Using Metrics for Assessing the Quality of ATL Model Transformations*

M.F. van Amstel, M.G.J. van den Brand

Department of Mathematics and Computer Science
Eindhoven University of Technology, Eindhoven, The Netherlands
{M.F.v.Amstel|M.G.J.v.d.Brand}@tue.nl

Abstract. Model transformations play a pivotal role in model-driven engineering. Since they are in many ways similar to traditional software artifacts, they have to be treated similarly. Therefore, it is necessary to assess their quality. We propose to use metrics to assess various quality attributes of model transformations In this paper, we focus on model transformations created using ATL, which ATL is currently one of the most widely used model transformation formalisms. We have collected metrics data from a heterogeneous collection of seven model transformations. The quality of the same transformations has been evaluated manually by nineteen ATL experts. We assess whether the metrics are appropriate predictors for the quality attributes by correlating the metrics data with the expert data. To support or refute the correlations, we also acquired qualitative statements from the ATL experts. Although the study was intended as a first exploration of the relation between metrics and quality attributes, some significant correlations were found that are supported by statements of the participants.

1 Introduction

Model transformation is one of the key components of model-driven engineering (MDE) [1]. Since MDE is gradually being adopted by industry [2], model transformations become increasingly important. Model transformations are in many ways similar to traditional artifacts, i.e., they have to be used by multiple developers, have to be changed according to changing requirements and should preferably be reused. Therefore, it is necessary to define and assess their quality. Currently, one of the most widely used formalisms for performing model transformation is ATL [3, 4]. Therefore we focus in this paper on the assessment of the quality of ATL model transformations.

We propose the following six quality attributes for measuring the quality of model transformations: *understandability*, *modifiability*, *conciseness*, *completeness*, *consistency*, and *reusability*. Note that these quality attributes may not

* This work has been carried out as part of the FALCON project under the responsibility of the Embedded Systems Institute with Vanderlande Industries as the industrial partner. This project is partially supported by the Netherlands Ministry of Economic Affairs under the Embedded Systems Institute (BSIK03021) program.

be independent, e.g., understandability may influence reusability since a model transformation needs to be understood, at least partially, before it can be reused. Most of these quality attributes have already been defined earlier for software artifacts in general [5]. However, they are relevant for model transformations as well. In [6], we defined a number of metrics for measuring various aspects of ATL model transformations. However, metrics alone are not enough for assessing quality. They need to be related to the quality attributes. In this paper, we present the preliminary results of an empirical study aimed at finding this relation. Metrics data were automatically collected from a collection of seven ATL model transformations with different characteristics. Quality attributes of the same collection of transformations were quantitatively assessed by nineteen ATL experts using a questionnaire. To establish a relation between the metrics and the quality attributes, we analyzed the correlations between the metrics data and the expert feedback. In addition to the quantitative evaluation task, we used the questionnaire to obtain qualitative statements from the experts. These qualitative statements were used to support or refute the results of the quantitative analysis. This study provides initial insights regarding the appropriateness of using the automatically collected metrics as predictors for the quality attributes.

The remainder of this paper is structured as follows. In Section 2, a subset of the metrics we defined to predict the quality attributes are described. Section 3 describes the results of our empirical study. In Section 4, related work is described. Conclusions and directions for further research are given in Section 5.

2 Metrics

In this section, we present a subset of the metrics we have defined for measuring characteristics of ATL model transformations. For an overview of all the metrics we defined, the reader is referred to [6]. We do not consider all those metrics here, because some of them are too detailed for the purpose of the study presented in this paper. For a more in-depth study or for different quality attributes, they may be relevant.

The metrics can be divided into four categories, viz., rule metrics, helper metrics, dependency metrics, and miscellaneous metrics. In the remainder of this section, we will address each of these categories and elaborate on the metrics belonging to them.

2.1 Rule Metrics

A measure for the size of a model transformation is the *number of transformation rules* it encompasses. In ATL, there are different types of rules, viz., matched rules, lazy matched rules, and called rules. We have defined metrics for measuring the number of rules of every type.

Matched rules are scheduled by the ATL virtual machine, hence they do not have to be invoked explicitly. Lazy matched rules and called rules do need to

be invoked explicitly. Therefore, it may be the case that there are lazy matched rules or called rules in an ATL model transformation that are never invoked. This can have a number of reasons, e.g., the rule has been replaced by another rule. To detect this form of dead code, we propose to measure the *number of unused lazy matched rules* and the *number of unused called rules*.

