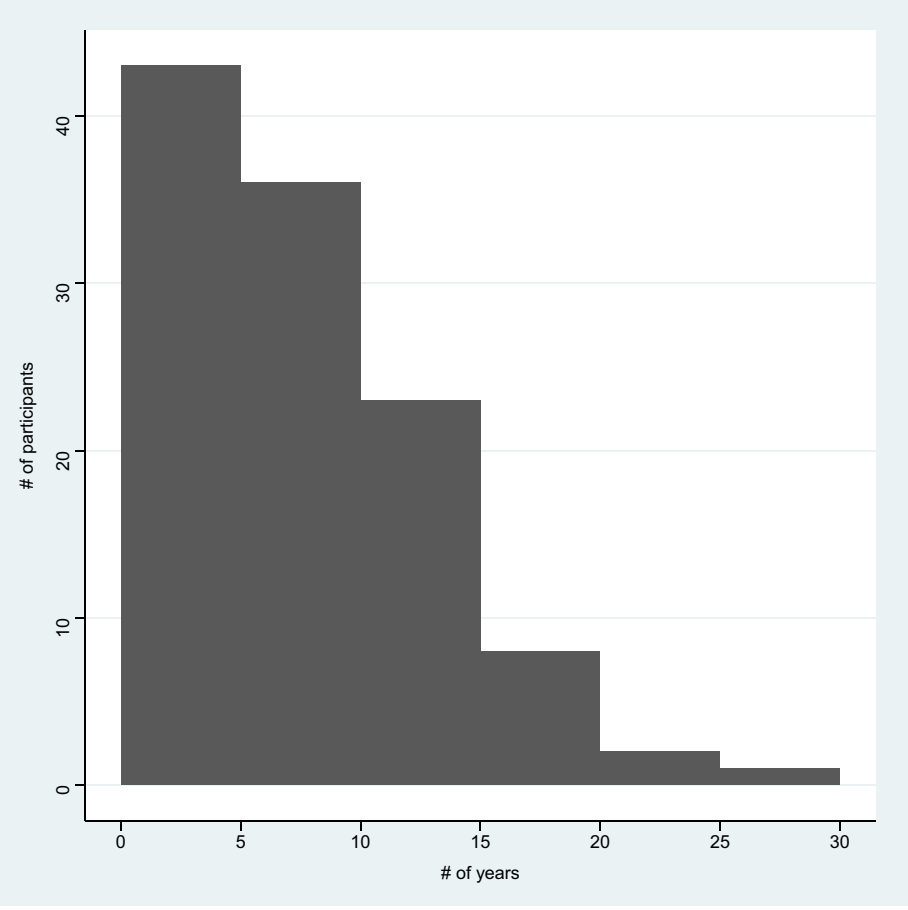

Fig. 5 Histogram of participants' total experience in years

4.1 Experience in Developing Model Transformations (ξ_{12})

Our survey captured model transformation developers with wide range of experience. The experience span (x_{13}) ranges from the least experience participant with half a year of experience up to the one with most experience of 30 years. Figure 5 shows a histogram of the experience stated by participants. Over half of all participants have between 1 to ten years of experience in writing model transformations. Three stated to have more than 20 years in total. On average our participants have 9 years of experience with a median of 8 years of experience.

How much time participants spend developing transformations each month (x_{14}) also greatly varies. Some participants have not developed transformations in recent time whereas others stated to spend 70 or more hours each month on transformation development. Figure 6 shows an overview over the hours participants spend each month in developing transforma-

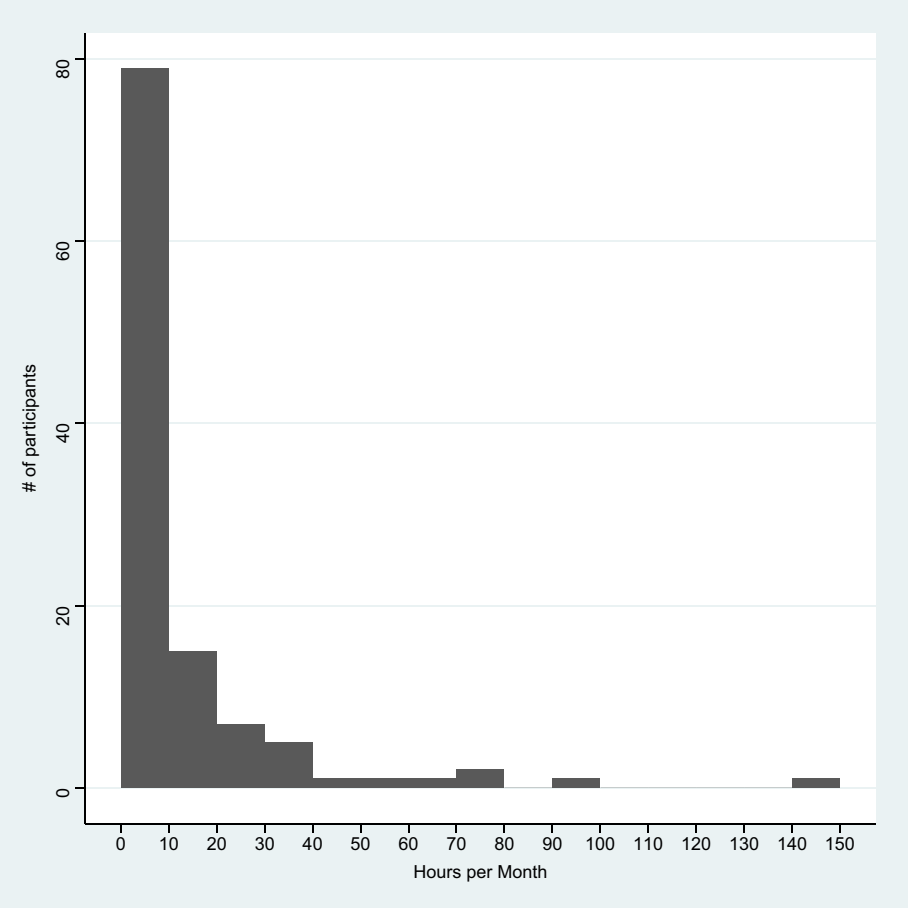

Fig. 6 Histogram of participants recent experience in hours per month

Table 2 Overview of languages used by participants

Language	# number of participants
Java	70
ATL	58
Xtend	52
ETL	29
QVTo	22
Henshin	14
JavaScript	12
eMoflon	7
Fujaba	5
Python	4

tions. The vast majority spends around 1 to 10 h each month on transformation development. Nine stated that they did not develop any transformation in recent times. Their responses are still included in the analysis because they all had prior experience with model transformations. Their experiences just were more than one year ago. On average our participants spend about 14 h per month developing model transformations.

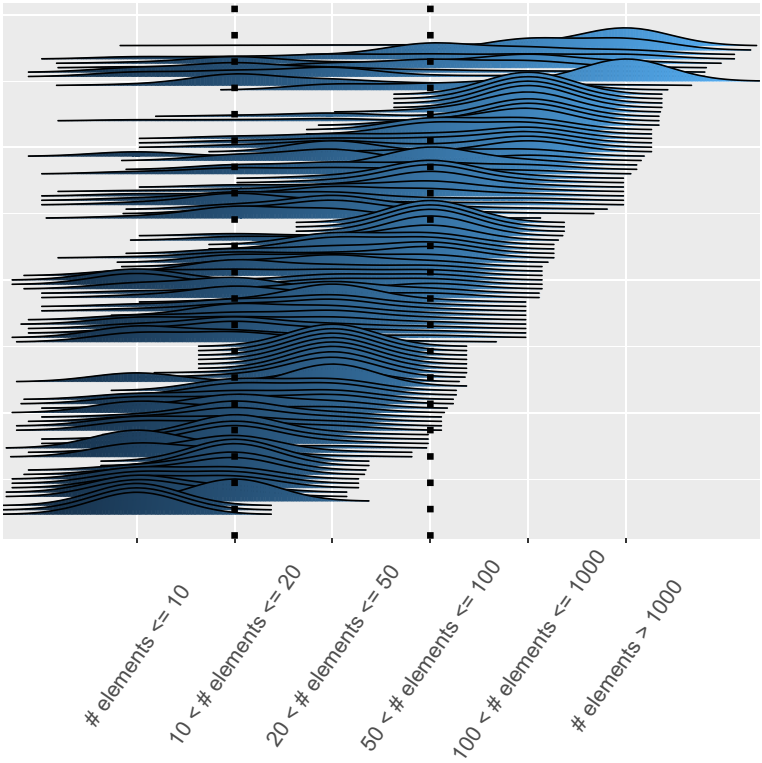

Fig. 7 Distribution of meta-model sizes per participant

4.2 Languages Used for Developing Model Transformations (ξ_{10}) and Experience Therein (ξ_{11})

To develop their transformations, participants use a wide array of languages. In total 43 languages (x_{10}) have been named 24 of which are unique languages used only by a single participant.

The language that has been used by the most participants is Java, a general purpose language. Java has been used by 70 of the 113 participants. The most used MTL is ATL with 58 users closely followed by another GPL, namely Xtend with 52 users. Table 2 shows how many participants use one of the ten most used languages for developing transformations.

Overall the prevalence of general purpose programming languages is higher than expected. This might be explained by the large number of existing MTLs which reduce the amount of total users per language while only four different GPLs are used.

4.3 Sizes (ξ_{12},ξ_{14})

The size distribution of meta-models (x_{15}) transformed by participants is shown in Fig. 7. On the x-axis the given intervals of meta-model sizes are shown and on the y-axis the distribution for each participant is shown.

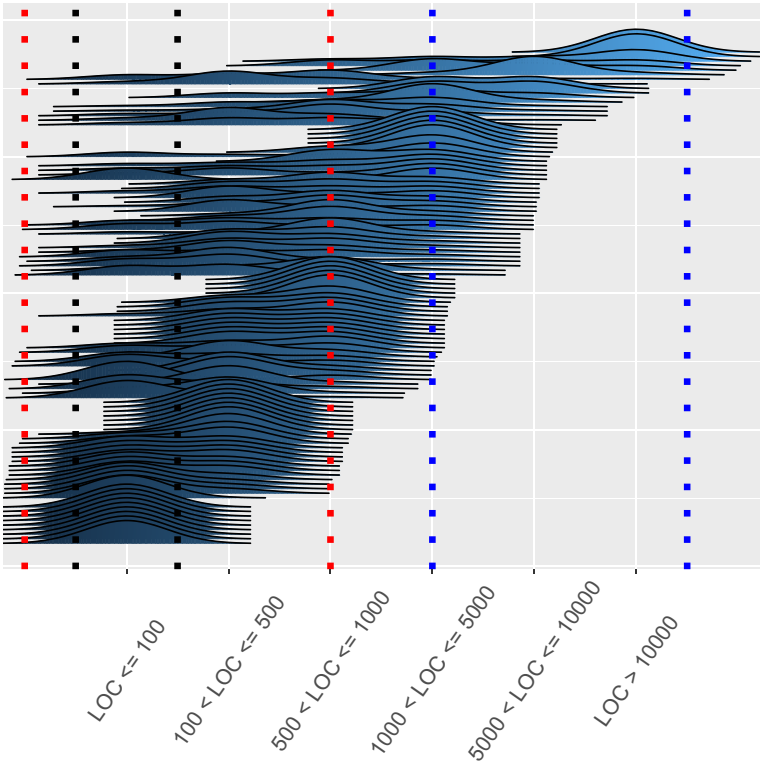

Fig. 8 Distribution of transformation sizes per participant

The figure illustrates that most transformations involve meta-models with 20 to 100 meta-model elements. Moreover, most participants have some experience with small meta-models while only a handful of them has experience with transformations involving large meta models of more than 1.000 elements.

