2021 47th Euromicro Conference on Software Engineering and Advanced Applications (SEAA) | 978-1-6654-2705-0/21/$31.00 ©2021 IEEE | DOI: 10.1109/SEAA53835.2021.00037

Automated quality assessment of interrelated modeling artifacts

Francesco Basciani
University of L'Aquila, Italy
Email: francesco.basciani@univaq.it

Davide Di Ruscio
University of L'Aquila, Italy
Email: davide.diruscio@univaq.it

Ludovico Iovino
Gran Sasso Science Institute
L'Aquila, Italy
ludovico.iovino@gssi.it

Alfonso Pierantonio
University of L'Aquila, Italy
Email: alfonso.pierantonio@univaq.it

***Abstract*—Over the last decade, several repositories have been proposed by the Model-Driven Engineering (MDE) community to enable the reuse of modeling artifacts and foster empirical studies to analyze specifications and tools made available by MDE researchers and practitioners. In this respect, different approaches have been proposed to measure the quality of, e.g., models, metamodels, and transformations, with respect to characteristics defined by quality models. However, when a modeling ecosystem is available, measuring the constituting artifacts singularly might not be enough. This paper proposes a quality assessment approach, which considers the relationships among the artifacts under analysis as part of the quality measurement process. For instance, to assess the quality of model transformations, further than measuring their structural characteristics, users might be interested in quality aspects like coverage and information loss related to the depending metamodels and the way models are consumed by transformations, respectively. The proposed approach is based on weaving models, which permit to link quality definitions of different kinds of artifacts, and it can generate Epsilon Object Language (EOL) programs by means of a model-to-code transformation to perform the specified quality assessment process.**

***Index Terms*—MDE, Quality model, Quality Evaluation Systems, Modeling ecosystems**

I. Introduction

Model-Driven Engineering (MDE) [1] is a software discipline, which promotes the adoption of models to support the specification, development, and analysis of complex systems. Models are specified with constructs defined in corresponding metamodels and are consumed by model transformations to generate different kinds of artifacts, including documentation and source code.

Having a quality assurance process, definable on different kinds of artifacts, is crucial to build qualified systems. Defining the quality of an MDE artifact means measuring structural characteristics to determine some qualitative properties on the subject artifact. The metrics calcualted on these artifacts provide helpful information to define related properties, e.g., an excessive number of metaclasses in a metamodel could affect their modularity. Furthermore, the composition of multiple properties can lead to new quality definitions for these specific artifacts.

Over the last years, several model repositories have been proposed by the MDE community in response to the need for systems enabling the reuse of already defined modeling artifacts [2]. The availability of model repositories fostered the development of quality assessment approaches able to measure and understand the characteristics of stored modeling artifacts, including models, metamodels, and transformations [3]. Existing approaches identify artifact-specific metrics and discuss how they contribute to assessing quality characteristics like understandability, modularity, reusability and readability (e.g., [4], [5]).

In [6], we proposed a model-based approach to define custom quality models underpinning the analysis of different artifacts, including metamodels, models, and transformations. Thus, the set of elements under analysis are measured according to the characteristics precisely defined in quality models, consisting of hierarchically organized quality characteristics. The limitations of the approach presented in [6] are that modeling artifacts are singularly analyzed without considering their possible interrelationships. For instance, employing the approach in [6], users can assess the reusability, understandability, extendibility of model transformations, but they cannot measure the coverage [7] of their metamodels, or their potentially induced information loss [8].

Coverage and information loss are characteristics that cannot be measured by analyzing the transformation of interest singularly. Their assessment involves the analysis of additional artifacts that might be related to the considered transformations. For instance, concerning coverage, a modeler can prioritize the usage of given transformations with respect to their metamodels: as software testing coverage [9], finding the area of a requirement not implemented by a set of test cases can correspond to find which metamodel portions are not covered by the transformations under analysis. Information loss is a characteristic that represents the information that is lost by the transformation under analysis if it is applied on the given input model. Thus, information loss cannot be measured by analyzing the transformation singularly, but also the input models have to be involved in the analysis.

In this paper, we propose a weaving-based approach to assess the quality of interrelated modeling artifacts that are part of the same ecosystem [10]. The proposed methodology extends the work in [6] by enabling the specification of quality

978-1-6654-2705-0/21/$31.00 ©2021 IEEE
DOI 10.1109/SEAA53835.2021.00037

models that explicitly link different kinds of artifacts that are of interest to the given quality assessment. Runnable quality evaluators are automatically generated from input weaving-based specifications.

The paper is organized as follows: Section II motivates the work by presenting an explanatory example, which is used throughout the paper. Section III presents the proposed approach. Section IV discusses performed experiments and related threats to validity are discussed in Section V. The related work is presented in Section VI, whereas Section VII concludes the paper and outlines some perspective future work.

II. Background and Motivating Scenarios

The availability of model repositories discloses several possibilities, including fostering empirical studies to understand the structural characteristics of, e.g., models, metamodels, and transformations. Thus, inexperienced modelers can acquire knowledge from already developed artifacts and learn best practices and patterns. However, to this end, it is essential to provide users with mechanisms and tools to measure the quality of stored artifacts so that they can filter and focus only on those that satisfy the wanted quality requirements [11].

Quality assessment is witnessing increasing interest in the MDE community, as confirmed by the large corpus of research that has been recently produced [3]–[5]. Metamodels are considered qualitative if they serve their purpose, i.e., they contain all needed abstractions of the represented domain and are built using sound principles [12]. The quality of metamodels might be compromised by the introduction of smells [13], and the perceived quality can be driven by the metamodels completeness, correctness and modularity [14].

Assessing the quality of other modeling artifacts further than metamodels, is also a crucial task, and in the context of model transformations, quality has been classified as *internal* and *external* [15]. The internal quality of a model transformation is related to the artifact itself in terms of characteristics like understandability, modifiability, reusability, and modularity. The external quality of model transformations corresponds to the change of quality induced to the processed models. Thus, in this respect, it is necessary to involve in the quality assessment process also the relationships among the different kinds of artifacts belonging to the same ecosystem [10].

To better characterize and demonstrate the importance of enabling the quality assessment of interrelated modeling artifacts, we consider the metamodels in Fig. 1 and 2, and explanatory model transformations among them to introduce *information loss* and *coverage* as quality attributes, which cannot be assessed by singularly analyzing modeling artifacts.