We have also defined a number of metrics on the input and output patterns of rules. The metrics *number of elements per input pattern* and *number of elements per output pattern* measure the size of the input and the output pattern of rules respectively. For an example, see Listing 1.1. In this example a rule is shown that has one input pattern element and two output pattern elements. To initialize target model elements, an output pattern has bindings that are typically initialized with attributes and references derived from elements in the input pattern (this is also shown in Listing 1.1). We propose to measure the *number of unused input pattern elements* to detect input pattern elements that are never referred to in any of the bindings and may therefore be obsolete. A related metric is the metric *number of direct copies*. This metric measures the number of rules that copy (part of) an input model element to an output model element without changing any of the attributes. Note that this only occurs when the input metamodel and the output metamodel are the same. Called rules do not have an input pattern. Therefore, the metric number of elements per input pattern does not include called rules. Instead, for called rules we measure the *number of parameters per called rule*. It may be the case that some of these parameters are never used. To detect this, we propose to measure the *number of unused parameters per called rule*.

```
rule In2Out {
  from  in_1  : InMetamodel!MetaClassA
  to    out_1 : OutMetamodel!Metaclass1
                  binding_1   <- in_1.AttributeA,
                  binding_2   <- in_1.ReferenceA
                ),

        out_2 : OutMetamodel!Metaclass2
                  binding_1   <- in_1.AttributeB
                )
}
```

Listing 1.1. Example transformation rule

The input pattern of a matched rule can be constrained by means of a filter condition. The metric *number of rules with a filter condition* measures the amount of rules that have such an input pattern. Using such filter conditions, a rule matches only on a subset of the model elements defined by the input pattern. Therefore, it may be the case that there are multiple matched rules that match on the same input model elements. We defined the metric *number of matched rules per input pattern* to measure this. Note that it is required, except

in case of rule inheritance, to have a filter condition on the input pattern since ATL does not allow multiple rules to match on the same input pattern.

Transformation rules can have local variables to provide separation of concerns, i.e., to split the calculation of certain output bindings in orderly parts. To measure the use of local variables in rules, we defined the metric *number of rules with local variables*.

ATL allows the definition of imperative code in rules in a `do` section. This can be used to perform calculations that do not fit the preferred declarative style of programming. To measure the use of imperative code in a transformation, we defined the metric *number of rules with a do section*.

2.2 Helper Metrics

Besides transformation rules, an ATL transformation also consists of helpers. Therefore, the size of a model transformation is also influenced by the *number of helpers* it includes. ATL allows defining helpers in separate units, or libraries as they are called in ATL terminology. Therefore, we also measure the *number of helpers per unit*. This gives an idea of the division of helpers among units.

Similarly to lazy matched rules and called rules, helpers need to be invoked explicitly. Therefore, again, it may be the case there are some helpers present in a model transformation that are never invoked. To detect such unused helpers, we propose to measure *number of unused helpers*.

Helpers are identified by their name, context, and, in case of operation helpers, parameters. It is possible to overload helpers, i.e., define helpers with the same name but with a different context. To measure this kind of overloading we propose to measure *number of helpers per helper name*.

Conditions are often used in helpers. The metric *helper cyclomatic complexity* is related to McCabe's cyclomatic complexity [7], it measures the amount of decision points in a helper. Currently, only `if` statements are considered as decision points. In the future, it could be extended to take into consideration other constructs that influence the flow of control, such as the `for` loop. Also the complexity of OCL expressions could be taken into account when measuring the complexity of a helper [8]. Similar to rules, helpers also allow the definition of local variables. We define the metric *number of variables per helper* to measure the use of variables in helpers.

2.3 Dependency Metrics

An ATL model transformation can consist of multiple units that depend on each other. For measuring this dependency, we defined four metrics. The metrics *number of imported units* and *number of times a unit is imported* are used to measure the import dependencies of units. To measure how the internals of units depend on each other, we defined the metrics *unit fan-in* and *unit fan-out*.

On a lower level, transformation rules and helpers also depend on each other. Transformation rules can invoke (other) lazy matched rules, called rules, and

helpers. Helpers can invoke (other) helpers and called rules. These dependencies are measured using the metrics *rule fan-out*, *helper fan-out*, *lazy rule fan-in*, *called rule fan-in*, and *helper fan-in*. Rules can also refer to each other in an implicit way, i.e., using a `resolveTemp()` expression. This dependency is measured using the metrics *number of calls to* `resolveTemp()` and *number of calls to* `resolveTemp()` *per rule*.

2.4 Miscellaneous Metrics

We defined more metrics that do not fit the discussed categories. The metric *number of units* measures the number of units that make up a model transformation. This metric can provide insight in the size and modularity of a model transformation.

The last two metrics provide insight in the context of the model transformation. It is to be expected that model transformations involving more models are more complex. Therefore, we propose to measure the *number of input models* and the *number of output models*.

