From the Heterogeneity Jungle to Systematic Benchmarking*

M. Wimmer[1], G. Kappel[1], A. Kusel[2], W. Retschitzegger[2], J. Schoenboeck[1], and W. Schwinger[2]

[1] Vienna University of Technology, Austria
lastname@big.tuwien.ac.at
[2] Johannes Kepler University Linz, Austria
firstname.lastname@jku.at

Abstract. One of the key challenges in the development of model transformations is the resolution of recurring semantic and syntactic heterogeneities. Thus, we provide a systematic classification of heterogeneities building upon a feature model that makes the interconnections between them explicit. On the basis of this classification, a set of benchmark examples was derived and used to evaluate current approaches to the specification of model transformations. We found, that approaches on the conceptual level lack expressivity whereas execution level approaches lack support for reuse. Moreover, only few of the approaches evaluated provide key features such as an automatic trace model or the ability to reuse specifications by inheritance.

Keywords: Syntactic and Semantic Heterogeneities, Mapping Benchmark.

1 Introduction

With the rise of model-driven engineering (MDE), models and associated transformations for migrating, merging, or evolving models have become the main artifacts of the software engineering process [3]. One of the key challenges in this respect is the resolution of recurring heterogeneities between the corresponding metamodels (MMs) to preserve semantics, (i) at the conceptual level by means of mapping tools that provide reusable components as, for instance proposed in [7] and [21], and (ii) at the execution level by means of dedicated transformation languages [5]. Heterogeneities result from the fact that semantically similar metamodeling concepts (M2) can be defined by different meta-metamodeling concepts (M3), which leads to differently structured metamodels. As a simple example, Fig. 1 shows two MMs of fictitious, domain-specific tools that administrate publications. Whereas `Tool1` models the type of a publication by the

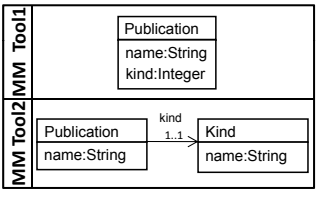

Fig. 1. Heterogenous MMs

* This work has been funded by the Austrian Science Fund under grant P21374-N13.

J. Dingel and A. Solberg (Eds.): MODELS 2010 Workshops, LNCS 6627, pp. 150–164, 2011.

attribute `Publication.kind` (e.g., conference or journal), `Tool2` represents the same semantics by using the class `Publication`, which refers to a class `Kind`, to determine the kind of publication.

Up to now, it has been unclear which kinds of heterogeneity must be resolved in model-to-model transformations. Therefore, this paper proposes a systematic classification of heterogeneities between object-oriented MMs by adapting and extending existing classifications in data and ontology engineering. The design rationale was to identify a complete set of potential points of variation between two Ecore-based MMs. This provided the basis for establishing a benchmark that allows evaluation of existing approaches with respect to their ability to resolve heterogeneities. To show the applicability of this benchmark, existing mapping tools and transformation languages from different domains were evaluated by using selected example scenarios defined for our benchmark, each posing certain challenges to the benchmarked approaches. Further example scenarios of our benchmark can be found in [22] and on our project web page[1].

The remainder of this paper is structured as follows. Section 2 presents the identified points of variation of meta-metamodels and gives a first overview of our classification. The exemplary benchmark scenarios together with their challenges and how existing approaches deal with them are discussed in Section 3. Lessons learned are summarized in Section 4. Finally, related work is referred to in Section 5, and Section 6 reports on future work.

2 Systematic Classification of Heterogeneities

According to a substantial body of literature [2,4,9,12,13,14,15,18,20], heterogeneities can be divided into two main classes: (i) *syntactic heterogeneities*, i.e., differences with respect to *how* something is represented by a MM and (ii) *semantic heterogeneities*, i.e., differences with respect to *what* is represented by a MM.