The metamodel in Fig. 1 permits users to specify lists of `Persons`, and lists of `Cards` each with a unique identifier. The metamodel in Fig. 2 is for representing lists of `Clients`, each with an associated `Badge` allowing the owner to enter a facility, and with a possible `Subscription` for specific services.

For instance, when a person arrives at a sporting centre with a provided card, she can become a client, and the card becomes a badge for the entrance. Moreover, the client can subscribe to a monthly, quarterly, half a year or yearly subscription. This translation can be embodied in model transformations, as for instance T_1, T_2, T_3 and T_4, respectively, reported in the following. The explanatory transformations are written in ATL [16], and they consist of rules defining the mappings among concepts defined in the source and target metamodel(s).

The transformation T_1 is reported in Listing 1 and it consists of 3 matched rules: `List2List` is responsible of translating input person lists into corresponding lists of clients. `Person2Client` matches input `Person` elements to create corresponding `Client` instances, and sets the client name with the input person's name. Finally, `Card2Badge` creates an entrance `Badge` for each non initialized `Card` by using the input `uid` attribute.

```
1 rule List2List{
2  from s:SOURCE!PersonList
3  to t: TARGET!ClientList(
4   clients <- s.persons,
5   badges <- s.cards
6  )
7 }
8 rule Person2Client{
9  from s:SOURCE!Person
10  to t: TARGET!Client(
11   name <- s.name
12  )
13 }
14 rule Card2Badge{
15  from s:SOURCE!Card
16  to t: TARGET!Badge(
17  uid <- s.uid
18  )
19 }
```

Listing 1. *Transformation T1*

The transformation T_2 in Listing 2 consists of two rules, the first one is the same in T_1. The second rule creates a `Client` instance from an input `Person` as done in T_1. Additionally, the `Card2Badge` rule creates a yearly `Subscription` for that client.

```
1 rule List2List{
2  from s:SOURCE!PersonList
3  to t: TARGET!ClientList(
4   clients <- s.persons,
5   badges <- s.cards
6  )
7 }
```

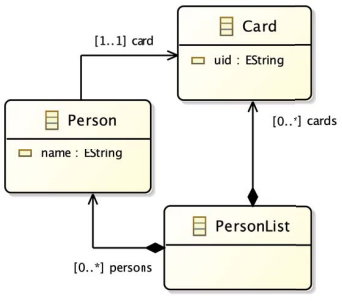

Fig. 1. Source metamodel

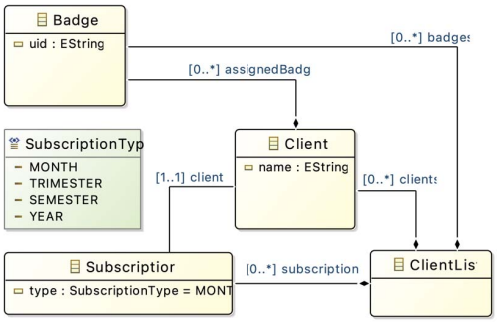

Fig. 2. Target model

```
8 rule Person2Client{
9  from s:SOURCE!Person
10  to t: TARGET!Client(
11   name <- s.name
12  ),
13  t1: TARGET!Subscription(
14   client <- t,
15   type <- #YEAR
16  )
17 }
```

Listing 2. *Transformation T2*

Transformations T_3 and T_4 (which are not reported due to the lack of space) are as T_1 by differing only for line 17. In particular, T_3 does not include the binding for the attribute `uid`. Such a missing binding might introduce loss of information if the input model has `Card` instances with the attribute `uid` set. Constrariwise, if the value for `uid` is unset, then the transformation T_3 would not affect the information passed to the output model. Concerning T_4, it has a default value assigned to the `uid` attribute, which is assigned by the binding `uid` $\leftarrow$ `'1234'` at line 17.

Despite the very simple differences of the four explanatory transformations, in the following we discuss their potential effects in terms of information loss by applying them on the same input model.

Transformation Information Loss *Comparing the target models of T1 and T2 with the same input model.*

Figure 3 shows the parallel application of T_1 and T_2 on the same input model depicted on the left-hand side of the figure. The corresponding output model that are produced by T_1 and T_2 are shown on the right-hand side of Fig. 3. The model generated by T_1 consists of two `Client` instances and of the corresponding `Badge` elements. Note that the values of the attributes `name` and `uid` are copied from the elements of the input model, whereas the associations between the `Person` and `Card` instances have been lost during the transformation phase. The model produced by T_2 includes `Subscription` instances in relation with corresponding `Client` elements. The input `Card` instances are not translated, with a consequent loss of information.

If we quantitatively analyze the generated models, they contain four instances each, and the model generated by T_2 contains two additional associations. Thus, the output model of T_2 contains more information than that produced by T_1.

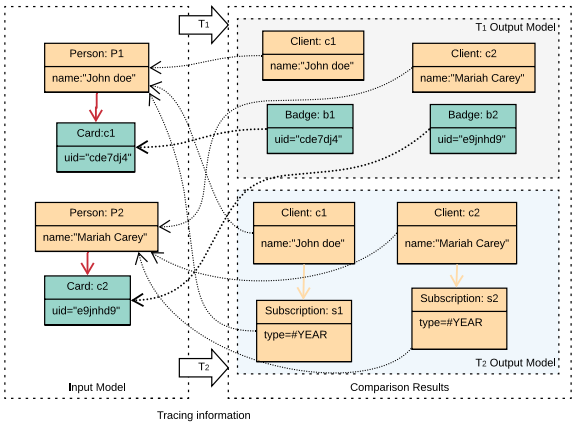

Fig. 3. Application of T_1 and T_2 on the same input model

Comparing the target models of T1, T3 and T4 with the same input model.

Figure 4 shows the results obtained by the parallel application of T_1, T_3 and T_4 on the same input model. Due to the differences in the `Card2Bedge` rule previously presented, T_3 induces loss of information related to the missing `uid` binding, whereas T_4 induces loss of information since it sets the `uid` attribute with a static value loosing the real `uid` of the input cards.