3 Empirical Study

The quality of a model transformation is not measurable directly. Therefore, we resort to metrics, which can be measured directly. Before these metrics can be used for assessing the quality of model transformations, we have to establish a relation between the metrics and quality attributes relevant for model transformations. To explore this relation, we conducted an empirical study. In the empirical study we used a collection of seven ATL model transformations with different characteristics. For each of these transformations, metrics data were collected using the metrics collection tool presented in [6]. Quality attributes of the same collection of transformations were quantitatively assessed by nineteen ATL experts using a questionnaire. To establish a relation between the metrics and the quality attributes, we analyzed the correlations between the metrics data and the expert feedback. In this section, we describe the design and results of the empirical study.

3.1 Objects

The objects of our empirical study are seven ATL model transformations. To diversify the object set, we selected transformations from various sources created by different developers. The transformations differ in size, style, structure and functionality.

The first transformation is used to perform some type checking and reference resolving of a PicoJava program. PicoJava is a subset of the full Java language [9]. The transformation was developed in a research project. The second transformation generates a relational database model taking as input a UML class model. It also generates a trace model specifying links between source and

target model elements. Transformation users can configure the transformation behavior by means of a configuration model. The transformation has been used as a case study in a research project [10]. The third transformation has been used for educational purposes. Students were asked to develop a transformation that generates code from a simple state machine language. The purpose of the fourth transformation is to transform an R2ML model into an XML model with R2ML syntax elements. The Rule Markup Language (R2ML) is a general web rule markup language used to enable sharing rules between different rule languages. The transformation language is part of the ATL transformation zoo [11]. The fifth transformation copies a UML2 model. The transformation is part of a collection of MDE case studies [12]. The sixth transformation is part of a chain of refining model transformations presented in [13]. The purpose of the transformation is to enable reliable communication over a lossy channel between two communicating objects. It was developed as part of a research project. The last transformation is the transformation used for conducting the experiment presented in this paper. It takes an ATL model transformation and extracts the metrics presented in Section 2 from it. Table 1 provides an overview of the objects of our empirical study along with some of their characteristics.

No.	Transformation	LOC	# Rules	# Helpers	Purpose
1.	PicoJavaType	227	12	14	PicoJava type checking and reference resolving
2.	R2ML2XML	1125	55	1	Generate an XML document of an R2ML model
3.	SM2NQC	158	13	1	Generate NQC code from state machines
4.	UML2DB	5152	84	76	Generate a relational database model from a UML class model
5.	UML2Copy	4158	199	1	Copy a UML 2 model
6.	Lossy2Lossless	1003	37	2	Enable reliable communication over an unreliable channel
7.	ATL2Metrics	2110	93	28	Extract metrics from ATL transformations

Table 1. Characteristics of the analyzed model transformations

3.2 Participants

The participants in the study were nineteen experienced users of ATL with various backgrounds. Some of them are part of the ATL development team, whereas others use ATL only occasionally. None of the authors participated in the study. The developers of some of the transformations presented in Section 3.1 were among the participants. In order to avoid biased results, measures were taken such that the participants did not evaluate a model transformation they developed themselves.

3.3 Task

The task of the participants was to quantitatively evaluate the quality of one or more of the objects, i.e., ATL model transformations. They were asked to fill

out a questionnaire consisting of 21 questions each addressing one of the quality attributes. To enable checking the consistency of the answers provided by participants, the questionnaire contained at least three similar, but different questions for every quality attribute. For instance, in one of the questions the participants were asked to rate the understandability of the model transformation and in another one they were asked to indicate how much effort it would cost them to comprehend the model transformation. In each question, the participants had to indicate their evaluation on a seven-point Likert scale. For all objects, the same questionnaire was used.

It was likely that the participants had no previous knowledge of the transformations they needed to evaluate. To give them an idea about the purpose of the transformation under study, they were provided with a brief description explaining that purpose. Moreover, the input and output metamodels of the transformation were provided, such that they could run the transformation if desired. The participants could use as much time for the evaluation task as they needed.

In addition to the quantitative evaluation task, we used the questionnaire to obtain qualitative statements from the participants. They were requested to indicate what characteristics of an ATL model transformation in their opinion influences each of the quality attributes.

In total, nineteen participants participated in the empirical study. All participants evaluated at least one of the transformations. There were two participants that evaluated three transformations. This leads to a total of 22 evaluations.