The size distribution of model transformations (x_{17}) written by participants is shown in Fig. 8. Similarly to the meta-model sizes, the figure illustrates that most participants have some experience with small transformations of sizes up to 100 lines of code. Most also have experience with large transformations up to 1.000 lines of code. More than 25% of all participants also have experience with large and very large transformations ranging from 5.000 up to more than 10.000 lines of transformation code.

Overall the experience of our participants includes many moderately large to large transformations. Thus, the experiences of the participants closely align with those one would expect in real-world cases. This strengthens our confidence that our results are reliable.

4.4 Conceptual Distance Between Meta-models (ξ_{16})

The similarity distribution of meta-models involved in the transformations of our participants is shown in Fig. 9 for the similarity of meta-model structures (x_{18}) and Fig. 10 for the similarity of data types (x_{19}). Both show a even mix between structurally similar and distant

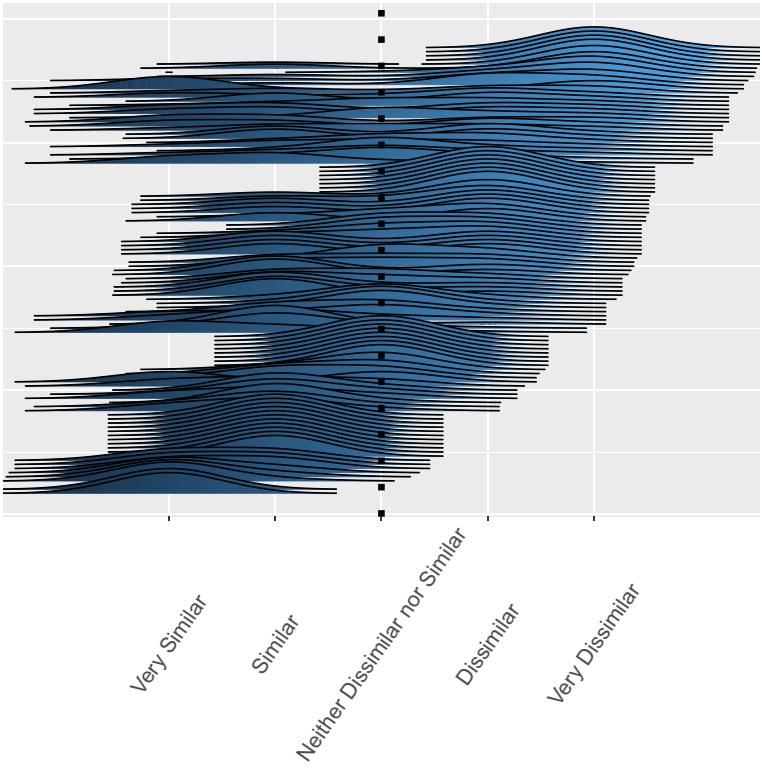

Fig. 9 Distribution of input output meta-model structure similarity

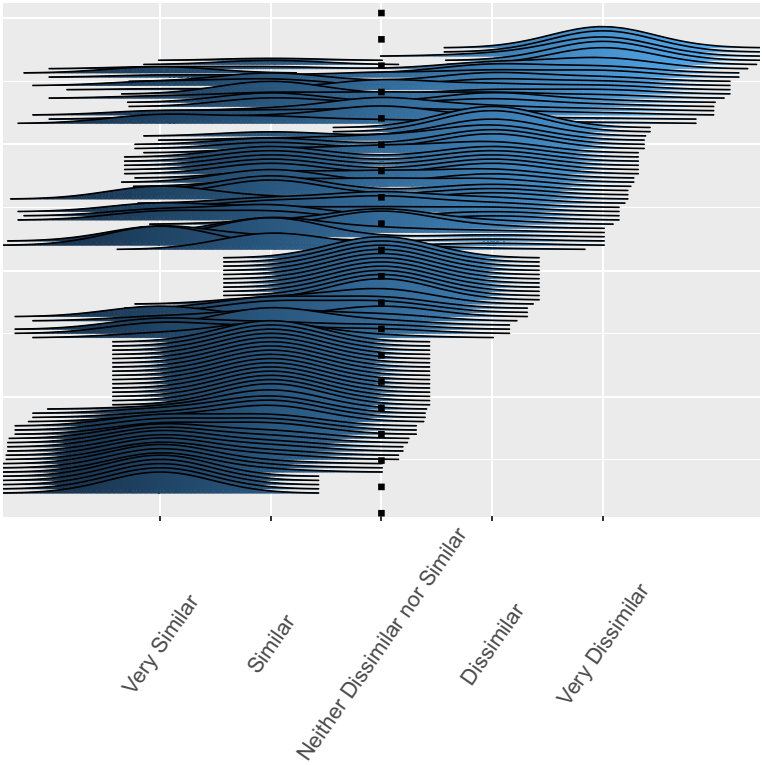

Fig. 10 Distribution of input output meta-model attribute types similarity

meta-models as well as similar and dissimilar attribute types within the elements that are transformed into each other.

4.5 Meta-model Quality (ξ_{17})

Participants agreed that the vast majority of meta-models they transform are well structured (x_{20}). This means there is little to no additional burden put onto development solely due to unfavourably structured meta-models. The distribution of structure assessment per participant is shown in Fig. 11.

The situation is different with documentation (x_{21}). Most participants stated that they have experience with badly or even very badly documented meta-models (Fig. 12). For many participants, this constitutes the majority of meta-models they work with.

5 Results

In this section, we present the results of our analysis of the questionnaire responses using universal structure modelling structured around the research questions **RQ1-4**. The quantitative results for all influences between MTL capabilities and MTL properties are shown in Table 3 in Appendix A. On the x-axis the different *MTL Properties* are shown. On the

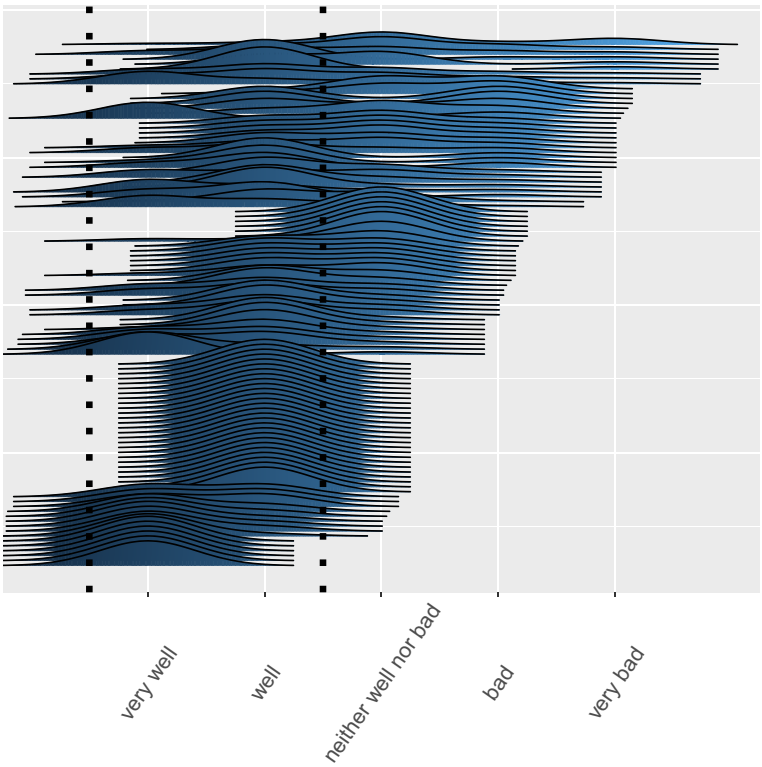

Fig. 11 Distribution of structure quality of meta-models per participant

y-axis the *MTL Capabilities* are shown. The first number in a cell describes the average simulated effect. The second number describes the overall explained absolute deviation. The third number shows the significance value. A significance value lower or equal to 0.01* (the chosen significance level) is indicated with one asterisks. The effect strengths of moderation effects can be found in Tables 4, 5, 6, 7, 8, 9, 10, 11, 12, and 13 in Appendix A. Each table describes the moderation effect of one of the moderating factors on all influences between MTL Capabilities and MTL Properties.

The rest of this section presents our results in context of the four research questions. We focus on the most salient influences that we deem interesting for the respective research question. Detailed interpretation and discussion of the implications of the presented results are done in Sect. 6.

5.1 RQ1: Which of the Hypothesised Interdependencies Withstands a Test of Significance? & RQ4: What Additional Interdependencies Arise from the Analysis that were not Initially Hypothesised?