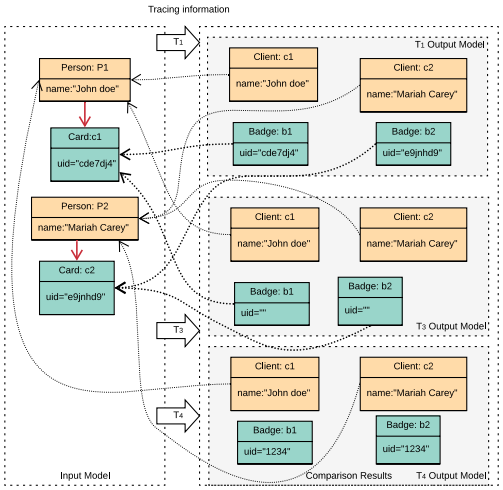

Fig. 4. Application of T1, T3, and T4 on the same input model

Model/Metamodel Coverage. If we consider the concept of *model coverage* borrowed from [17] and inspired by category-partition testing [18], an additional evaluation can be performed on the considered ecosystems. The idea is to evaluate the adequacy of a model with respect to the coverage of its corresponding metamodel.

Figure 5 depicts two models conforming to the metamodel in Fig. 1. On the left hand-side of Fig. 5, the model PersonList list1 is characterized by a complete coverage of its metamodel.

In fact, all the concepts of the metamodel are instantiated. On the contrary, the model PersonList list2 is characterized by a lower coverage level. Thus, the first model looks preferable for all the activities in the ecosystem, but what about the model PersonList list3? The answer is "it depends on the purpose of the models": if the model is going to be used as input for a transformation then information loss has to be taken into account; if the model is used as test case for a model transformation, then the coverage may be considered for testing purposes [19], [20].

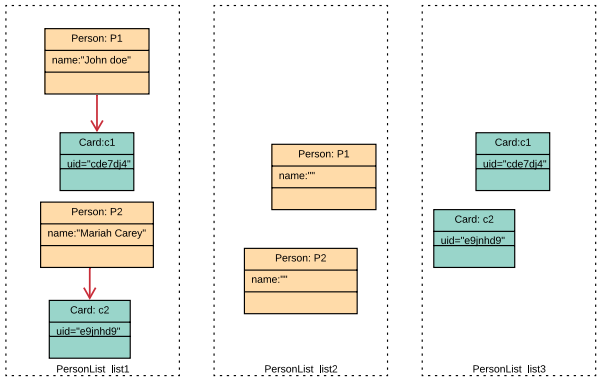

Fig. 5. Model Coverage in comparison

For this reason it becomes essential to have a customizable approach for multi-artefacts quality evaluation in which the modeler can define a custom quality definition for different types of artifacts belonging to the same ecosystem.

III. Proposed Approach

In this section we propose an approach to characterize modeling ecosystems with user-defined quality attributes. The proposed approach is model-based and for this reason we first characterize the involved models, and subsequently we explain them one by one. The process starts with the definition of quality models conforming to the metamodel shown in Fig. 6 and referred with `QualityMM` hereafter. It is an extension of the metamodel proposed in [21] to add the support for defining quality specifications involving multiple artifacts. According to `QualityMM`, a quality model consists of `QualityAttributes`, which in turn are defined in terms of `Operations`, SUM or AVG calculated on the specified `Metrics`. A metric is a qualitative evaluation, as for instance *"The information that is lost by a model transformation in terms of source instances that are not considered for generating the target ones."*. Metrics are implemented in terms of `metricProviders`, which are developed by means of the EOL language [22]. EOL is an imperative programming language for creating, querying and modifying EMF models. Each metric provider can be specified in a EOL library and this is indicated in the same quality model, with the property `library`, which binds the specific metric with the executor operation. Moreover, operations can be nested to build complex formulas. For instance, *getInfoLoss* is an example of metric provider implemented in the `IL.eol` library shown in Listing 3 (lines 1–13) which is the code used to determine the information loss in the Sec. IV.

```
1  operation getIInfoLoss(_trace: Any, inmodel: Any) : Real
       {
2  //get all instances in the input model not cited in the
       trace
3  var I=0;
4  var mapped_instances=_trace.getAllOfKind("TraceLink").
       sourceElements.collect(o|o.object).flatten();
5  for (c in sourceMM!EClass.all) {
6          for (i in inmodel.getAllOfKind(c.name)) {
7          if(mapped_instances->select(obj|emfTool.
                ecoreUtil.equals(obj,i)).size()=0){
8                  I=I+weight_i_produced;
9              }
10         }
11 }
12         return I;
13 }
14 ...
15 operation getCoverage(inmodel: Any, classes: List<
       sourceMM!EClass>): Real{
16 var modelcoverage : Real =0.0;
17 var coverageattrs: Real =0.0;
18 var coveragerefs: Real =0.0;
19         for (c in getConcreteClasses()) {
20         var coverage=0.0;
21         if(inmodel.getAllOfType(c.name).size>0){
22                 //at least one instance per type should
                       be covered
23                     coverage+=weight_i_cov;
24         coverageattrs = getACoverage(inmodel, c.
               eStructuralFeatures.select(f|f.isTypeOf(
               sourceMM!EAttribute)).asSequence());
25         coveragerefs = getRCoverage(inmodel, c.
               eStructuralFeatures.select(f|f.isTypeOf(
               sourceMM!EReference)).asSequence());
26         }
27         //customize coverage if needed
28         modelcoverage+=((coverage+coverageattrs+
               coveragerefs)/3);
29         }
30         return modelcoverage/getConcreteClasses().size;
31 }
```

Listing 3. *Explanatory metric provider implemented in the IL library*

This operation calculates the information loss induced by a transformation from a given input model (see the input parameter `inmodel`). The operation takes as parameter also an input trace model (`_trace`), so that with a single model we can derive the executed transformation, and the produced output model.

A trace model is a model derived by the execution of a transformation [23], and it contains trace links holding how the instances in the input model have been transformed into instances of the output and with which transformation rules. This model is fundamental for this operation since it can give information related to information loss, and for this reason, it has been used as a parameter of the library. This specific quality attribute operates on multiple artifacts as anticipated by the presentation of the approach, i.e. transformation, metamodels and models.