3.4 Relating Metrics to Quality Attributes

To establish a relation between metrics and quality attributes, we analyzed the correlation between them. The data acquired from the questionnaire is ordinal. Therefore, we use a non-parametric rank correlation test [14]. Since the data set is small and we expect a number of tied ranks, we use Kendall's τ_b rank correlation test [15]. This test returns two values, viz. a correlation coefficient and a significance value. The correlation coefficient indicates the strength and direction of the correlation. A positive correlation coefficient means that there is a positive relation between metric and quality attribute and a negative correlation coefficient implies a negative relation. The significance indicates the probability that there is no correlation between metric and quality attribute even though one is reported, i.e., the probability for a coincidence. Note that correlation does not indicate a causal relation between metric and quality attribute. Table 2 shows the correlations that were acquired. The significant correlations are marked. Since we are performing an exploratory study and not an in-depth study, we accept a significance level of 0,10.

The number of input and output models that a transformation takes correlates in a negative way with the quality attributes understandability, modifiability, and completeness. When a transformation takes multiple models as input and generates multiple output models, it is to be expected that the transformation rules of that transformation are more complex, and thereby less understandable

	Understandability		Modifiability		Completeness		Consistency		Conciseness		Reusability	
Metric	C.C.	Sig.	C.C.	Sig.	C.C.	Sig.	C.C.	Sig.	C.C.	Sig.	C.C.	Sig.
# Input models	-,407	,029	-,407	,029	-,391	,038	-,122	,524	-,345	,066	-,218	,249
# Output models	-,407	,029	-,407	,029	-,391	,038	-,122	,524	-,345	,066	-,218	,249
# Units	-,407	,029	-,407	,029	-,391	,038	-,122	,524	-,345	,066	-,218	,249
# Transformation rules	,024	,885	-,086	,604	-,082	,623	-,364	,031	-,273	,099	-,092	,582
# Non-lazy matched rules	,014	,931	,000	1,000	,034	,839	-,029	,861	-,129	,435	-,383	,022
# Lazy matched rules	-,081	,633	,041	,812	-,201	,243	-,058	,741	,051	,765	,239	,168
# Called rules	-,128	,482	-,263	,149	-,158	,389	-,308	,098	-,365	,046	-,347	,060
# Rules with filter	-,005	,977	-,038	,818	-,005	,977	-,049	,771	-,129	,435	-,402	,016
# Rules with do-section	,000	1,000	-,153	,385	-,057	,747	,018	,922	-,108	,540	-,173	,332
# Direct copies	,227	,197	,040	,822	,322	,070	-,059	,745	-,125	,478	-,196	,271
# Rules with local variables	-,013	,944	-,051	,780	,032	,861	-,137	,460	-,178	,328	,302	,100
# Helpers	-,178	,296	-,229	,180	-,201	,243	-,047	,786	-,246	,152	,187	,281
# Unused helpers	-,407	,029	-,407	,029	-,391	,038	-,122	,524	-,345	,066	-,218	,249
# Unused lazy matched rules	-,025	,893	,138	,460	-,152	,419	,026	,892	,076	,687	,217	,251
# Unused called rules	-,128	,482	-,263	,149	-,158	,389	-,308	,098	-,365	,046	-,347	,060
# Calls to `resolveTemp()`	-,352	,049	-,358	,045	-,159	,380	-,088	,632	-,236	,189	-,179	,323
# Elements per output pattern	-,375	,026	-,215	,202	-,228	,180	,124	,472	-,146	,389	-,122	,474
# Rules per input pattern	-,109	,507	-,114	,489	-,130	,434	,029	,861	-,014	,931	,315	,059
# Unused input pattern elements	-,032	,854	,059	,737	-,033	,854	-,056	,758	,033	,854	,297	,097
# Parameters per called rule	-,407	,029	-,407	,029	-,391	,038	-,122	,524	-,345	,066	-,218	,249
# Unused parameters per called rule	-,407	,029	-,407	,029	-,391	,038	-,122	,524	-,345	,066	-,218	,249
# Helpers per helper name (overloadings)	-,033	,855	-,066	,714	-,054	,769	,220	,237	,060	,741	,007	,971
Helper cyclomatic complexity	-,248	,142	-,154	,364	-,357	,037	-,026	,882	-,175	,304	,126	,461
# Variables per helper	,006	,972	,063	,727	,013	,944	-,111	,550	,178	,328	,328	,074
# Variables per rule	-,038	,834	-,076	,676	,070	,700	-,111	,550	-,242	,184	,225	,220
# Imported units	-,252	,164	-,080	,661	-,323	,078	,110	,554	-,027	,883	-,223	,225
# Times a unit is imported	-,318	,088	-,138	,459	-,379	,045	,086	,654	-,084	,656	-,247	,192
Lazy rule fan-in	-,356	,037	-,143	,404	-,397	,021	,005	,976	-,021	,905	-,062	,719
Called rule fan-in	-,407	,029	-,407	,029	-,391	,038	-,122	,524	-,345	,066	-,218	,249
Helper fan-in	,138	,402	,000	1,000	-,024	,885	-,236	,162	-,196	,236	-,005	,977
Rule fan-out	-,223	,175	-,124	,453	-,333	,046	-,157	,351	-,235	,157	,024	,885
Helper fan-out	,020	,907	,183	,278	-,105	,537	,302	,081	,264	,120	,136	,426
# Helpers per unit	-,178	,296	-,229	,180	-,201	,243	-,047	,786	-,246	,152	,187	,281
Unit fan-in	-,407	,029	-,407	,029	-,391	,038	-,122	,524	-,345	,066	-,218	,249
Unit fan-out	-,407	,029	-,407	,029	-,391	,038	-,122	,524	-,345	,066	-,218	,249
# Calls to `resolveTemp` per rule	-,326	,068	-,306	,087	-,106	,558	-,061	,741	-,236	,189	-,153	,399