Our first research question is aimed at evaluating the accuracy of the structure model developed in the preceding interview study (Höppner et al. 2022). We do so by subjecting all hypothesised influences to a significance test during analysis. The significance test can also be used to directly gain insights into interdependencies missed in the initial model. Thus we

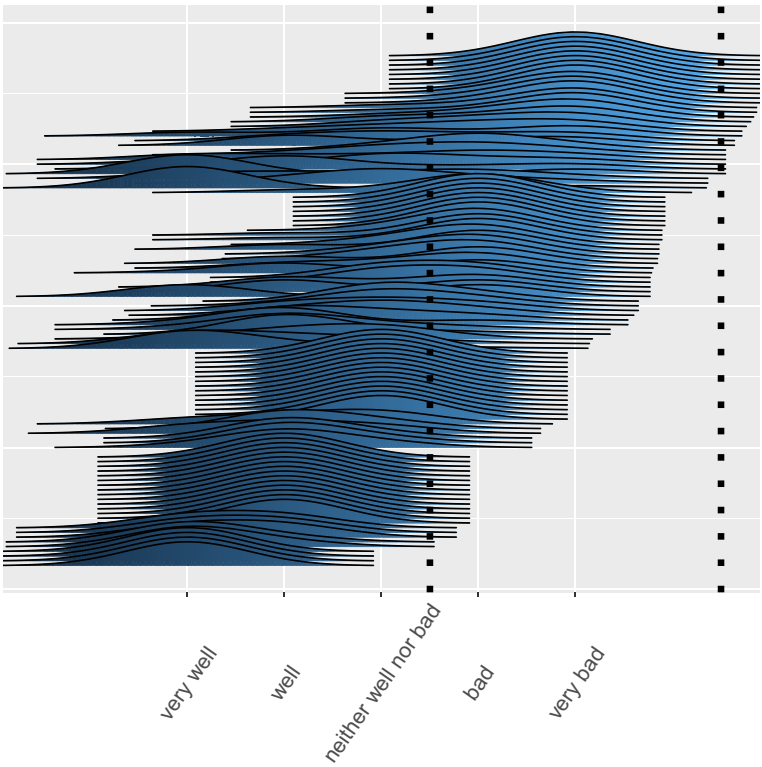

Fig. 12 Distribution of meta-model documentation quality per participant

discuss both the rejection of previously hypothesised influences as well as the extension of the model through newly discovered significant interdependencies in this section.

Most initially hypothesised influences withstand the test of significance but there are several exceptions. Most notably all but one(*Maintainability*) of the hypothesised influences of *Bidirectionality* functionality of MTLs have to be rejected. This means that from our results we can not conclude that the presence of *Bidirectionality* functionality in a language changes how people perceive the *Comprehensibility*, *Ease of Writing Expressiveness*, *Tool Support*, *Productivity* and *Reusability* of the language.

Similarly, half of the influences on *Ease of Writing* and *Expressiveness* are also rejected. This means that the presence of *Bidirectionality*, *Incrementality*, *Model Management* and *Model Traversal* functionality do not change how people perceive the *Ease of Writing* transformations with a language. And that the presence of *Bidirectionality*, *Incrementality*, *Model Navigation*, *Model Traversal* and *Reuse* functionality do not change how people perceive the *Expressiveness* of a language.

The hypothesised influences on *Comprehensibility* are confirmed (apart from the one exerted by *Bidirectionality*). The same goes for *Productivity* and *Reusability*.

We also found that the perceived quality in *Tool Support* and *Maintainability* are influenced by most of the *MTL Properties*. A result that was not apparent from previous interview study. *Tool support* was hypothesised to be influenced only by the chosen language and *Maintainability* only by *Bidirectionality* and *Mapping* functionality. Our results however

show, that the perceived *Maintainability* of transformations written in a language is influenced by all MTL functionality considered in this study with the exception of *Incrementality*.

Moreover, several additional influences on *Productivity* and *Reusability* were also discovered. The perceived *Productivity* and *Reusability* of transformations in a language are influenced by *Mappings*, *Model Management*, *Model Navigation*, *Model Traversal*, *Pattern Matching*, *Reuse Mechanisms* and *Tracability* functionality.

Regarding the moderating effects, our findings suggest that they need to be considered separately for each influence on a quality attribute. The hypothesis that context moderates all influences on an MTL Property still holds but the strength of the moderation effects varies greatly.

As hypothesised, we are able to observe that *Comprehensibility* and *Ease of Writing* are the two properties moderated by the most context variables. But the moderation is only significant for a hand full of influences on these properties. This can be seen e.g. in the moderation effects of *Meta-Model Size* on influences on *Comprehensibility* depicted in Table 5 in Appendix A. Changes in the *Meta-model sizes* participants worked with had next to no effect on how their usage of *Bidirectionality* functionality affected their view on the *Comprehensibility* of transformations. The impact on the influence of *Model Management* on *Comprehensibility* is orders of magnitudes higher.

Another observation that stands out is the impact of *Language Choice* and *Language Experience*. The moderation effects of both variables are negligible or even 0 for all influences. We believe this is due to the large number of languages considered in this study. It makes analysing the effects of choosing one of the languages difficult.

Overall the results for research questions **RQ1 & 4** suggest that our initial structure model contains many relevant interdependencies but several more have to be considered as well. We do have to reject several direct influences due to low significance and moderation effects have to be considered on a per influence basis instead of being generalised for each *MTL Property*.

5.2 RQ2: How Strong are the Influences of Model Transformation Language Capabilities on the Properties Thereof?

Our second research question is intended to provide numbers that can help to identify the most important factors to consider when evaluating the advantages and disadvantages of model transformation languages empirically. We do this by considering both the average simulated effect of influences calculated by NEUSREL as well as the overall explained absolute deviation of influences compared to each other. As explained earlier in this section all numbers can be found in Table 3.

Overall the effects identified in our analysis are lower than anticipated. They range from 0.29 down till 6.5e−8. It is possible, that this stems from the large number of variables that are involved and the overall complexity of the matter under investigation. Nonetheless there are meaningful insights that can be drawn when comparing the influences for each *MTL Property* with each other.

Of the influences hypothesised from the preceding interview study (Höppner et al. 2022) *Traceability* is the most impactful *MTL Capability*. Its usage exerts the highest influence on perceived *Comprehensibility* with 0.29. Similarly it has the highest influence for *Ease of Writing* though with a value of 0.0021 the effect is small. We were, however, already able to show empirical evidence that MTLs utilising automatic trace handling provide clear advantages for writing transformations compared to GPLs (Höppner et al. 2021).

For the properties *Tool Support*, *Maintainability* and *Productivity* the availability of *Reuse Mechanisms* seems to be the strongest driving factor with an average simulated effect of 0.1, 0.1, 0.1 and 0.2, respectively. No other factor has an ASE or effect strength as high as *Reuse Mechanisms* for these properties. This result is significant as the influences were not raised even once during our interview study.

Overall, automatic tracing and reuse mechanisms appear to be the most influential factors for MTL properties. This suggests to us two main pathways for further research. First, to improve model transformation languages more research should be devoted to developing effective ways to reuse transformations or parts of transformations. Current usage of reuse mechanisms is limited, as shown by the available techniques identified by Chechik et al. (2016). From our experience, current mechanism are hard to use and are especially unsuited for different use-cases. Secondly, the first area to address for improved adoption of model transformation concepts in general purpose languages should be the development of mechanisms for automatic trace handling.

5.3 RQ3: How Strong are Moderation Effects Expressed by the Contextual Factors *Use-case*, *Skills & Experience* and *MTL Choice*?

As expressed in Sect. 5.1 the results of our analysis suggest that moderation effects need to be considered separately for each influence on a quality attribute. In this section we go into detail on these nuances.

As hypothesised the size of meta-models moderates the influences on *Comprehensibility*. The moderation strength differs greatly between the different causing factors though. For example, *Meta-model size* exerts the strongest moderation on the influence of *Model Management* onto *Comprehensibility* with 0.14. All other moderation effects are far lower. The second highest moderation effect, the moderation of *Meta-model size* on the influence of *Traceability* on *Comprehensibility*, is about half es strong (0.0778) and the lowest, the moderation of *Meta-model size* on the influence of *Bidirectionality* functionality on *Comprehensibility*, is only 0.0009. The moderations make sense intuitively as larger meta-models would make implementing these tasks manually more labour intensive and thus clutter the code unnecessarily.

Model size exerts similar moderation effects as meta-model size. Its strongest moderation effect is also on the influence of *Model Management* on *Comprehensibility* (0.36). Moreover, *Model size* also strongly moderates the influence of *Traceability* functionality on the *Ease of Writing* transformations (0.17). Most other moderation effects of *Model size* are far lower than 0.1.

Transformation size seems to be the most relevant moderating factors across the board. It has many noteworthy moderation effects on all influences of *MTL Capabilities* on *Tool Support*, none being less than 0.16, and *Productivity*, most being above 0.12. We assume this is because the larger transformations get, the more reliant developers are on tooling and abstractions that reduce the development effort.

Another interesting effect we found is, that *developer experience* moderates the influence of many of the domain specific abstractions, e.g. *Mappings* and *Model Traversal*, on *Productivity*. This makes sense because these specific features often break with how developers are used to develop programs and thus need practice to use them effectively.

The *semantic gap between input and output* meta-models exerts its moderation strongest on the influences on *Maintainability*. Most notable are the moderations on the influences of *Model Traversal* (0.194), *Pattern Matching* (0.239) and *Reuse Mechanisms* (0.237).

Lastly, there is a strong moderation effect of the *meta-model sanity* onto the influence of *Model Management* facilities and *Bidirectionality* on *Comprehensibility*. Both being about 0.2. This makes sense as badly structured or poorly documented meta-models are harder to handle and thus the tasks revolving around working with the structure are most influenced by that.

Overall, we believe that the scale of transformations is the most important moderating variable. However, the assumption about the relevance of language choice could not be confirmed. This is most likely due to the large amount of languages each participant has had experience with which weakens the ability to elicit the effect of differences of language choice between participants.

5.4 Summary of Results

In this section, we aim to provide a less technical overview of the key findings from our research, along with a brief preview of their implications which we will discuss in detail in Sect. 6.

On the side of direct influences exerted by the capabilities of model transformation languages, we found that both bidirectionality and incrementality functionality had significantly less impact than hypothesised. In fact, most of their influences did not cross the significance threshold. Traceability and reuse mechanisms on the other hand emerged as the most impactful aspects. They affect the highest number of MTL properties and also exhibited the highest average simulated effect and overall explained deviation. This suggests that these are currently the most important properties of model transformation languages.

For researchers, this underscores the importance of focusing on traceability and reuse mechanisms in empirical evaluations of MTLs. For language developers, it highlights the need to allocate resources mainly for the improvement and further innovation of these specific language features. Especially for reuse mechanisms we see a lot of potential for future progress.

On the side of moderating factors, we were unable to determine the impact of language choice and skills. All other investigated moderating factors demonstrated significant effects on all MTL properties. While this was not initially hypothesized, it aligns with our broader observations that the use-case for which MTLs are applied is crucial for their effectiveness. This observation is underscored by the fact that the most potent moderation effects were exerted by the factors surrounding meta-models. Considering the fact that meta-models and their meta-data are the closest data we measure that relate to the use-case, this outcome is somewhat expected.

6 Discussion

The results of our analysis provide useful insights for research on model transformation languages. In this section, we discuss the implications of our results for evaluation and development of MTLs. Additionally, we provide a critical evaluation of our methodology with regards to the goals of this study.

6.1 Implications of Results

The topic of influences on the quality properties of model transformation language is vastly complex, as reflected in the already large structure model that we set out to analyse. While we were able to reject some of the hypothesized influences, our analysis also identified several new influences. As a result, the structure model depicting the influences grew in complexity, further highlighting the need for comprehensive studies of the factors that influence MTL quality properties. The updated structure model can be seen in Fig. 13.