The reported *getInfoLoss* operation iterates on all the metaclasses in the source metamodel. For each iteration, if there is a trace link, which maps the source metaclass and there are no instances of that class in the trace model, then the information loss is increased. This library also offers a way

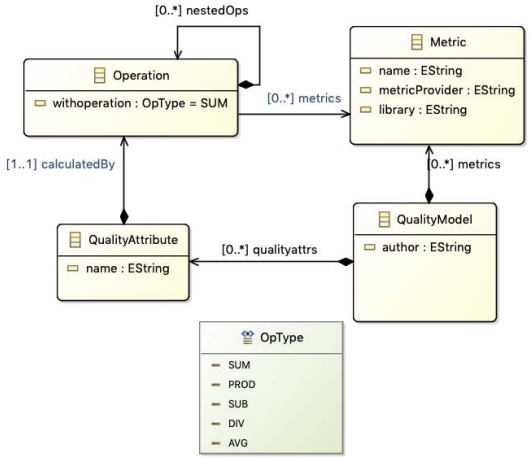

Fig. 6. Quality Definition Metamodel

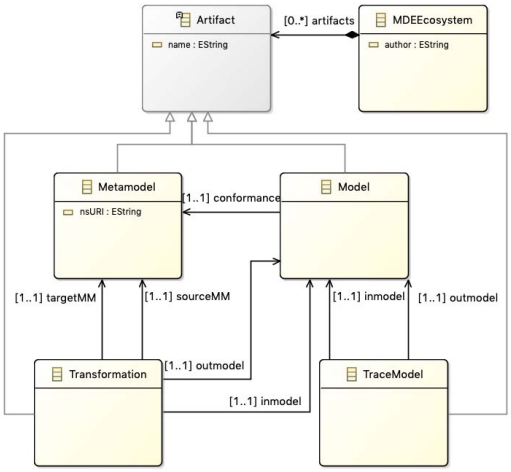

Fig. 7. MDE ecosystem Metamodel

of customizing the weight given to the information coming from instances, attribute values or references (see the constant `weight_i_produced` at line 8 of Listing 3).

Model coverage is calculated by the *getCoverage* operation shown at lines 15-34 of Listing 3. The computed model coverage is given at line 31, where coverage at instance level is added to the coverage of the structural features set in the model. Intuitively, each attribute type has a coverage operation that can be customized. For instance, a boolean attribute defined in the metamodel has to be defined in the model with at least a FALSE and a TRUE value. This is inspired by the coverage definition given in software testing [24].

The quality model and specifically the metric provider elements have an attribute to specify how the metric is calculated and, in particular, the name of the operation in the loaded library. This permits to the engine interpreting the quality

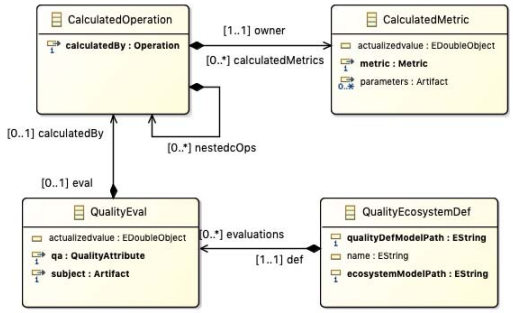

Fig. 8. Quality-Ecosystems Weaving Metamodel

model to bind the metrics to the implemented operations and get the result to be stored in the same model.

Another constituting element of the proposed approach is the `Ecosystem metamodel` shown in Fig 7. By employing such metamodel, MDE ecosystems can be represented as interrelated artifacts. For instance, the motivating example presented in Section II can be represented in a single model conforming to the `Ecosystem metamodel` and consisting of the following elements:

- two `PersonListMM` and `ClientListMM` metamodels shown in Fig.1 and Fig. 2, respectively;
- an input model conforming to the source metamodel named `persons.model`;
- four model transformations, i.e., T_1, T_2, T_3, and T_4 (T_1, and T_2 are reported in Listing 1 and Listing 2, respectively);
- four output models produced by the transformations, named *out*T_1, *out*T_2, *out*T_3, and *out*T_4;
- four tracing information models, i.e., *trace*T_1, *trace*T_2, *trace*T_3, and *trace*T_4.

Once we have all the elements corresponding to the artefacts involved in the selected ecosystem, we can correlate the quality characteristics we want to explore for the artifacts under analysis. To express such links, we conceived the weaving metamodel reported in Fig. 8. A weaving model is a model used to capture relationships between other models [25]. The salient points of the proposed weaving metamodel are the way to express a quality evaluation of the artifact to be analyzed. The metaclass `QualityEval` contains two references to external metamodels, the reference `qa` to link the quality attribute defined in the quality model, e.g., information loss, and the reference `subject` pointing at the artifact defined in ecosystem specification, i.e. a specific transformation.

As for the quality model, also in this woven model the modeler can specify `CalculatedOperation` and `CalculatedMetric`. The semantic of these elements is the same as the quality model, but they also offer the attribute for storing the `actualized value` if the output of the elaboration has to be a model. The variation from the quality model making it a weaving model is also represented by the `parameters` reference. This reference offers the possibility

to link the artifacts of the ecosystem to the metric, mapping the artifacts to the parameters or subject of the evaluation. To make clear this task, we represented the overall process in Fig. 9. This process starts with defining the two main actors on top, namely `Quality Definition Model` and an `MDE Ecosystem`. These two models are then linked by a weaving model called `Quality Ecosystem Model`.

An explanatory weaving model is shown in Fig. 10. On the upper side of the figure, it is possible to notice the initial setting that the modeler can create. The shown weaving model has been managed by Modelink[1]. It consists of three panels: in the central model, `Quality Evaluations` are created, and all of them can be linked to a specified quality attribute (e.g., the information loss as shown on the left-hand side of Fig. 10) and to the artifact, which is the subject of the evaluation (e.g., the trace model of T_1 as shown on the right-hand side of Fig. 10). In this way, we can specify for each artifact in our ecosystem what kind of evaluations we want to obtain at the end of the process.

As shown in Fig. 9, once the weaving model has been specified the `EnrichQualityDef` transformation gets executed. It processes and enriches the weaving model to generate `Enriched Quality-Ecosystem Model`. It is a weaving model with additional information coming from the quality definition used by the modeler. This model is produced by an ETL transformation[2], adding to the quality evaluations the metrics that have to be considered as expressed in the quality model. For instance inspecting the enriched weaving model in Fig. 10 in the center, we can notice that the elements `CalculatedMetrics` have been added to the relative elements. It is worth noting that navigating the enriched weaving model we can obtain all the information needed, as the property view on the bottom left is showing. This property view shows that particular metric for the information loss in terms of instances will be calculated by an EOL operation called `getIInfoLoss`, expressed in the IL.eol library (shown in listing 3). The property view on the right instead shows that the calculated metric will be associated as subject of the evaluation to the `traceT1` (highlighted by the weaving link selector) and it will take as parameters two artifacts of the ecosystem, namely `traceT1` and `inmodel`, corresponding to the arguments of the operation in the IL.eol library (see line 1 of Listing 3).