C.C.: Correlation coefficient
Sig. : Significance (two-tailed)

Table 2. Kendall's τ_b correlations

and modifiable, since information from multiple sources needs to be combined and assigned to the correct targets. The data partially support this. A significant, positive correlation exists between the metrics number of output models

and number of elements per output pattern. Since the transformations in our study all have one element per input pattern, no correlation can be found between this metric and the metric number of input models. In the qualitative part of the empirical study, the participants indicated that the number as well as the complexity of the involved metamodels has a negative effect on understandability, modifiability and also completeness. The reason for this is that the metamodels need to be understood before the transformation can be understood and modified. Moreover, it is hard to detect incompletions in a transformation if the metamodels and their interrelations are not fully understood.

We conducted a similar empirical study for model transformations developed using the ASF+SDF term rewriting system [16]. In that study we found that the size of a transformation expressed in terms of the number of transformation functions correlates negatively with understandability and modifiability. Therefore, we expect to find similar correlations for ATL as well. In Table 2 it is shown that no significant correlation is found between the metrics that measure the amount of transformation rules and helpers, and the quality attributes understandability and modifiability. However, the participants indicated in the qualitative part of the empirical study that they see size as a negative influence on both the understandability and modifiability of an ATL model transformation. Most of the participants indicated that the amount of called rules in particular has a negative impact on understandability and modifiability. The correlation coefficient between the metric number of called rules and understandability and modifiability is negative, however insignificant. A reason that was mentioned by the participants for this negative influence is that called rules (and also `do`-sections) are resorted to when a specific solution to part of a transformation problem is required. This is also addressed as a reason for low reusability of called rules. Table 2 shows that significant negative correlations have been found between the metrics called rule fan-in and lazy rule fan-in and a number of quality attributes. The participants mentioned that the use of, specifically, called rules leads to more complex rule interaction and coupling between rules. Although this argument holds for lazy matched rules as well, it is mentioned less often.

ATL allows defining helpers in separate units, or libraries as they are called in ATL terminology. Table 2 shows that the metric number of units correlates negatively with the quality attributes understandability, modifiability, completeness, and conciseness. Modularizing software is generally considered to be beneficial for its quality. Therefore, a positive correlation would have been expected here. It must be noted that the participants mentioned the use of libraries as an influence on quality only with respect to conciseness and reusability. However, no significant correlation has been found between the metric number of units and reusability. In Table 2 it is also shown that some other metrics related to the use of libraries, viz., unit fan-in, and unit fan-out correlate in a negative way with the quality attributes understandability, modifiability, completeness, and conciseness. A high value for unit fan-in and fan-out indicates a high coupling between the units that comprise a model transformation. In traditional software development, it is considered to be desirable to have low coupling between mod-

ules [17]. A reason for this is that having less interconnections between units reduces the time needed by developers to understand the details of other units. Moreover, a change in a unit can cause a ripple effect, i.e., the effect of the change is not local to the unit. Similarly, an error in one unit can affect other units. The metrics number of imported units and number of times a unit is imported also relate to the use of libraries. However, they correlate less with the quality attributes.

The participants mentioned that simple transformations with one-on-one mappings tend to be the most complete. The number of direct copies measures one-on-one copies. This metric correlates positively with completeness, supporting the claim of the participants. Having simple, or small, transformations was mentioned as having a positive influence on reusability as well. Some of the participants indicated that model transformations should be split up in a chain of smaller transformations, because this increases reusability and, as mentioned before, also understandability. A negative influence on reusability and also modifiability that was mentioned is the use of the `resolveTemp()` expression. The correlations presented in Table 2 partly support this. The reason that was mentioned for this influence is that the use of `resolveTemp()` expressions increases coupling between rules. This is also an explanation for the negative correlation that was found between understandability and the metrics number of calls to `resolveTemp()` and number of calls to `resolveTemp()` per rule.