It contains 36 more interdependencies than the one we started our analysis with but also misses some initially hypothesised ones. The difference to our initial structure model are highlighted as follows: Removed interdependencies are coloured red. Newly added interdependencies are coloured blue.

Our analysis produced a number of interesting observations that have important implications for further research. In particular, we now discuss the implications for empirical evaluations. Additionally, we highlight the implications of our results for further development of MTLs and domain-specific features thereof.

6.1.1 Suggestions for Further Empirical Evaluation Studies

The lack of empirical studies on the quality aspects of Model Transformation Languages can be attributed to the inherent complexity of the field and the challenges associated with obtaining suitable datasets. Recognizing this gap, this section aims to offer researchers suggestions for directing their efforts and structuring their datasets strategically. By providing a set of suggestions, we aim to guide researchers in overcoming the challenges posed by the intricate nature of MTLs, facilitating the generation of meaningful and insightful results.

Traceability plays one of the most important roles in the development of model transformations, exerting the strongest influence on the perceived quality of both the ease of writing and the comprehensibility of the resulting code. To assess the value of MTL abstractions for writing and understanding transformations, it is essential to consider scenarios involving tracing. Equally important is the evaluation of situations where tracing is not necessary, to discern the impact of MTL abstractions. Conducting experiments with various transformation cases, encompassing development, maintenance, or understanding, is crucial for focused studies in these areas. To gauge the practical significance of this feature, it is imperative to analyze the prevalence of real-world use cases requiring traceability. Large-scale meta-studies that scrutinize existing transformation problems are well-suited for this purpose. By systematically considering these factors, a comprehensive understanding of the value of MTL abstractions in the context of writing and comprehending transformations can be obtained.

For evaluation of Maintainability, *Reuse Mechanisms* as well as *Model Traversal* functionality are important capabilities to consider. We therefore believe that researchers focusing on such an evaluation must make sure to use transformations that utilise these capabilities. Moreover, the most important context to consider is the semantic gap between input and output meta-models. Empirical evaluations focusing on maintainability should therefore make sure to evaluate transformation cases with varying degrees of differences between input and output meta-models. Because of the complexity of the factors and quality attributes to consider, case studies are most suitable. They can also be set up in cooperation with industry partners to gain insights into real-world application of model transformations. These studies should then analyse how much the effectiveness of MTLs and GPLs changes in light of the semantic gap between input and output. There exist some studies in the area of MDE, such

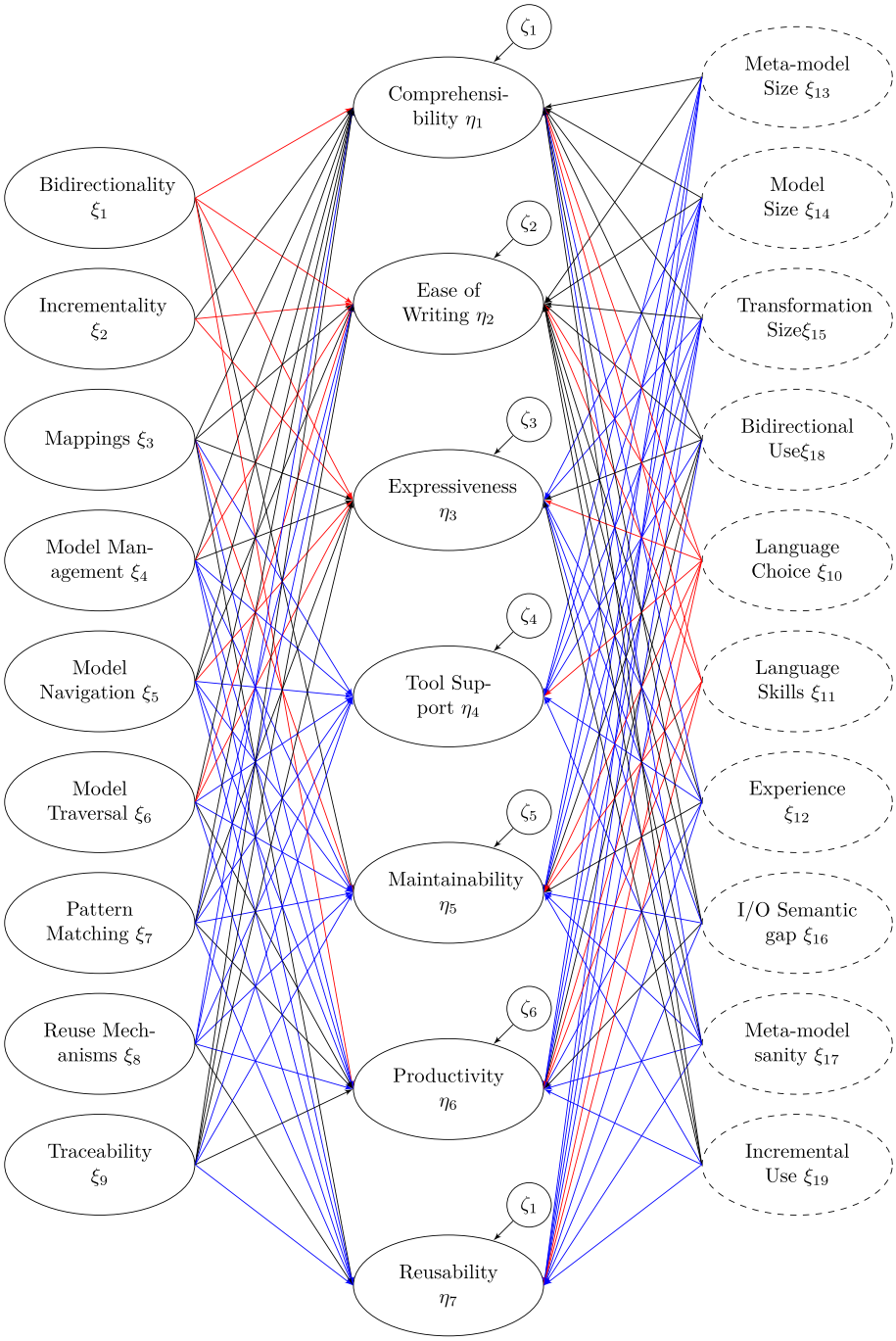

Fig. 13 Structure model depicting the confirmed influence and moderation effects of factors on MTL properties

as the studies by Staron (2006); Mohagheghi et al. (2013a), but these lie far in the past and no new studies have been conducted.

When choosing transformations for evaluation, it is crucial to take their size into account. Our findings indicate that size has the most pronounced impact on the influence of other factors on properties. In other words, the larger the transformation, the more discernible the effect of all capabilities will be. Therefore, it is essential to concentrate on large transformation use-cases when designing a study to evaluate MTLs.

6.1.2 Suggestions on Language Development

The most significant finding of this study is the importance of reuse functionality. Tool support, maintainability, productivity, and reusability, among the quality attributes examined, are predominantly influenced by the effectiveness of reuse functionality. This is especially significant because there was no indication of this in our interviews (Höppner et al. 2022). We suppose this influence stems from the fact that reuse mechanisms allow for more abstraction and thus less code that can be developed and maintained more efficiently.

As a result, more focus should be put on developing transformation specific reuse mechanisms. We are aware that some languages, e.g. ATL, already provide general reuse mechanisms through concepts like inheritance. However, these concepts are limited by the fact that they rely on the object-oriented nature of the involved models. This means that they can only be used to define reusable code within transformations of a single meta-model. Defining transformation behaviour that can be reused between different meta-models is not possible. But this would be important to further reduce redundancy in transformation development.

Lastly, the development of reuse mechanisms tailored to MTs is also important to focus on. In order to stand out compared to the reuse mechanisms of GPLs, it may be valuable to explore ways to define and reuse common transformation patterns independently of meta-models. Higher order transformations are sometimes used to allow reuse too (Kusel et al. 2015), but from our experience current implementations are too cumbersome to be used productively. Chechik et al. (2016) provide a number of suggestions for transformation specific reuse mechanisms but to the best of our knowledge there exist no implementations of their concepts.

6.1.3 Suggestions for Transformation Development

Our results also have direct implication for the decision making process when choosing a language for developing transformations. Here we present a number of key points to consider when making that decision.

Are there special requirements like bidirectionality or incrementality? Our results point toward higher impact of bidirectional and incremental functionality provided by MTLs if the use-case requires such special case transformations. This can reduce development effort. However, there currently do not exist empirical studies to back up this claim, apart from the data we provide here.

Is the amount of required tracing between input and output high or low? Dedicated support for tracing is one of the main advantages of MTLs. This is shown by the high effect strength exhibited by this property in our results. Thus, the higher the amount of tracing between input and output of the transformation that is required the more advantage developers can derive from using a MTL.

How high is the demand on the reusability of the transformations? As shown by our results, reusability functionality has a high impact on the perception of model transformation

languages. The higher the demand for reusability the more prevalent this effect will be. MTLs currently do provide reusability features solely on a per meta-model basis. If it is anticipated that the transformations should also be reused in different contexts MTLs will not provide ways of abstraction for that. In such cases using GPLs might be favourable.

6.2 Interesting Observations Outside of USM

When discussing model transformation languages, it is often stated that they are only demonstrated on 'toy examples' that have little to no real world value. This argumentation has for example been raised several times in the interview study (Höppner et al. 2022). However, the demographic data collected in our study disputes this.

There are several participants that stated to have worked solely on small transformations with small meta-and input models. But this group is opposed by a similarly large group of participants that have worked with huge transformations, dissimilar and large meta-models as well as large inputs. From this we conclude, that there are large use-cases where model transformations and MTLs are applied but they rarely get described in publications. It seems likely that such examples are not used for highlighting important aspects authors want to discuss due to the space describing such cases would take up. However, we argue that it is paramount that such case-studies are published to diminish the preconception that MTLs are only useful for small examples.

Another noteworthy observation based on the demographic data of our participants is that documentation pertaining meta-models is predominantly perceived as inadequate. We believe that this is primarily due to the fact that many of meta-models stem from research projects that prioritize expeditious prototyping over the long-term viability of the artefacts. Nonetheless, we are convinced that there is an urgent need to enhance the documentation surrounding model transformations. This issue is not limited solely to the meta-models, but also extends to the languages that are known for their challenging learning curve because of lack of tutorials (Höppner et al. 2022).