After the modeler has set up the parameters referring to the artifacts of the ecosystem, the last task is to invoke the `generateEOLProgram` model-to-code transformation, which is shown in Listing 4. It is an EGL script, which can generate the EOL main program able to evaluate the artifacts expressed in the input model. As specified in lines 7-9, the generator generates the `import` instructions for the EOL libraries needed for the execution. Such libraries are specified in the metric providers in the quality model (see the attribute `library` in the metamodel in Fig. 6). Lines 12-14 can be customized with the weights given to instances, attributes, and references in the quality aspects, e.g., instances will weight 1 as attributes and 0.5 for references. Afterwards, the generator produces the calls for all the evaluations to the metric provider constituting the quality attributes required by the modeler.

```
1  [%
2  var tobeimported=new Set;
3  for (eval in qecosystem.evaluations.calculatedBy.collect
       (m|m.calculatedMetrics.metric.library)) {
4      tobeimported+=(eval);
5  }
6  %]
7  [%for (i in tobeimported) { %]
8  import "[%=i%]";
9  [%}%]
10 var emfTool : new Native("org.eclipse.epsilon.emc.emf.
       tools.EmfTool");
11 //customize the weights
12 var  weight_i_produced=1.0;
13 var  weight_a_set=1.0;
14 var  weight_r_set=0.5;
15 [%for (eval in qecosystem.evaluations) { %]
16 "[%=eval.qa.name%] for [%=eval.subject.name%]".println()
       ;
17 var [%=eval.qa.name.toUpperCase().trim()%]_[%=eval.
       subject.name%]=
18     [%
19
20 var op;
21 switch (eval.calculatedBy.calculatedBy.withoperation.
       name) {
22     case "SUM" : op="+";
23     case "PROD" : op="*";
24     case "SUB" :op="-";
25     case "DIV" : op="/";
26     ...
27     default : "unsupported operation please add it
           to the model".println();
28     }
29     %]
30 ([%for (cm in eval.calculatedBy.calculatedMetrics) { %]
31     [%=cm.metric.metricProvider%](
32     [%for (par in cm.parameters) { %]
33         [%=par.name%][%if (hasMore){%],[%}%]
34     [%}%]
35     )
36     [%if (hasMore){%]
37     [%=op%]
38     [%}%]
39 [%}%]).println();
40 [%}%]
```

Listing 4. Fragment of the *generateEOLProgram* template

An excerpt of the generated EOL specification for the previously presented case study is reported in Listing 5.

```
1  import "IL.eol";
2  ...
3  " InformationLoss for traceT1".println();
4  var INFORMATIONLOSS_traceT1=(getAInfoLoss(traceT1,
       inmodel, T1) + getIInfoLoss(traceT1, inmodel) +
       getRInfoLoss(traceT1, inmodel,T1)).println();
5
6  "InformationPreserved for traceT1".println();
7  var INFORMATIONPRESERVED_traceT1=(  getAInfoProd(
       traceT1, T1) + getIInfoProd(traceT1) + getRInfoProd
       (traceT1, T1)).println();
8
9  " InformationLoss for traceT2".println();
10 var INFORMATIONLOSS_traceT2=(getAInfoLoss(traceT2,
       inmodel,  T2)  + getIInfoLoss(traceT2,
       inmodel) + getRInfoLoss(traceT2, inmodel,T2)).
       println();
11
12 "Coverage of the model inmodel".println();
13 var coverage_inmodel=(getCoverage(inmodel, sourceMM!
       EClass.all)).println();
14 ...
```

Listing 5. *Generated EOL program for quality evaluation*

[1]Modelink is the Weaving tool part of the Epsilon ecosystem [22] platform.

[2]The ETL transformation is not reported for lack of space but it is available here: https://bit.ly/3fKeKzZ