Unused elements are usually not beneficial for the understandability and modifiability of a transformation because they clutter it. Since unused elements are in principal superfluous, they have an obvious negative effect on conciseness. For the metrics number of unused helpers, number of unused called rules, and number of unused parameters of called rules such correlations have been found. However, no significant correlations can be found for the metrics number of unused lazy rules and number of unused input pattern elements.

In the qualitative part of the empirical study, more feedback was acquired regarding the quality of ATL model transformations of which some cannot be related to metrics directly. The participants indicated that lay-out of the code of a model transformation, as well as the use of proper naming for rules, helpers, and variables will increase the understandability and modifiability of the transformation. Enforcing proper naming by means of a coding convention was mentioned as a positive influence on consistency as well. Proper comments and additional documentation of the requirements and design of the transformation were mentioned as positive influences on completeness and, again, understandability. According to the participants, the use of helpers has a positive influence on almost all quality attributes, albeit that they should not become too complex. Helpers prevent duplication of code, which increases consistency and conciseness. It was mentioned that navigation of the source model should be delegated to helpers rather than implementing it as part of a rule. Besides being beneficial for the quality attributes considered in this paper, experiments have shown that this has a positive effect on performance as well [18]. Rule inheritance has, according to the participants, a positive influence on the quality attributes understandabil-

ity, conciseness, and consistency. The use of rule inheritance can lead to rules that are more concise and have a common pattern, which are in general more understandable. Although, a deep inheritance tree should be avoided, since it decreases understandability again. Similarly to what holds for inheritance trees in object-oriented software, a rule deeper in the rule inheritance tree may be more fault-prone because it inherits a number of properties from its ancestors [19]. Moreover, in deep hierarchies it is often unclear from which rule a new rule should inherit from.

3.5 Threats to Validity

When conducting empirical studies, there are always threats to the validity of the results of the study [20]. Here, we address the potential threats to validity we identified for the empirical study we performed.

The objects used in the study are seven ATL model transformations that have been developed and applied for various purposes. Therefore, they have different characteristics. It must be noted, however, that there is only one transformation that is largely imperative, i.e., 57 of the 84 transformation rules of the UML2DB transformation are called rules and all of the 84 rules have a `do`-section. In Figure 1, a scatter plot is depicted of the data concerning the number of called rules in the transformations and their reusability. Since almost all data points are on the left side of the graph, the significant negative correlation that has been found between the number of called rules and reusability is mainly explained by the two outlying data points that originate from the UML2DB transformation. Even though we use a ranked correlation test to reduce the influence of outlying values, these correlation would not have been found if this transformation would not have been in the object set. Therefore, we cannot base our conclusions solely on the correlations we found. To address this threat to validity, we also collected qualitative data. The participants were requested to indicate the characteristics of an ATL model transformation that in their opinion influences each of the quality attributes. These qualitative statements were used to support or refute the results of the quantitative analysis.

The participants in our study have different backgrounds. Some of them are part of the ATL development team, whereas others use ATL only occasionally. In the questionnaire, the participants were requested to rate their knowledge of ATL. Most of the participants rated their knowledge as *high* or *very high*. Therefore, we do not consider the experience of the participants as a threat to the validity of this study.

The task the participants had to perform was evaluating different quality attributes of model transformations. Evaluating model transformations is not a typical task for a model transformation developer. Moreover, participants are not always the most careful readers [21]. Both these issues may decrease the validity of the data. To address both threats to validity, we posed for each of the quality attributes at least three similar but different questions. The results showed that the responses provided by the participants to each of the similar

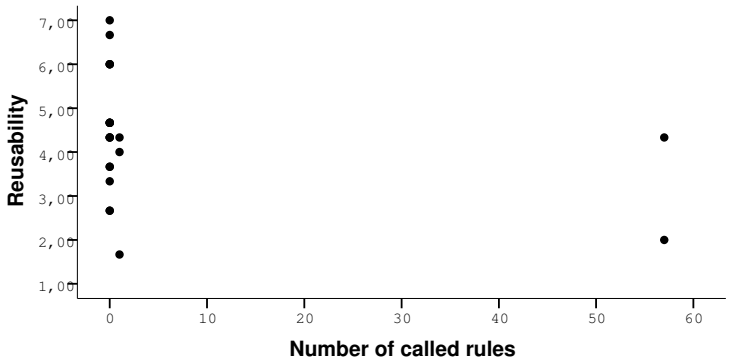

Fig. 1. Relating the number of called rules to reusability

questions were relatively consistent. Also the answers among the participants were relatively consistent. Therefore, we minimized these threats to validity.