6.3 Critical Assessment of the Used Methodology

The appeal of using structural equation modelling for analysing the responses to our survey was to have a method of analysis that can be used to investigate a complex hypothesis system in its entirety. Moreover, analysis is straight forward after an initial setup due to the sophisticated tooling for this methodology. Instead of presenting participants with a case that they should assess we also opted for querying them on their overall assessment of MTL quality attributes. These design decisions have implications and ramifications that we discuss in this section.

First, the effects observed in our study are small. We assume this stems from the intricate and large structure model and the comparatively small sample size. As explained in Sect. 3 it is suggested to have between 5 to 10 times as many participants as the largest number of parameters to be estimated in each structural equation. In light of the newly discovered paths in our structure model, the 113 total participants are close to the minimum sample size required. Moreover, because of the large number of influences we do expect the influence of a single factor to be much smaller than in structure models where only 2-3 factors are relevant. The results therefore reinforce our assessment that it is a very complex topic.

We also ran into some difficulties when using NEUSREL to analyse our data. The structure model was so large that sometimes the tool crashed during calculations. The online tooling

to set everything up was also painfully inefficient leading to more problems during setup like browser crashes. It took us some trial and error to find a way to get everything set up and run the analysis without crashes.

We chose to execute a study based on our study design in hopes of producing a complete theory independent of the use case under consideration. The results of the study complement other studies on the topic that used a more narrow scope, investigating effects under one or several specific use-cases. Hebig et al. (2018), for example, investigate the effects of using model transformation languages for comprehension and ease of writing in a controlled experiment. During discussion of their results, the authors have to concede that their findings are made under narrow conditions and thus can not be generalised to other use-cases. Similar to our questionnaire setup, Groner et al. (2021) also use a questionnaire focused on the general assessment of participants about performance engineering in model transformation development and are able to produce statistically significant results. Nonetheless, additional studies need to be conducted to confirm our results for different use-cases.

7 Threats to Validity

Our study is carefully designed and follows standard procedures for this type of study. There are, however still threats to validity that stem from design decisions and limitations. In this section we discuss these threats.

7.1 Internal Validity

Internal validity is threatened by manual errors and biases of the involved researchers throughout the process.

The two activities where such errors and biases can be introduced are the subject selection and question creation. The selection criteria for study subjects is designed in such a way, that no ambiguities exist during selection. This prevents researcher bias.

The survey questions and answers to the questions pose another threat to internal validity. We used neutral questions to prevent subconsciously influencing the opinions of research subjects. We also provide explanations for ambiguous terms used in the survey. However, there are several instances where we can not fully ensure that each participant interprets terms the same way. The questions on quality properties of model transformation languages allow room for interpretation in that we do not provide a clear metric what terms such as ‘Very Comprehensible’ or ‘Very Hard to write’ mean. Similarly, the questions on meta-model quality leave room for interpretation on the side of participants. We opted for this limitation because there are no universal ways to quantify such estimates and because the subjective assessment is what we want to collect. The reason for this is, that subjective experiences are the main driving factor for all discussions on development when people are the main subject.

To ensure overall understandability and prevent errors in the setup of the survey we used a pilot study.

7.2 External Validity

External validity is threatened by our subject sampling strategy and the limitations on the survey questions imposed by the complexity of the subject matter.

We utilise judgement, voluntary and convenience sampling. As a result, the representativeness of the participants for the target population is hampered. Convenience sampling can limit how representative the final group of interviewees is. Since we do not know the target populations makeup, it is difficult to asses the extend of this problem.

Moreover, using research articles as a starting point introduces a bias towards researchers. There is little potential to mitigate this problem during the study design, because there exists no systematic way to find industry users.

Due to the complexity and abstractness of the concepts under investigation, a measurement via reflective of formative indicators is not possible. Instead we use single item questions. We further assume that positive and negative effects of a feature are more prominent if the feature is used more frequently. This can have a negative effect on the external validity of our results. We consciously decided for these limitations to be able to create a study that concerns itself with all factors and influences at once.

7.3 Construct Validity

Construct validity is threatened by inappropriate methods used for the study.

Using the results of online surveys as input for structural equation modelling techniques is common practice in market research (Weiber and Muhlhaus 2021). It is less common in computer science. However, we argue that for the purpose of our study it is an appropriate methodology. Structural equation modelling is designed to analyse large hypothesis systems to quantify the interdependencies between variables (Weiber and Muhlhaus 2021). As such, it suits the goal of our study to extract influence strengths and moderation effects of factors on different properties.

Furthermore, due to tool limitations we had to add a likelihood of 100% to all interactions, instead of using 100% for those hypothesised in the interviews and 50% for those that were not hypothesised. This increases the initial weight that is put onto those influences during analysis. This has no impact on the complexity of the hypothesis system analysed, as we had not planned to exclude any influences completely. Thus, it has no impact on our results, but only on the tool's internal analysis process, which may take longer to reach its termination condition.

Lastly, no single language of the languages our participants have used boasts all MTL capabilities. As a result, depending on which languages participants have used, they may not have experience with all MTL capabilities. This reduces the total amount of answers we get for each MTL capability.

7.4 Conclusion Validity

Conclusion validity is mainly threatened by biases of our survey participants.

It is possible that people who do research on model transformation languages or use them for a long time are more likely to see them in a positive light. As such there is the risk that too little experiences will be reported on in our survey. However, this problem did not present itself in a previous study by us on the subject matter (Höppner et al. 2022). In fact researchers were far more critical in dealing with the subject. As a result, there might be a slight positive bias in the survey responses, but we believe this to be negligible.

8 Related Work

There are numerous works that explore the possibilities gained through the usage of MTLs such as automatic parallelisation (Sanchez Cuadrado et al. 2020; Biermann et al. 2010; Benelallam et al. 2015), verification (Lano et al. 2015; Ko et al. 2015) or simply the application of difficult transformations (Anastasakis et al. 2007). There is, however, only a small amount of works trying to evaluate the languages to gain insights into where specific advantages or disadvantages associated with the use of MTLs originate from. Several other works that can be related to our study also exist. The related work is divided into studies focused on the impact of model driven software engineering, studies focused on the investigation of properties of model transformation languages and empirical studies on model transformation languages.

8.1 Studies on Model Driven Software Engineering

There exist several studies that evaluate the usefulness of MDSE as a whole. Staron (2006) used expert interviews from two companies which were planning to adopt MDSE to find out how this is done in industry. Their main findings are that tooling and processes were not ready to base development on models only. Tooling, specifically surrounding MTLs was also criticised.

Whittle, Hutchinson, Rouncefiled et al. used questionnaires and interviews (Whittle et al. 2013; Hutchinson et al. 2011, 2010, 2014) to investigate the positive and negative effects of using MDSE in industry. They also investigated the factors that driver or hinder MDSE adoption in industry. Their findings point towards organisational advantages such as better communication and flexibility when faced with requirement changes as well as gains in productivity. Factors for the success of MDSE are mainly in the areas of organisational tasks like choosing the right use-case to apply MDSE to and committing to see through all required changes within the company. They also point out several open challenges for research such as opening communities to educate people about MDSE and focusing research on the most important aspects surrounding industrial adoption.

Mohagheghi et al. executed empirical studies focused on factors and consequences of adopting MDSE in industry. For this purpose they used both surveys and interviews at several companies (Mohagheghi et al. 2013a, b) as well as a literature review (Mohagheghi and Dehlen 2008). Their results show that MDSE is not suited for small projects because the adoption overhead is too high.

8.2 Studies on the Properties of Model Transformation Languages

Cabot and Gerard (2019) conducted on a online survey and open discussion at the 12th edition of the International Conference on Model Transformations (ICMT'2019). The goal of the survey was to identify reasons why developers decided to use or dismiss MTLs for writing transformations. They also tried to gauge the communities sentiment on the future of model transformation languages. At ICMT'2019, where the results of the survey were presented, they then held an open discussion on this topic and collected the responses of participants. Their results show that MTLs have fallen in popularity. They attribute this to 3 types of issues, technical issues, tooling issues and social issues, as well as the fact that GPLs have assimilated many ideas from MTLs. The results of their study are a major driver in the motivation of our work. While they identified issues and potential avenues for future research, their results are qualitative and broad which we try to improve upon with our study.

In a prior study of ours (Götz et al. 2021), we conducted a structured literature review which forms the basis of much of our work since then. The literature review aimed at extracting and categorising claims about the advantages and disadvantages of model transformation languages as well as the state of empirical evaluation thereof. We searched over 4000 publication for this purpose and extracted 58 that directly claim properties of MTLs. In total 137 claims were found and categorised into 15 quality properties of model transformation languages. The results of the study show that little to no empirical studies to evaluate MTLs exist and that there is a severe lack of context and background information that further hinders their evaluation.

Lastly, there is our interview study (Höppner et al. 2022) the data of which forms the basis for the reported study. We interviewed 56 people on what they believe the most relevant factors are that facilitate or hamper their advantages for different quality properties identified in the prior literature review. The interviews brought forth insights into factors from which the advantages and disadvantages of MTLs originate from as well as suggested a number of moderation effects on the effects of these factors. These results for the data basis for this study.

8.3 Empirical Studies on Model Transformation Languages

To identify differences in transformation development between MTLs and GPLs Tehrani et al. (2016) used an interview study with five participants. They found that all projects where MTLs were used started out as greenfield projects because incorporating MTLs in existing, GPL focused projects is difficult. Furthermore their results point towards a lack of systematic process on how to develop model transformations in general.

Jakumeit et al. (2014) did an in-depth comparison of different MTLs based on the Transformation Tool Contest (TTC). For this purpose they analysed all submissions for the contest in 2011 and summarise how the case was solved using different languages highlighting the differences between them.

Hebig et al. (2018) report on a controlled experiment to evaluate how the use of different languages, namely ATL, QVT-O and Xtend affects the outcome of students solving several transformation tasks. During the study student participants had to complete a series of three model transformation tasks. One task was focused on comprehension, one task focused on modifying an existing transformation and one task required participants to develop a transformation from scratch. The authors compared how the use of ATL, QVTo and Xtend affected the outcome of each of the tasks. Unfortunately their results show no clear evidence of an advantage when using a model transformation language compared to Xtend. However, they concede that the conditions under which the observations are made, were narrow.

We published a study on how much complexity stems from what parts of ATL transformations (Götz and Tichy 2020) and compared these results with data for transformations written in Java (Höppner et al. 2021) to elicit advantageous features in ATL and to explore what use-cases justify the use of a general purpose language over a model transformation language. In the study, the complexity of transformations written in ATL were compared to the same transformations written in Java SE5 and Java SE14 allowing for a comparison and historical perspective. The Java transformations were translated from the ATL transformations using a predefined translation schema. The results show that new language features in Java, like the Streams API, allow for significant improvement over older Java code, the relative amount of complexity aspects that ATL can hide stays the same between the two versions.