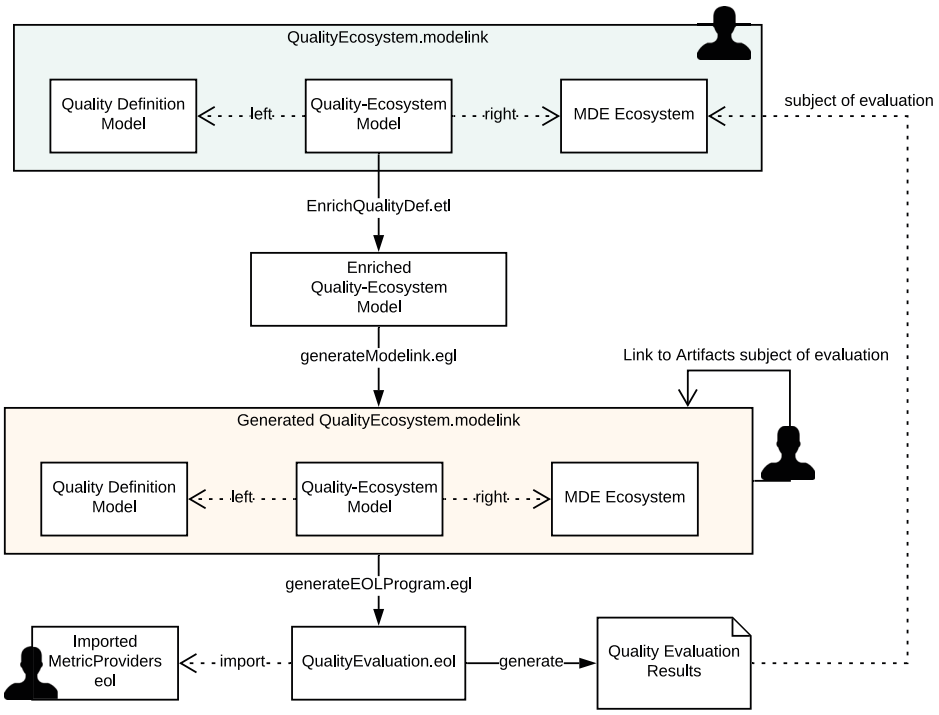

Fig. 9. Proposed Process

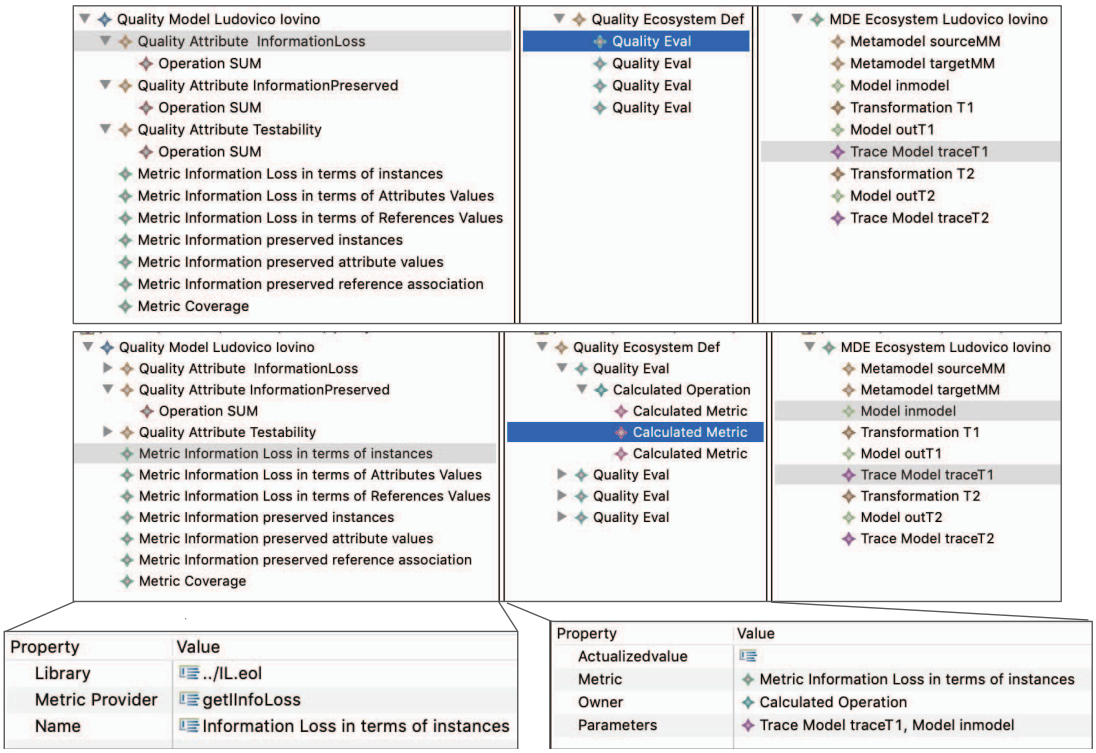

Fig. 10. Explanatory weaving models

Line 1 reports the import needed by the program to invoke the metric providers, and since all the metric providers were part of the same library only one import is generated. Line 4 is responsible for calculating the information loss for the artifact `traceT1` (as highlighted in the weaving model on top of Fig. 10). The called operations from the imported library are specified in the model and can be seen in the field `metric` of the property view in the right-hand side of Fig. 10. The same for the other evaluations from line 6 to 10, corresponding to the two `QualityEval` elements in the weaving model. The execution of the generated EOL program depends on the logic of the generator that can be customized to store the results in the original model or simply print them in the console like in this case.

The result of the execution is printed in the console and it is reported in Fig. 11. From this execution we can notice that, since in Fig. 10 four quality evaluations are declared, the console log prints the result for four evaluations on various artifacts linked by the weaving model.

It is worth noting that the shown evaluation has been computed on multi-artifacts of the same ecosystem, and by considering multiple quality attributes, i.e. information loss, information preserved, and coverage.

IV. Experiments

In this section we present experiments that have been operated to perform the quality assessment presented in the previous section, on the transformations presented in Section II. The goal is to confirm that the information loss and information preserved visually quantified in Section II are confirmed by the automatically obtained result, in order to confirm the validity of the process shown in Fig. 9; then we evaluate a corpus of transformations and input models and we discuss some of the interesting results. We leave out of the evaluation the model coverage since the considered quality model already includes a multi-artifact evaluation scenario.

The result of the presented process on the running example is reported in Table I. The approach selects T_1 as preferable in terms of information loss and information preserved, since the instances of badge carries information of an input instance that is not considered by T_1. Also the `uid` is maintained by T_1 as specified in the binding in line 17 of Listing 1.

T_2 comes after, with loss 5 and preserved 9; T_3 and T_4 are ranked after T_2. In relation to these two last transformations, even if T_4 contains an additional binding than T_3, such a binding is actually producing a static value, not coming from the input model, i.e., 1234. For this reason, the information loss and information preserved are the same and neither of

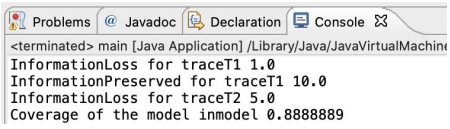

Fig. 11. Log of the execution on the running example

TABLE I
Results of the running example

Artifacts			Information	
In model	*Transformation*	*Out Model*	*Loss*	*Preserved*
persons	T1	T1_out	1	10
persons	T2	T2_out	5	7
persons	T3	T3_out	3	8
persons	T4	T4_out	3	8

those are preferable. It is visible the preference to select T_1 if information loss and preserved are considered as quality attributes in our ecosystem. T_3 and T_4 map two instances more than T_2, where the information coming from a single instance is divided to make two of them in the target model. Moreover, these transformations have the same number of mapped information from attributes. In any case we can conclude that four similar transformations produce different results in terms of information loss and information preserved and analyzing them seems to give useful information, for this case study.

To demonstrate the applicability of the approach and how we have been able to generate all the needed artifacts to calculate the information loss and information preserved, we used a dataset of 5 transformations available on GitHub [26]. For each of them we applied a mutation by adding/removing rules or bindings. Such mutations, in combination with the input models, produce variations of the propagated information. Table II shows the performed experiments in terms of expected and obtained results. In the following we describe the applied mutations to discuss the corresponding results[3]

TABLE II
Validation summary and expected results

Subject			Expected		Effective	
Transformation	**In Model**	**Var**	**IL**	**IP**	**IL**	**IP**
Book2Publication	Book.model	(1)	+	-	+1.5	-1.0
Collaboration2Paper	Collaboration.model	(2)	+	-	+4	-8
Families2Persons	Families.model	(3)	-	+	-1.5	+1.5
Grafcet2Petrinet	Grafcet.model	(4)	=	=	0	0
Company2CRM	Company.model	(5)	=	=	0	0

1) For the variation `Book2Publication.var` the original transformation has been modified with a removal of the bindings at line 23 and 36. Removing this binding predicating over the `title` of the sections, and the `authors` of the books increases the information loss as expected;
2) The variation applied to the `Collaboration2Paper` transformation is simply the removal of the second rule `Researcher2Author`, with the effect of increasing the information linked to the instances of `Researcher` and its properties. Since the initial model contains mul-

[3]Please refer to the content of [26] while reading the description of the considered cases.

tiple instances of authors, the result is quite straightforward, in the increase of the information loss.