Syntactic Heterogeneities. Syntactic heterogeneities result from the fact that semantically similar concepts can be defined by different metamodeling concepts, which leads to differently structured metamodels. To obtain a systematic classification of different kinds of syntactic heterogeneity, we investigated potential points of variation between two Ecore[2]-based metamodels. Fig. 2 depicts the relevant part of the Ecore meta-metamodel potentially causing syntactic heterogeneities, and omits all the Ecore concepts which

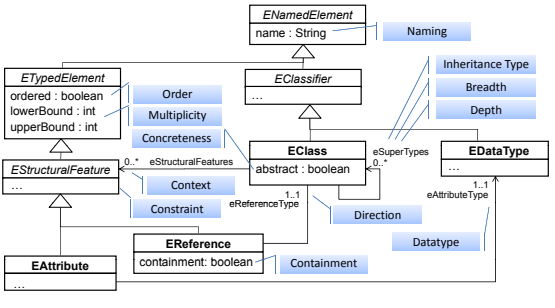

Fig. 2. Variation Points in Ecore-based MMs

[1] `www.modeltransformation.net`

[2] `http://www.eclipse.org/modeling/emf/`

Table 1. Common Core Concepts in Different Domains

Common Core Concepts	Ecore	XML Schema	OWL
Class	EClass	<xs:complexType>	<owl:Class>
Attribute	EAttribute	<xs:attribute>	<owl:DatatypeProperty>
Reference	EReference	<xs:key>, <xs:keyRef>	<owl:ObjectProperty>
Inheritance	eSuperTypes	<xs:extension base>	<rdfs:subClassOf>

are used merely for Java code generation in the EMF framework. It must be emphasized at this point that the core concepts of Ecore resemble the fundamental ingredients of semantic data models [11], which are also prevalent in other domains such as data and ontology engineering (as depicted in Table 1). Hence, the following findings apply to a broader field.

Based on this design rationale, we introduce a classification of heterogeneities (cf. Fig. 3). It is expressed by means of the feature model formalism [6], which allows us to identify clearly the interconnections between the different kinds of heterogeneity. We distinguish two types of *syntactic heterogeneity*: simple *naming differences* (i.e., differences in the values of the `name` attribute of `ENamedElement`: cf. Fig. 2) and more challenging *structural differences*. Although names play an important role when deriving the semantics of a concept, the semantics cannot be inferred automatically, which leads to the synonym and homonym problem. With respect to structural differences, two main cases can be distinguished: *core concept differences* and *inheritance differences*. The former occur due to different usage of classes, attributes, and references (represented by `C`, `A`, and `R` in Fig. 3) and can be further divided into heterogeneities between (a) the same and (b) different metamodeling concepts. Two main differences may emerge in case (a) – either the concepts exhibit different attribute/reference settings (cf. Fig 2) or a different number of concepts has been used in the MMs to express the same semantic concept (cf. *Source-Target-Concept Cardinality* in Fig. 3). An example of the first case would be that one of two `EClasses` used is defined as `abstract`, which leads to a *concreteness heterogeneity*. An example of the second case is that in the left hand side (LHS) MM, two `EAttributes`, `firstName` and `lastName`, are used whereas in the right hand side (RHS) MM, this information is contained in just one `EAttribute`: `name`. Concerning case (b), heterogeneities are derived by systematically combining the identified core concepts. For instance, an `EAttribute` in the LHS MM is represented by an `EClass` in the RHS MM (cf. example in Fig. 1). Finally, heterogeneities may not only be caused by the concepts of classes, attributes, and references but also by the concept of inheritance. In this respect, we distinguish between heterogeneities that may occur although both MMs use inheritance (cf. *"same metamodeling concept inheritance"* in Fig. 3) and heterogeneities that occur if only one MM makes use of inheritance (cf. *"different metamodeling concept inheritance"* in Fig. 3).

Semantic Heterogeneities. Two main cases of semantic heterogeneity can be distinguished: (i) differences in the *number of valid instances* and (ii) differences in the *interpretation of the instance values* [12]. In case (i), all the set-theoretic relationships may occur as modeled by the corresponding subfeatures. In case (ii), a variety of modifications of the values may be necessary