Gerpheide et al. (2016) use a mixed method study consisting of expert interviews, a literature review and introspection, to formalize a quality model for the QVTo model transformation standard. The quality model is validated using a survey and used to identify the necessity of quality tool support for developers.

There are two study templates for evaluating model transformation languages that have been proposed but not yet used. Kramer et al. (2016) propose a template for a controlled experiment to evaluate comprehensibility of MTLs. The template envisages using a questionnaire to evaluate the ability of participants to understand what presented transformation code does. The influence of the language used for the transformation should then be measured by comparing the average number of correct answers and average time spent to fill out the questionnaire. Struber and Anjorin (2016) also propose a template for a controlled experiment. The aim of the study is to evaluate the benefits and drawbacks of *rule refinement* and *variability-based rules* for reuse. The quality of reusability is measured through measuring the comprehensibility as well as the changeability collected in bug-fixing and modification tasks.

9 Conclusion

Our study provides the first quantification of the importance of model transformation language capabilities for the perception of quality attributes by developers. It once again highlight the complexity of the subject matter as the effect sizes of the influences are small and the final structure model grew in size.

As demonstrated by the amount of influences contained in the structure model many language capabilities need to be considered when designing empirical studies on MTLs. The results however point towards Traceability and Reuse Mechanisms as the two most important MTL capabilities. Moreover, the size of the transformations provides the strongest moderation effects to many of influences and is thus the most important context factor to consider.

Apart from implications for further empirical studies our results also point a clear picture for further language development. Transformation specific reuse mechanisms should be the main focus as shown by their relevance for many development lifecycle focused quality attributes such as Maintainability and Productivity.

Appendix

A USM Results for Moderation Effects

See Tables 4, 5, 6, 7, 8, 9, 10, 11, 12, and 13.

Table 3 Average simulated effect, overall explained absolute deviation and significance of direct influences

	Comprehensibility	Ease of writing	Expressiveness	Tool support	Maintainability	Productivity	Reusability
Bidirectionality	-3.2e−07/ 3.3e−07/ 1	6.6e−08/ 9.2e−05/ 1	1.5e−09/ 4.2e−06/ 1	-1.5e−05/ 2.0e−05/ 1	−0.06/ 0.05/ 0.1*	-8.3e−06/ 8.9e−07/ 1	1.4e−06/ 5.5e−06/ 1
Incrementality	0.02/ 0.01/ 0.01*	-3.5e−07/ 0.0003/ 1	7.4e−07/ 0.002/ 1	0.0002/ 0.0002/ 1	0.0004/ 1.7e−05/ 1	−0.0001/ 6.5e−07/ 1	0.02/ 0.005/ 0.05
Mappings	0.03/ 0.01/ 0.01*	-1.8e−05/ 0.01/ 0.01*	-6.8e−06/ 0.01/ 0.01*	−0.0116/ 0.01/ 0.01*	−0.000793/ 0.003/ 1	−0.005/ 0.04/ 0.01*	−0.02/ 0.01/ 0.01*
Model management	0.03/ 0.1/ 0.01*	4.4e−05/ 0.003/ 1	0.0006/ 0.006/ 0.01*	0.02/ 0.07/ 0.01*	0.03/ 0.04/ 0.01*	−0.0005/ 0.05/ 0.01*	0.06/ 0.03/ 0.01*
Model navigation	−0.01/ 0.03/ 0.01*	-2.3e−05/ 0.01/ 0.01*	5.9e−05/ 0.005/ 0.05	0.06/ 0.2/ 0.01*	−0.004/ 0.05/ 0.01*	0.05/ 0.07/ 0.01*	−0.08/ 0.08/ 0.01*
Model traversal	0.008/ 0.009/ 0.01*	2.1e−07/ 0.002/ 1	-9.4e−05/ 0.0005/ 1	−0.09/ 0.2/ 0.01*	0.07/ 0.1/ 0.01*	−0.003/ 0.03/ 0.01*	0.007/ 0.01/ 0.01*
Pattern matching	0.05/ 0.07/ 0.01*	-8.4e−05/ 0.01/ 0.01*	−0.0001/ 0.006/ 0.01*	0.05/ 0.06/ 0.01*	−0.04/ 0.08/ 0.01*	0.005/ 0.1/ 0.01*	−0.06/ 0.06/ 0.01*
Reuse mechanisms	0.1/ 0.08/ 0.01*	-7.8e−05/ 0.008/ 0.01*	−0.0002/ 0.002/ 1	0.1/ 0.1/ 0.01*	0.1/ 0.2/ 0.01*	0.1/ 0.2/ 0.01*	0.2/ 0.1/ 0.01*
Traceability	0.29/ 0.12/ 0.01*	0.002/ 0.02/ 0.01*	-2.1e−05/ 0.007/ 0.01*	−0.05/ 0.2/ 0.01*	−0.02/ 0.1/ 0.01*	0.04/ 0.05/ 0.01*	0.08/ 0.09/ 0.01*

Please note that the significance values obtained through the NEUSREL tool may exhibit reduced accuracy compared to standard approaches due to the bootstrapping method used for their estimation

Table 4 Overview of moderation effects of meta-model size

	Comprehensibility	Ease of writing	Expressiveness	Tool support	Maintainability	Productivity	Reusability
Bidirectionality	0.0009	0.1644	0.0004	0.0341	0.0247	0.0197	0.0218
Incrementality	0.0174	0.0003	0.0086	0.0335	0	0.0224	0.0054
Mappings	0.018	0.0309	0.0058	0.0049	0	0.0012	0.0106
Model management	0.1389	0.0264	0.0006	0.023	0	0.0177	0.0199
Model navigation	0.001	0.0438	0.0019	0.0128	0.0032	0.0422	0.018
Model traversal	0.0382	0.059	0	0.0422	0.0023	0.037	0.0106
Pattern matching	0.0091	0	0.0003	0.03	0.0028	0.0418	0.0093
Reuse mechanisms	0.0495	0	0.0077	0.0542	0.0693	0.0943	0.0021
Traceability	0.0778	0.0464	0.00001	0.0714	0.021	0.076	0.0223

Table 5 Overview of moderation effects of model size

	Comprehensibility	Ease of writing	Expressiveness	Tool support	Maintainability	Productivity	Reusability
Bidirectionality	0.1	0.066	0.036	0.048	0.03	0.069	0.012
Incrementality	0.065	0.14	0.019	0.036	0.031	0.052	0.009
Mappings	0.085	0.033	0.076	0.042	0.032	0.083	0.003
Model management	0.36	0.089	0.059	0.023	0.047	0.079	0.04
Model navigation	0.074	0.007	0.04	0.053	0.05	0.072	0.011
Model traversal	0.125	0.069	0.038	0.048	0.031	0.104	0.008
Pattern matching	0.133	0.052	0.038	0.056	0.024	0.036	0.014
Reuse mechanisms	0.096	0.00009	0.029	0.005	0.11	0.093	0.036
Traceability	0.078	0.173	0.02	0.048	0.096	0.108	0.016

Table 6 Overview of moderation effects of transformation size

	Comprehensibility	Ease of writing	Expressiveness	Tool support	Maintainability	Productivity	Reusability
Bidirectionality	0.1	0.15	0	0.27	0.033	0.097	0.044
Incrementality	0.086	0.0075	0.0086	0.33	0.043	0.092	0.050
Mappings	0.13	0.0055	0	0.17	0.042	0.16	0.034
Model management	0.32	0.094	0.021	0.23	0.012	0.13	0.056
Model navigation	0.15	0.058	0.0022	0.25	0.049	0.13	0.064
Model traversal	0.069	0.045	0.0038	0.21	0.061	0.15	0.015
Pattern matching	0.16	0.055	0.040	0.28	0.075	0.12	0.055
Reuse mechanisms	0.21	0.019	0.0077	0.16	0.27	0.23	0.087
Traceability	0.11	0.11	0	0.25	0.067	0.12	0.040

Table 7 Overview of moderation effects of the amount of bidirectional use-cases

	Comprehensibility	Ease of writing	Expressiveness	Tool support	Maintainability	Productivity	Reusability
Bidirectionality	0.067	0.011	0.136	0.101	0.026	0.092	0.010
Incrementality	0.036	0.005	0.120	0.115	0.003	0.099	0.019
Mappings	0.029	0.034	0.065	0.084	0.002	0.072	0.001
Model management	0.143	0.048	0.175	0.066	0.010	0.053	0.014
Model navigation	0.058	0.003	0.155	0.049	0.037	0.071	0.020
Model traversal	0.024	0.068	0.142	0.071	0.023	0.117	0.012
Pattern matching	0.045	0.006	0.110	0.092	0.002	0.058	0.002
Reuse mechanisms	0.040	0.026	0.130	0.220	0.100	0.190	0.013
Traceability	0.065	0.120	0.150	0.160	0.060	0.150	0.012

Table 8 Overview of moderation effects of developer experience

	Comprehensibility	Ease of writing	Expressiveness	Tool support	Maintainability	Productivity	Reusability
Bidirectionality	0.05	8.4E-06	0.011	0.08	0.037	0.084	0.033
Incrementality	0.045	0.0003	0.007	0.083	0.032	0.055	0.046
Mappings	0.00032	0	0.0013	0.072	0.052	0.1	0.039
Model management	0.15	0.025	0.032	0.085	0.05	0.073	0.034
Model navigation	0.053	0.043	0.013	0.08	0.032	0.074	0.05
Model traversal	0.045	0.0008	0.011	0.1	0.024	0.14	0.054
Pattern matching	0.013	0.024	0.0052	0.081	0.032	0.078	0.048
Reuse mechanisms	0.047	0	0.0015	0.08	0.077	0.055	0.022
Traceability	0.072	0.017	0.011	0.055	0.042	0.066	0.04

Table 9 Overview of moderation effects of the semantic gap between input and output

	Comprehensibility	Ease of writing	Expressiveness	Tool support	Maintainability	Productivity	Reusability
Bidirectionality	0.01	0.033	0.008	0	0.089	0.056	0.041
Incrementality	0.006	0.067	0.009	0	0.119	0.046	0.045
Mappings	0.013	0.033	0.002	0	0.053	0.049	0.043
Model management	0.1	0.023	0.021	0.009	0.163	0.052	0.054
Model mavigation	0.027	0.01	0.001	0.007	0.112	0.1	0.043
Model traversal	0.023	0.032	0	0.002	0.194	0.067	0.029
Pattern matching	0.032	0.033	0	0.029	0.239	0.096	0.032
Reuse mechanisms	0.016	0.033	0.008	0.007	0.237	0.068	0.033
Traceability	0.032	0.027	0.0003	0.012	0.184	0.058	0.038