3) The variation applied at the transformation `Families2Persons` is the addition of a binding at line 19 predicating over the `gender` of a member of the family. Adding this binding the information loss from the gender set for a member is then propagated, for this reason it is expected to grow.
4) The variation applied to `Grafcet2Petrinet` is simple rather than not obvious. The `location` attribute in the model is not often filled if the model is not related to a graphical editor. For this reason since the removed binding propagates this attribute, and in the input model it is unfilled, then the result of the variation induces no information loss. This is conformed by the execution of the tool chain.
5) Last variation applied to the subject transformation `Company2CRM` does not seem to impact on the information loss. In fact the variation consists of the removal of two rules, one matching the `Unit` and one the `ServiceLine`. Moreover, a binding predicating over the attribute `employed` of the `Person` metaclass. The effect of this variation actually does not impact because the input model does not contain instances of the metaclasses matched by the removed rules. Same for the reference **employed** related to associations with instances of those metaclasses. For this reason the approach confirmed the expected result with 0 delta.

V. Threads to Validity

In this section we discuss peculiar aspects that might compromise the validity of the performed experiments.

a) Transformation under Analysis Inspection: The transformation subject of the evaluation is inspected in combination with its related trace model since the last one only contains the mapping among the in / out instances, without including mapping between features. For this reason we included in the `IL.eol` library operations for navigating the subject transformation bindings. This enables the evaluation of the information loss and information preserved in terms of attributes and set of references. For this reason, since the evaluation is demanded to the developer building the metric providers, the inspection of the transformation can be enriched with additional checks contributing to the customized information loss calculation. An example, which is used in this specific library we built, is the operation `getIInfoLoss`. This operation retrieves the instances lost in the transformation execution inspecting the trace links and then the mappings, but we also enriched this with an additional check adding information loss if the type of the inspected instance is used in one of the bindings. This means that some attributes or references of that instance are used and passed as information is one of the rule. For this reason if the binding does not exist, additional information loss could be introduced.

b) Small subset of transformations: The evaluation has been conducted on a small dataset of simple transformations. The transformations chosen are quite simple since the library developed include operations for navigating the ATL transformation definition, using the `emftvm` compiled version. This model is quite complex and its complexity reflects the complexity of the subject transformation. For this first experiment we selected transformations without complex syntax usage, e.g., helpers or called rules. But for the future work we plan to extend further the dataset and then the inspection mechanism to support more complex transformations.

c) Transformation language specificity: The approach demonstrated in this paper is built on top of the Eclipse Modeling Platform, Ecore as metamodeling language and then ATL as transformation engine. Moreover, the approach is enabled by the trace models generation, which is based on the EMFTVM, so this lead to delimit the approach to a set of specific transformation languages working in the EMF platform. In any case, this does not exclude that the approach can be re-implemented in a different setting, by representing transformation mappings in a different way, e.g., using ad-hoc weaving models.

VI. Related work

Multiple modelling quality evaluation frameworks are methods and tools for evaluating modeling languages in model-driven engineering environments [27]. Novadays, this topic is active and the work in [28] evaluates the applicability of the these methods in comparison with other existing approaches. The authors performed an evaluation by applying quantitative approaches on the results of the performance measures compared with the perception of subjects. This experiment confirms that the subjects did not succeed in identifying all quality problems, highlighting the needs for frameworks like the one proposed in this paper.

Quality attributes in model transformations is an emerging topic widely investigated in literature [29], [30]. In [31] the authors propose an approach for refactoring ATL transformations. Although they focus on improving the quality of transformations to be used in terms of reusability, flexibility, understandability, functionality, extendibility, effectiveness, they do not take into account the instances given as input to the transformations themselves.

The authors in [7] define two correctness properties of model transformations. Weak executability of a rule, expresses whether a rule may be safely applied without breaking the target metamodel constraints while the coverage of a rule set, analyzes whether the rules cover all elements of the source and target metamodels. The authors also presented a tool to check these properties on transformations expressed in ATL language. In [29] the authors present metrics for measuring ATL model transformations with the aim of quality evaluation. Analogously, in [30] metamodel coverage techniques are employed to assess the quality of model transformations with respect to a proposed quality model. of the transformation with respect to the reuse or the maintainability of the artifact.

Another approach presented in [32] for measuring the quality of model transformations relies on the claim that the quality

of model transformations should be assessed during their execution on input models. Indeed, this approach measures metrics both on models and transformations. This approach uses a technique to specify metrics according to models, based on the fact that the metrics of model transformation can be defined from the metrics of the models that appear in the transformation. This is quite related at least in the idea that models are important if considered in combination with the transformations consuming them.

VII. Conclusion and Future Work

In this paper a model-based approach is proposed to support the quality assessment of interrelated artifacts belonging to the the same MDE ecosystem. Users can define quality models to specify the quality attributes to be measured, and the kinds of artifacts that have to be the subject of the evaluation. To show the applicability of the approach in practice, we presented the case of characterizing the quality of model transformations, with respect to the information loss and information preserved when they are applied on given input models. In the future we plan to apply and validate the approach on a larger dataset and by involving domain experts who can specify their own quality concepts and then evaluate their own MDE ecosystem.

Acknowledgment

The research described in this paper has been partially supported by the AIDOaRt Project, EU H2020-ECSEL European Programme, https://www.aidoart.eu/.

References

[1] D. C. Schmidt, "Guest editor's introduction: Model-driven engineering," *Computer*, vol. 39, no. 2, pp. 25–31, 2006.

[2] J. Di Rocco, D. Di Ruscio, L. Iovino, and A. Pierantonio, "Collaborative repositories in model-driven engineering [software technology]," *IEEE Software*, vol. 32, no. 3, pp. 28–34, May 2015.

[3] P. Mohagheghi and J. Aagedal, "Evaluating quality in model-driven engineering," in *International Workshop on Modeling in Software Engineering (MISE'07: ICSE Workshop 2007)*. IEEE, 2007, pp. 6–6.

[4] J. J. López-Fernández, E. Guerra, and J. de Lara, "Assessing the quality of meta-models," in *Proceedings of the 11th Workshop on Model-Driven Engineering, Verification and Validation co-located with 17th International Conference on Model Driven Engineering Languages and Systems, MoDeVVa@MODELS 2014, Valencia, Spain, September 30, 2014*, ser. CEUR Workshop Proceedings, F. Boulanger, M. Famelis, and D. Ratiu, Eds., vol. 1235. CEUR-WS.org, 2014, pp. 3–12. [Online]. Available: http://ceur-ws.org/Vol-1235/paper-02.pdf

[5] Z. Ma, X. He, and C. Liu, "Assessing the quality of metamodels," *Frontiers of Computer Science*, vol. 7, no. 4, pp. 558–570, Aug. 2013.