4 Related Work

In [16], we presented the results of an empirical study similar to the one presented here. In that paper we focused on model transformations developed using the ASF+SDF term rewriting system. Like the study presented here, that study had an exploratory character as well. Since the number of active users of ASF+SDF is rather small, the number of participants in that study was low. However, for most of the quality attributes we found metrics that correlate with them.

The metrics presented in this paper are specific for ATL. However, conceptually similar metrics can be defined for other model transformation formalisms as well. We have defined metric sets for measuring characteristics of QVT operational mappings and Xtend. In [22], we compare these metrics sets and the metrics set for ATL to find overlap and differences among them. Although we would like to do so, no empirical study has been performed yet to validate whether the metric sets for QVTO and Xtend are valid predictors for the quality attributes.

Kapová et al. have defined a set of metrics for evaluating maintainability of model transformations created with QVT Relations [23]. Most of the 24 metrics they defined are similar to the metrics we have defined. Their extraction process of 21 of their metrics has been automated by means of a tool in a similar way as we have done for ATL [6]. They have applied their tool to three different transformations to demonstrate how to judge the maintainability of a model transformation using their metrics. This judgment is based on expectations rather than empirical evidence. Performing empirical validation is a point they indicate for future work.

Empirical studies have been performed for other types of software artifacts. Basili et al. performed an empirical study to assess whether a set of metrics for

measuring characteristics of object-oriented systems are suitable quality predictors [19]. The participants in their study, eight groups of three students each, had to develop a medium-sized system. Metrics were extracted from the source code of the various developed systems at the end of the implementation phase. Data was also collected on faults detected during the testing phase. Correlations between the metrics data and the fault data were analyzed to assess whether the metrics can be used to detect fault-prone classes. A similar study for model transformations can give valuable insights into the causes for faults in model transformations, and thereby on the influences on their quality. Based on the result of such a study, guidelines can be formulated aimed at decreasing the probability for faults. Lange presents a number of empirical studies aimed at assessing and improving the quality of UML models [24]. In one of the studies, industrial UML models were analyzed for defects. Based on this study, two follow-up studies were conducted to assess whether respectively modeling conventions and the use of visualization techniques can reduce the number of defects in UML models.

5 Conclusions and Future Work

We addressed the necessity for a methodology to assess the quality of model transformations. In software engineering, metrics are frequently used for this purpose. However, metrics alone do not suffice. They have to be related to quality attributes in order to establish whether they serve as valid predictors for these quality attributes. We presented the results of an empirical study in which we try to find relations between metrics that can be automatically derived from a set of ATL model transformations and a quantitative quality evaluation of the same set of transformations by a group of ATL experts. In this study, we also requested the participants to state what in their opinion influences the different quality attributes of an ATL model transformation. Although the study was intended as a first exploration of this relation, some significant correlations were found that are supported by statements of the participants.

To acquire more insights into the relation between metrics and quality attributes, additional empirical studies are required. One type of empirical study that could be considered is one in which the task of the participants is similar to a typical task of a transformation developer in practice, i.e., developing and maintaining model transformations. Although such a study can provide insights that can really help improving the quality of model transformations, finding enough qualified participants for it is hard.

Quality is a subjective concept, i.e., everybody has his or her own perspective on it. Therefore, it is necessary to develop some consensus about quality of, in this case, model transformations. Performing empirical studies to determine this consensus and to validate means for assessing quality are therefore invaluable.

Acknowledgements

We would like to thank all the participants of our empirical study for their time to fill out the questionnaire and the valuable feedback.

We would also like to thank Jan Stoop for his help with the statistical part of this work.