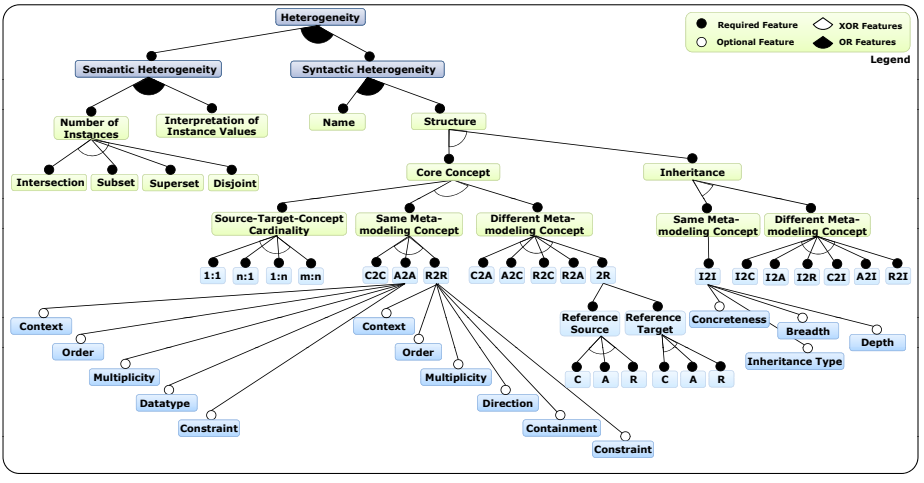

Fig. 3. Heterogeneity Feature Model

to translate LHS MM values into RHS MM values such that the values conform to the *interpretation* of the RHS MM. Thus, semantic heterogeneities cannot be derived from the syntax (since in both cases the MMs can be represented syntactically in the same way) but only by incorporating *interpretation*, i.e., by assigning meaning to each piece of data [10]. For further details about the classification the reader is referred to [22].

3 Benchmark Examples Applied

The proposed classification was then used to derive appropriate benchmark examples. Since the classification makes the interconnections between heterogeneities explicit, a systematic set of benchmark examples, i.e., a set that fully covers the feature model, can be built. Each benchmark example is characterized by a description, source and target metamodels and corresponding models. To aid comprehension, examples using ontological concepts were preferred over those using linguistic concepts. Below, we present three of the proposed examples that we used to evaluate the ability of mapping tools and transformation languages to resolve certain heterogeneities. Each example is a representative of a main branch in the feature model: (i) core concepts with same metamodeling concept heterogeneities, (ii) core concept with different metamodeling concept heterogeneities and (iii) inheritance heterogeneities.

Evaluated Approaches. The benchmark examples were applied to a carefully selected bundle of approaches: at the conceptual level, three mapping tools from different domains and, at the execution level, two dedicated model transformation languages. Among the mapping tools were *AMW* [7] from the area of model engineering, the commercial tool *MapForce*[3] from data engineering, and

[3] `http://www.altova.com/de/mapforce.html`

MAFRA[4] from ontology engineering. *AMW* allows the definition of so-called weaving links between Ecore-based MMs to form a mapping definition which can be transformed into executable *ATL* code. In contrast, *MapForce* allows mapping definitions between diverse schema languages, for instance, relational or XML schemas. For executing the specified mapping definitions, several target languages such as Java and XSLT are supported. Finally, *MAFRA* supports for mappings between RDF- and OWL-based ontologies and XML schemas which are directly executed within the tool. Among the transformation languages evaluated were *ATL*[5], a representative of hybrid rule-based transformation languages, and *AGG*[6], a declarative graph-based transformation language. The results of the comparison are summarized in Table 2 and described in detail below.

3.1 Benchmark Example 1

The first benchmark example belongs to the category of core concept heterogeneities between the same MM concepts (cf. Fig. 3), and poses four main challenges, as detailed below (cf. Fig. 4). Since the overall goal of all our transformations is to minimize *information loss* and to produce only *valid instances*, instance `P2` remained in the RHS although it does not reference any journal publication in the LHS model. Interestingly, the RHS MM in this example is more restrictive than the LHS MM, since the `EAttribute` `Prof.bornIn` always requires a value, and since each instance of `Prof` requires at least one link to a journal publication. Since these restrictions do not exist in the LHS MM, instances of it may break them. Therefore, some resolution strategy is needed – either by auto-generating values or by incorporating user-interaction in order to produce valid instances of the RHS MM.

Challenge 1: A2A, Multiplicity Difference, Datatype Difference. In our example, this challenge arises between the `EAttributes` `Professor.dateOfBirth` and `Prof.bornIn`. The main challenge is to extract the year of birth as an integer value from the LHS date structure. In the absence of a date either (i) a null-value (with semantics *exists but not known*, leading to an invalid target model), or (ii) a (user- or auto-generated) value requiring a corresponding *function* is produced. All evaluated approaches were able to meet this challenge, although the specification effort varied. For example, in *MapForce* dedicated components such as `substitute-missing` (cf. Fig. 5 (a)) are available, whereas in the other tools the function must be defined from scratch.