Table 10 Overview of moderation effects of the sanity of involved meta-models

	Comprehensibility	Ease of writing	Expressiveness	Tool support	Maintainability	Productivity	Reusability
Bidirectionality	0.2	0.0001	0.00002	0.035	0.025	0.074	0.045
Incrementality	0.06	0.06	0.001	0.03	0.013	0.097	0.047
Mappings	0.09	0.03	0.03	0.07	0.005	0.074	0.059
Model management	0.21	0.02	0.02	0.006	0.03	0.086	0.073
Model navigation	0.11	0.12	0.002	0.06	0.04	0.099	0.044
Model traversal	0.1	0.08	0	0.08	0.02	0.13	0.069
Pattern matching	0.09	0	0.01	0.05	0.01	0.07	0.081
Reuse mechanisms	0.11	0	0.008	0.03	0.05	0.094	0.059
Traceability	0.11	0.001	0	0.05	0.05	0.14	0.079

Table 11 Overview of moderation effects of the amount of incremental use-cases

	Comprehensibility	Ease of writing	Expressiveness	Tool support	Maintainability	Productivity	Reusability
Bidirectionality	0	0.06	0.039	0.0045	0.034	0	0.00031
Incrementality	0	0.00067	0.012	0.0042	0.079	0	0.006
Mappings	0	0.017	0.079	0.0018	0.061	0.0072	0.017
Model management	0.13	0.027	0.068	0.016	0.062	0.00038	0.039
Model navigation	0	0.24	0.066	0.037	0.098	0.033	0.023
Model traversal	0	0.047	0.025	0.028	0.083	0	0.0091
Pattern matching	0.052	0	0.024	0.00048	0.073	0.032	0.034
Reuse mechanisms	0.047	0	0.024	0.0036	0.14	0.051	0.034
Traceability	0.059	0	0.088	0.005	0.13	0	0.029

Table 12 Overview of moderation effects of the choice of language

	Comprehensibility	Ease of writing	Expressiveness	Tool support	Maintainability	Productivity	Reusability
Bidirectionality	0	0	0.045	0	0.022	0	0
Incrementality	0	0	0.058	0	0.0035	0	0
Mappings	0	0.0063	0.043	0	0.0047	0.0052	0
Model management	0.080	0	0.0050	0.0084	0.0077	0.0058	0
Model navigation	0	0.031	0.051	0.022	0.038	0.027	0.0015
Model traversal	0	0.00081	0.049	0.0075	0.040	0	0
Pattern matching	0.028	0.00081	0.018	3.4E-05	0.019	0.0076	0.0065
Reuse mechanisms	0.037	6.2E-09	0.042	0.0015	0.102	0.027	0.029
Traceability	0.036	0.022	0.0042	0.026	0.024	0	0.041

Table 13 Overview of moderation effects of the experience in the used languages

	Comprehensibility	Ease of writing	Expressiveness	Tool support	Maintainability	Productivity	Reusability
Bidirectionality	0	0	0	0	0.0006	0	0
Incrementality	0	0	0.0086	0	0	0	0
Mappings	0	2.4E-17	4E-07	0	0	0.0006	0
Model management	0.090	0	0.021	0.024	0	0.0012	0
Model navigation	0	0.008	0.0022	0.0088	0.0014	0.043	0.015
Model traversal	0	0.0008	0	0.0065	0.0001	0	0
Pattern matching	0.0005	1.5E-19	4.4E-05	0.00049	0.00069	0.0011	0.0065
Reuse mechanisms	0.0056	0.0014	0.0077	0.0071	0.018	0.025	0.0022
Traceability	0.022	2E-17	4E-06	0.028	0.014	0	0.016

B Mail Templates

Dear ${Author Name},

We found your contact information while conducting a structured literature search on model-to-model transformations. We seek your expertise on that topic.

We invite you to participate in our study 'The Impact of Model Transformation Language Capabilities on the Perception of Language Quality Properties' (see below for the URL). It will only take about 20-25 min to complete our online survey. We believe your answers can provide meaningful insights and help drive the field further.

Our survey is based on a large-scale interview study that qualitatively assessed what the community believes to be the main factors that drive the advantages and disadvantages of Model Transformation Languages for model-to-model transformations. Our results have been published in the Empirical Software Engineering journal https://doi.org/10.1007/s10664-022-10194-7

Our survey now quantifies our results to provide a clear picture of which of our identified factors are most important. The methodology for this survey has been peer reviewed at the Registered Reports track at ESEM'22 and is available under https://doi.org/10.48550/arXiv.2209.06570

The survey is available at
https://sp2.informatik.uni-ulm.de/limesurvey/index.php/112652?lang=en
It will be open till January 15, 2023.
All responses are completely anonymous.
Many thanks for supporting our research!
Best regards
Stefan Höppner

C Data Privacy Agreement

To invite people to participate in this survey, we used author information (first name, last name and email address) from published academic papers in the domain of model driven software engineering. We will delete your personal information 2 months after you received the invitation.

The participation in this online survey is anonymous. Your name and email address are only used to invite you.

For future publications, we will publish and further process the anonymous raw data collected in this survey. This includes aggregating and statistical analysis of answers provided by participants. We will perform this in a way that does not allow inferring the identity of individual participants (e.g., by stripping free text answers of information that could identify a participant). Contact Information

If you have any questions about this survey or the data you provided, please contact us:

Stefan Höppner
stefan.hoeppner@uni-ulm.de
Institute of Software Engineering and Programming Languages,
Ulm University,
James-Franck-Ring, 89069 Ulm, Germany

Funding Open Access funding enabled and organized by Projekt DEAL.

Declarations

Conflicts of interest The authors have no competing interests to declare that are relevant to the content of this article.

References

Anastasakis K, Bordbar B, Georg G, Ray I (2007) UML2Alloy: A Challenging Model Transformation. In: Berlin Weil F (ed) Engels G, Opdyke B, Schmidt D C. Model Driven Engineering Languages and Systems. Springer, Berlin Heidelberg, pp 436–450 (**ISBN:978-3-540-75209-7**)

Anjorin A, Buchmann T, Westfechtel B(2017) The Families to Persons Case. In: TTC'17

Arendt T, Biermann E, Jurack S, Krause C, Taentzer G (2010) Henshin: advanced concepts and tools for in-place EMF model transformations. In: Model driven engineering languages and systems. MODELS 2010. https://doi.org/10.1007/978-3-642-16145-2_9

Baker P, Loh S, Weil F (2005) Model-Driven Engineering in a Large Industrial Context – Motorola Case Study. In: Briand L, Williams C (eds) Model driven engineering languages and systems. Berlin, Heidelberg: Springer , pp. 476–491

Balogh A, Varró D (2006) Advanced model transformation language constructs in the VIATRA2 framework. In: Proceedings of the 2006 ACM symposium on applied computing. SAC '06. https://doi.org/10.1145/1141277.1141575

Benelallam A, Gómez A, Tisi M, Cabot J (2015) Distributed model-to-model transformation with ATL on MapReduce. In: Proceedings of the 2015 ACM SIGPLAN international conference on software language engineering, pp. 37–48

Bézivin J (2004) In search of a basic principle for model driven engineering. In: Novatica Journal, Special Issue 5.2, pp. 21–24

Biermann E, Ermel C, Taentzer G (2010) Lifting parallel graph transformation concepts to model transformation based on the eclipse modeling framework. In: Electronic communications of the EASST 26

Brambilla M, Cabot J, Wimmer M (2017) Model-Driven Software Engineering in Practice. Springer International Publishing. https://doi.org/10.1007/978-3-031-02549-5

Brown AW, Conallen J, Tropeano D (2005) Introduction: models, modeling, and model-driven architecture (mda). In: Model-Driven software development. Springer, pp. 1–16. https://doi.org/10.1007/3-540-28554-7_1

Buckler F, Hennig-Thurau T (2008) Identifying hidden structures in marketing's structural models through universal structure modeling. In: Marketing ZFP 30.JRM 2, pp. 47–66

Cabot LBJ, Gérard S (2019) The future of model transformation languages: an open community discussion. In: Journal of object technology. https://doi.org/10.5381/jot.2019.18.3.a7

Chechik M, Famelis M, Salay R, Strüber D (2016) Perspectives of Model Transformation Reuse. In: Ábrahám E, Huisman M (eds) Integrated Formal Methods. Springer International Publishing, Cham, pp 28–44

Cuadrado JS, Molina JG, Tortosa MM (2006) RubyTL: A Practical, Extensible Transformation Language. In: Model Driven Architecture - Foundations and Applications. ECMDA-FA 2006. https://doi.org/10.1007/11787044_13

Czarnecki K, Helsen S (2006) Feature-based survey of model transformation approaches. In: IBM Systems Journal. https://doi.org/10.1147/sj.453.0621

Evora J, Hernandez JJ, Hernandez M (2014) Advantages of model driven engineering for studying complex systems. In: Natural computing 14.1, pp. 129–144. https://doi.org/10.1007/s11047-014-9469-y

Fowler M (2011) Domain-specific languages. Addison-Wesley, p. 597

Fuchs C, Diamantopoulos A (2009) Using single-item measures for construct measurement in management research: Conceptual issues and application guidelines. In: Die Betriebswirtschaft 69.2, p. 195

George L, Wider A, Scheidgen M (2012) Type-safe model transformation languages as internal DSLs in scala. In: Theory and practice of model transformations. ICMT 2012. https://doi.org/10.1007/978-3-642-30476-7_11

Gerpheide CM, Schiffelers RRH, Serebrenik A (2016) Assessing and improving quality of QVTo model transformations. In: Software quality journal 24.3, pp. 797–834. https://doi.org/10.1007/s11219-015-9280-8. ISSN: 1573-1367

Götz S, Tichy M (2020) Investigating the origins of complexity and expressiveness in ATL transformations. In: J Object Technol 19.2, pp. 12–1

Graziotin D, Lenberg P, Feldt R, Wagner S (2021) Psychometrics in behavioral software engineering: a methodological introduction with guidelines". In: ACM Trans Softw Eng Methodol 31.1. https://doi.org/10.1145/3469888

Groner R, Juhnke K, Götz S, Tichy M, Becker S, Vijayshree V, Frank S (2021) A survey on the relevance of the performance of model transformations. In: J Object Technol 20.2. OPEN REGULAR ISSUE, 2:1–27.https://doi.org/10.5381/jot.2021.20.2.a5. http://www.jot.fm/contents/issue_2021_02/article5.html. ISSN:1660-1769

Götz S, Tichy M, Groner R (2021) Claimed advantages and disadvantages of (dedicated) model transformation languages: a systematic literature review. In: Software and systems modeling 20.2, pp. 469–503. https://doi.org/10.1007/s10270-020-00815-4