References

1. Schmidt, D.C.: Model-Driven Engineering. Computer **39**(2) (February 2006) 25–31
2. Mohagheghi, P., Fernandez, M.A., Martell, J.A., Fritzsche, M., Gilani, W.: MDE Adoption in Industry: Challenges and Success Criteria. In Chaudron, M.R.V., ed.: Models in Software Engineering: Workshops and Symposia at MoDELS 2008 Reports and Revised Selected Papers. Volume 5095 of Lecture Notes in Computer Science., Toulouse, France, Springer (Sept./Oct. 2008) 54–59
3. Jouault, F., Kurtev, I.: Transforming Models with ATL. In Bruel, J.M., ed.: MoDELS 2005 Satellite Events. Number 3844 in Lecture Notes in Computer Science, Montego Bay, Jamaica, Springer (October 2005) 128–138
4. Jouault, F., Allilaire, F., Bézivin, J., Kurtev, I.: ATL: A model transformation tool. Science of Computer Programming **72**(1-2) (June 2008) 31–39 Special Issue on Second issue of experimental software and toolkits (EST).
5. Boehm, B.W., Brown, J.R., Kaspar, H., Lipow, M., Macleod, G.J., Merrit, M.J.: Characteristics of Software Quality. Volume 1 of TRW Series of Software Technology. North-Holland (1978)
6. van Amstel, M.F., van den Brand, M.G.J.: Quality Assessment of ATL Model Transformations using Metrics. In: Proceedings of the Second International Workshop on Model Transformation with ATL (MtATL 2010). CEUR Workshop Proceedings, Málaga, Spain (June 2010) To appear.
7. McCabe, T.J.: A complexity measure. IEEE Transactions on Software Engineering **2**(4) (December 1976) 308 – 320
8. Cabot, J., Teniente, E.: A metric for measuring the complexity of ocl expressions. In: Workshop on Model Size Metrics (co-located with MODELS 2006), Genova, Italy, month = oct, (2006)
9. : PicoJava Checker. http://jastadd.org/jastadd-tutorial-examples/picojava-checker
10. Muñoz, J., Llacer, M., Bonet, B.: Configuring ATL transformations in MOSKitt. In: Proceedings of the Second International Workshop on Model Transformation with ATL (MtATL 2010). CEUR Workshop Proceedings, Málaga, Spain (June 2010) To appear.
11. : ATL transformations. http://www.eclipse.org/m2m/atl/atlTransformations/
12. Wagelaar, D.: MDE Case Studies. http://soft.vub.ac.be/soft/research/mdd:casestudies
13. van Amstel, M.F., van den Brand, M.G.J., Engelen, L.J.P.: An Exercise in Iterative Domain-Specific Language Design. In Capiluppi, A., Cleve, A., Moha, N., eds.: Proceedings of the Joint ERCIM Workshop on Software Evolution and International Workshop on Principles of Software Evolution (IWPSE-EVOL 2010). ACM International Conference Proceeding Series, Antwerp, Belgium, ACM (September 2010) 48–57
14. Fenton, N.E., Pfleeger, S.L.: Software Metrics: A Rigorous & Practical Approach. Second edn. PWS Publishing Co. (1996)

15. Field, A.: Discovering Statistics using SPSS. Second edn. Sage (2005)
16. van Amstel, M.F., Lange, C.F.J., van den Brand, M.G.J.: Using Metrics for Assessing the Quality of ASF+SDF Model Transformations. In Paige, R.F., ed.: Proceedings of the Second International Conference on Model Transformation (ICMT 2009). Volume 5563 of Lecture Notes in Computer Science., Zurich, Switzerland, Springer (June 2009) 239–248
17. Page-Jones, M.: The practical guide to structured systems design. Second edn. Yourdon Press (1980)
18. van Amstel, M., Bosems, S., Kurtev, I., Pires, L.F.: Performance in Model Transformations: Experiments with ATL and QVT. In Visser, E., Cabot, J., eds.: Proceedings of the Fourth International Conference on Model Transformation (ICMT 2011). Lecture Notes in Computer Science, Zurich, Switzerland, Springer (June 2011) To appear.
19. Basili, V.R., Briand, L.C., Melo, W.L.: A Validation of Object-Oriented Design Metrics as Quality Indicators. IEEE Transactions on Software Engineering **22**(10) (1996) 751 – 761
20. Yin, R.K.: Case Study Research: Design and Methods. Fourth edn. Volume 5 of Applied Social Research Methods Series. Sage (2009)
21. Oppenheimer, D.M., Meyvis, T., Davidenko, N.: Instructional manipulation checks: Detecting satisficing to increase statistical power. Journal of Experimental Psychology **45**(4) (July 2009) 867–872
22. van Amstel, M.F., van den Brand, M.G.J., Nguyen, P.H.: Metrics for model transformations. In: Proceedings of the Ninth Belgian-Netherlands Software Evolution Workshop (BENEVOL 2010), Lille, France (December 2010)
23. Kapová, L., Goldschmidt, T., Becker, S., Henss, J.: Evaluating Maintainability with Code Metrics for Model-to-Model Transformations. In Heineman, G., Kofron, J., Plasil, F., eds.: Proceedings of the Sixth International Conference on the Quality of Software Architectures (QoSA 2010). Volume 6093 of Lecture Notes in Computer Science., Springer (2010) 151–166
24. Lange, C.F.J.: Assessing and Improving the Quality of Modeling: A Series of Empirical Studies about the UML. PhD thesis, Eindhoven University of Technology, Eindhoven, The Netherlands (2007)