[6] F. Basciani, J. Di Rocco, D. Di Ruscio, L. Iovino, and A. Pierantonio, "A tool-supported approach for assessing the quality of modeling artifacts," *Journal of Computer Languages*, vol. 51, pp. 173–192, Apr. 2019.

[7] E. Planas, J. Cabot, and C. Gómez, "Two basic correctness properties for atl transformations: Executability and coverage," *CEUR Workshop Proceedings*, vol. 742, 07 2011.

[8] F. Basciani, M. D'Emidio, D. Di Ruscio, D. Frigioni, L. Iovino, and A. Pierantonio, "Automated selection of optimal model transformation chains via shortest-path algorithms," *IEEE Transactions on Software Engineering*, pp. 1–1, 2018.

[9] J. J. Chilenski and S. P. Miller, "Applicability of modified condition/decision coverage to software testing," *Software Engineering Journal*, vol. 9, no. 5, pp. 193–200, 1994.

[10] L. Iovino, A. Pierantonio, and I. Malavolta, "On the Impact Significance of Metamodel Evolution in MDE," *JoT*, vol. 11, no. 3, pp. 3:1–33, Oct. 2012.

[11] F. Basciani, J. Di Rocco, D. Di Ruscio, L. Iovino, and A. Pierantonio, "Model repositories: Will they become reality?" in *CloudMDE@ MoDELS*, 2015, pp. 37–42.

[12] J. J. López-Fernández, E. Guerra, and J. De Lara, "Assessing the quality of meta-models." in *MoDeVVa@ MoDELS*. Citeseer, 2014, pp. 3–12.

[13] L. Bettini, D. Di Ruscio, L. Iovino, and A. Pierantonio, "Quality-driven detection and resolution of metamodel smells," *IEEE Access*, vol. 7, pp. 16 364–16 376, 2019.

[14] G. Hinkel, M. Kramer, E. Burger, M. Strittmatter, and L. Happe, "An empirical study on the perception of metamodel quality," in *2016 4th International Conference on Model-Driven Engineering and Software Development (MODELSWARD)*, 2016, pp. 145–152.

[15] M. F. van Amstel, "The right tool for the right job: Assessing model transformation quality," in *2010 IEEE 34th Annual Computer Software and Applications Conference Workshops*. IEEE, 2010, pp. 69–74.

[16] F. Jouault, F. Allilaire, J. Bézivin, and I. Kurtev, "Atl: A model transformation tool," *Science of computer programming*, vol. 72, no. 1-2, pp. 31–39, 2008.

[17] F. Fleurey, B. Baudry, P.-A. Muller, and Y. Le Traon, "Qualifying input test data for model transformations," *Software & Systems Modeling*, vol. 8, no. 2, pp. 185–203, 2009.

[18] T. J. Ostrand and M. J. Balcer, "The category-partition method for specifying and generating fuctional tests," *Communications of the ACM*, vol. 31, no. 6, pp. 676–686, 1988.

[19] P. Gómez-Abajo, E. Guerra, J. de Lara, and M. G. Merayo, "Wodel-test: a model-based framework for language-independent mutation testing," *Software and Systems Modeling*, pp. 1–27, 2020.

[20] J.-M. Mottu, S. Sen, M. Tisi, and J. Cabot, "Static analysis of model transformations for effective test generation," in *2012 IEEE 23rd International Symposium on Software Reliability Engineering*. IEEE, 2012, pp. 291–300.

[21] F. Basciani, J. D. Rocco, D. D. Ruscio, L. Iovino, and A. Pierantonio, "A customizable approach for the automated quality assessment of modelling artifacts," in *10th International Conference on the Quality of Information and Communications Technology, QUATIC 2016, Lisbon, Portugal, September 6-9, 2016*, 2016, pp. 88–93. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/QUATIC.2016.025

[22] D. S. Kolovos, R. F. Paige, and F. A. Polack, "The epsilon transformation language," in *International Conference on Theory and Practice of Model Transformations*. Springer, 2008, pp. 46–60.

[23] H. Saada, M. Huchard, C. Nebut, and H. Sahraoui, "Recovering model transformation traces using multi-objective optimization," in *2013 28th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2013, pp. 688–693.

[24] J. R. Horgan, S. London, and M. R. Lyu, "Achieving software quality with testing coverage measures," *Computer*, vol. 27, no. 9, pp. 60–69, 1994.

[25] M. D. Del Fabro, J. Bézivin, and P. Valduriez, "Weaving models with the Eclipse AMW plugin," in *Eclipse Modeling Symposium, Eclipse Summit Europe*, 2006.

[26] L. I. et al., "Github repository containing the dataset of transformations used for the validation," https://github.com/gssi/validation, 2019, [Online; accessed 10-Nov-2019].

[27] F. D. Giraldo, S. España, and O. Pastor, "Analysing the concept of quality in model-driven engineering literature: A systematic review," in *2014 IEEE Eighth International Conference on Research Challenges in Information Science (RCIS)*. IEEE, 2014, pp. 1–12.

[28] F. D. Giraldo, Á. J. Chicaiza, S. España, and Ó. Pastor, "Empirical validation of a quality framework for evaluating modelling languages in mde environments," *Software Quality Journal*, pp. 1–33, 2021.

[29] A. Vignaga, "Metrics for measuring ATL model transformations," *MaTE, Department of Computer Science, Universidad de Chile, Tech. Rep*, 2009.

[30] C. M. Gerpheide, R. R. H. Schiffelers, and A. Serebrenik, "Assessing and improving quality of QVTo model transformations," *Software Quality Journal*, vol. 24, no. 3, pp. 797–834, 2016.

[31] B. Alkhazi, C. Abid, M. Kessentini, and M. Wimmer, "On the value of quality attributes for refactoring atl model transformations: A multi-objective approach," *Information and Software Technology*, vol. 120, p. 106243, 2020.

[32] M. Saeki and H. Kaiya, "Measuring model transformation in model driven development," in *Proceedings of the CAiSE'07 Forum at the 19th International Conference on Advanced Information Systems Engineering. Volume 247 of CEUR Workshop Proceedings*. Citeseer, 2007.