Hebig R, Seidl C, Berger T, Pedersen JK, Wasowski A (2018) Model transformation languages under a magnifying glass: a controlled experiment with Xtend, ATL, and QVT". In: Proceedings of the 2018 26th ACM joint meeting on european software engineering conference and symposium on the Foundations of Software Engineering. ESEC/FSE 2018. https://doi.org/10.1145/3236024.3236046

Hermans F, Pinzger M, van Deursen A (2009) Domain-specific languages in practice: a user study on the success factors". In: Model driven engineering languages and systems. MODELS 2009. ISBN: 978-3-642-04425-0. https://doi.org/10.1007/978-3-642-04425-0_33

Hinkel G (2016) NMF: A Modeling Framework for the. NET Platform, KIT

Hinkel G, Burger E (2019) Change propagation and bidirectionality in internal transformation DSLs. In: Software & systems modeling. https://doi.org/10.1007/s10270-017-0617-6

Horn T (2013) Model querying with FunnyQT. In: Duddy K, Kappel G (eds) Theory and Practice of Model Transformations. Springer Berlin, Heidelberg, pp. 56–57. ISBN: 978-3-642-38883-5

Höppner S, Haas Y, Tichy M, Juhnke K (2022) Advantages and disadvantages of (dedicated) model transformation languages. In: Empirical Software Engineering 27.6, p. 159. https://doi.org/10.1007/s10664-022-10194-7. ISSN: 1573-7616

Höppner S, Kehrer T, Tichy M (2021) Contrasting dedicated model transformation languages versus general purpose languages: a historical perspective on ATL versus Java based on complexity and size. In: Software and systems modeling. https://doi.org/10.1007/s10270-021-00937-3. ISSN: 1619-1374

Höppner S, Tichy M (2022) The impact of model transformation language features on quality properties of MTLs: a study protocol. https://doi.org/10.48550/ARXIV.2209.06570

Höppner S, Tichy M (2023) The impact of model transformation language capabilities on the perception of language quality properties: survey overview". en. In: https://doi.org/10.18725/OPARU-50274

Hutchinson J, Rouncefield M, Whittle J (2011). Model-driven engineering practices in industry. In: Proceedings of the 33rd international conference on software engineering. ICSE '11. Waikiki Honolulu HI, USA: Association for Computing Machinery, 633–642. . https://doi.org/10.1145/1985793.1985882. ISBN: 9781450304450

Hutchinson J, Whittle J, Rouncefield M (2014) Model-driven engineering practices in industry: social, organizational and managerial factors that lead to success or failure. In: Science of computer programming 89. Special issue on success stories in model driven engineering, pp. 144–161. https://doi.org/10.1016/j.scico.2013.03.017. https://www.sciencedirect.com/science/article/pii/S0167642313000786. issn: 0167-6423

Hutchinson J, Whittle J, Rouncefield M, Kristoffersen S (2011) Empirical assessment of MDE in industry. In: Proceedings of the 33rd international conference on software engineering. ICSE '11. https://doi.org/10.1145/1985793.1985858

Jakumeit E et al (2014) A survey and comparison of transformation tools based on the transformation tool contest. In: Science of computer programming. https://doi.org/10.1016/j.scico.2013.10.009

Johannes J, Zschaler S, Fernández MA, Castillo A, Kolovos DS, Paige RF (2009) Abstracting complex languages through transformation and composition. In: Model driven engineering languages and systems. MODELS 2009. https://doi.org/10.1007/978-3-642-04425-0_41

Jouault F, Allilaire F, Bézivin J, Kurtev I, Valduriez P (2006) ATL: a QVT-like transformation language. In: Companion to the 21st ACM SIGPLAN symposium on object-oriented programming systems, languages, and applications. OOPSLA '06. https://doi.org/10.1145/1176617.1176691

Kahani N, Bagherzadeh M, Cordy JR, Dingel J, Varró D (2019) Survey and classification of model transformation tools. In: Software & systems modeling. https://doi.org/10.1007/s10270-018-0665-6
Kernighan BW, Pike R (1984) The Unix Programming Environment. Prentice Hall, Inc. ISBN: 0-13-937699-2
Ko J-W, Chung K-Y, Han J-S (2015) Model transformation verification using similarity and graph comparison algorithm. In: Multimedia tools and applications 74.20, pp. 8907–8920. https://doi.org/10.1007/s11042-013-1581-y. ISSN: 1573-7721
Kolovos DS, Paige RF, Polack FAC (2008) The epsilon transformation language. In: Theory and practice of model transformations. ICMT 2008. https://doi.org/10.1007/978-3-540-69927-9_4
Kramer ME, Hinkel G, Klare H, Langhammer M, Burger E (2016) Controlled experiment template for evaluating the understandability of model transformation languages. In: Goulao M (ed) 2nd International workshop on human factors in modeling, HuFaMo 2016; Saint Malo; France;4 October 2016 through. vol. 1805. CEUR Workshop Proceedings, CEUR Workshop Proceedings, pp. 11–18
Kuckartz U (2014) Qualitative text analysis: a guide to methods, practice and using software. Sage. ISBN: 978-1-4462-6774-5
Kusel A, Schönböck J, Wimmer M, Kappel G, Retschitzegger W, Schwinger W (2015) Reuse in model-to-model transformation languages: are we there yet? In: Software & systems modeling 14.2, pp. 537–572. https://doi.org/10.1007/s10270-013-0343-7. ISSN: 1619-1374
Lano K, Clark T, Kolahdouz-Rahimi S (2015) A framework for model transformation verification. In: Formal aspects of computing 27.1, pp. 193–235
Liebel G, Marko N, Tichy M, Leitner A,Hansson J (2016) Model-based engineering in the embedded systems domain: an industrial survey on the state-of-practice. In: Software & systems modeling 17.1, pp. 91–113. https://doi.org/10.1007/s10270-016-0523-3
Mens T, Gorp PV 2006) A taxonomy of model transformation. In: Electronic notes in theoretical computer science (GraMoT 2005). https://doi.org/10.1016/j.entcs.2005.10.021
Metzger A (2005) A systematic look at model transformations. In: Model-driven software development. Springer, pp. 19–33. https://doi.org/10.1007/3-540-28554-7_2
Mohagheghi P, Dehlen V (2008) Where Is the Proof? - A Review of Experiences from Applying MDE in Industry. In: Schieferdecker I, Hartman A (eds) Model Driven Architecture – Foundations and Applications. Springer, Berlin, Heidelberg, pp 342–443
Mohagheghi P, Gilani W, Stefanescu A, Fernandez MA (2013a) An empirical study of the state of the practice and acceptance of model-driven engineering in four industrial cases. In: Empirical software engineering. https://doi.org/10.1007/s10664-012-9196-x
Mohagheghi P, Gilani W, Stefanescu A, Fernandez MA, Nordmoen B, Fritzsche M (2013b) Where does model-driven engineering help? Experiences from three industrial cases. In: Software & systems modeling 12.3, pp 619–639. https://doi.org/10.1007/s10270-011-0219-7. ISSN:1619-1374
Mooney CZ, Mooney CF, Mooney CL, Duval RD, Duvall R (1993) Bootstrapping: A nonparametric approach to statistical inference. 95. sage
OMG (2002) Meta Object Facility(MOF)
OMG (2014) Object Constraint Language (OCL). https://www.omg.org/spec/OCL/2.4/PDF
OMG (2016) Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification. urlhttps://www.omg.org/spec/QVT/About-QVT/
Raggett D, Le Hors A, Jacobs I et al (1999) HTML 4.01 Specification. In: W3C recommendation
SAEMobilus (2004) Architecture Analysis and Design Language (AADL)
Sanchez Cuadrado J, Burgueno L, Wimmer M, Vallecillo A (2020) Efficient execution of ATL model transformations using static analysis and parallelism. In: IEEE Transactions on software engineering, pp. 1–1. https://doi.org/10.1109/TSE.2020.3011388
Schmidt D (2006a) Guest Editor's Introduction: Model-Driven Engineering. In: Computer-IEEE Computer Society. https://doi.org/10.1109/MC.2006.58
Schmidt D (2006b) Guest editor's introduction: model-driven engineering. In: Computer - IEEE Computer Society. https://doi.org/10.1109/MC.2006.58
Selic B (2003) The pragmatics of model-driven development. In: IEEE Software 20.5, pp.19–25. https://doi.org/10.1109/MS.2003.1231146
Sendall S, Kozaczynski W (2003) Model transformation: the heart and soul of model-driven software development. In: IEEE Software. https://doi.org/10.1109/MS.2003.1231150
Sprinkle J, Mernik M, Tolvanen J, Spinellis D (2009) Guest Editors' introduction: what kinds of nails need a domain-specific hammer? In: IEEE Software 26.4, pp. 15–18. https://doi.org/10.1109/MS.2009.92
Stachowiak H (1973) Allgemeine Modelltheorie. Springer. isbn: ISBN 3-211-81106-0
Staron M (2006) Adopting model driven software development in industry – a case study at two companies. In: Model driven engineering languages and systems. MODELS 2006. https://doi.org/10.1007/11880240_5

Steinberg D, Budinsky F, Merks E, Paternostro M (2008). EMF: eclipse modeling framework. Pearson Education

Strüber D, Anjorin A (2016) Comparing reuse mechanisms for model transformation languages: design for an empirical study. In: HuFaMo@ MoDELS. Citeseer, pp. 27–32

Tehrani SY, Zschaler S, Lano K (2016) Requirements engineering in model-transformation development: An interview-based study. In: International conference on theory and practice of model transformations. Springer, pp 123–137. https://doi.org/10.1007/978-3-319-42064-6_9

Van Deursen A, Klint P (2002) Domain-specific language design requires feature descriptions. In: Journal of computing and information technology. https://doi.org/10.2498/cit.2002.01.01

Weiber R, Mühlhaus D (2021) Strukturgleichungsmodellierung: Eine anwendungsorientierte Einführung in die Kausalanalyse mit Hilfe von AMOS, SmartPLS und SPSS. 3rd ed. Springer-Verlag. https://doi.org/10.1007/978-3-658-32660-9

Whittle J, Hutchinson J, Rouncefield M, Burden H, Heldal R (2013) Industrial adoption of model-driven engineering: are the tools really the problem? In: Model-driven engineering languages and systems. MODELS 2013. https://doi.org/10.1007/978-3-642-41533-3_1

Wilke CO (2019) Fundamentals of data visualization: a primer on making informative and compelling figures. O'Reilly Media. isbn: 978-1492031086

Authors and Affiliations

Stefan Höppner[1] · Matthias Tichy[1]

Matthias Tichy
matthias.tichy@uni-ulm.de

[1] Ulm University, James-Franck-Ring 1, 89081 Ulm, Germany