Challenge 2: Semantic Heterogeneity, A2A. The *second challenge* exhibits a *semantic heterogeneity* between the `EAttributes` `Professor.salary` and `Prof.salary`, since `Professor.salary` is encoded in dollars, whereas `Prof.salary` is encoded in euros, i.e., there is a difference in the interpretation of the values. A conversion of values from dollars to euros must thus be realized in a *function*. This imposes requirements similar to those in the first challenge, and

[4] `http://mafra-toolkit.sourceforge.net`
[5] `http://www.eclipse.org/atl`
[6] `http://user.cs.tu-berlin.de/~gragra/agg`

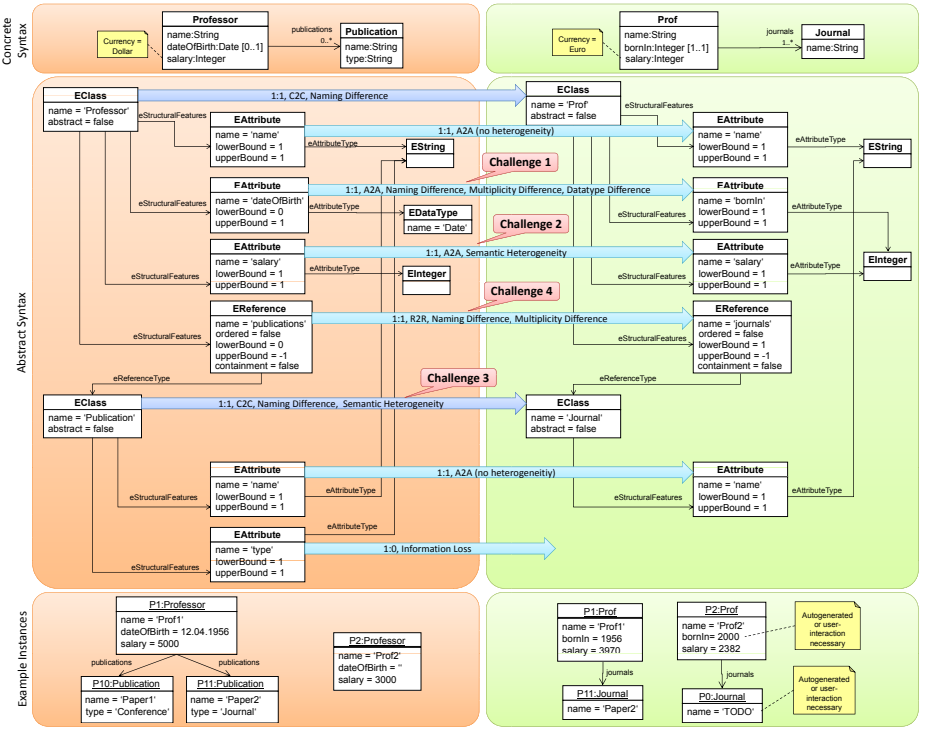

Fig. 4. Benchmark Example 1 – Heterogeneities between same MM concepts

the evaluation therefore results in similar findings. The exemplary solution of this challenge in *MapForce* is shown in Fig. 5 (a).

Challenge 3: Semantic Heterogeneity, C2C. The *third challenge* again includes a *semantic heterogeneity* – but this time a difference in the number of valid instances, since only journal instances should be transformed. Resolving the heterogeneity requires a corresponding *condition*, that identifies instances that remain valid in the context of the RHS `EClass`. All approaches were able to achieve this. The exemplary solution of this challenge in *AMW* is shown in Fig. 5 (b).

Challenge 4: R2R, Multiplicity Difference. Finally, the *fourth challenge* consists of a multiplicity difference between the `EReferences Professor.publications` and `Prof.journals`. Since challenge 3 requires transformation only of journal instances, the first sub-challenge here is to identify links that do not refer to journal instances. Ideally, this should be achieved automatically by a built-in trace model that keeps track of which source elements have been used to create certain target elements. Moreover, since the goal is to generate only valid target instances, the second sub-challenge is to generate journals and link them correctly (instead of generating null values with semantics *does not exist* when a professor does not have any). All approaches were able to resolve the heterogeneity of the first sub-challenge. However, the effort needed differed, since

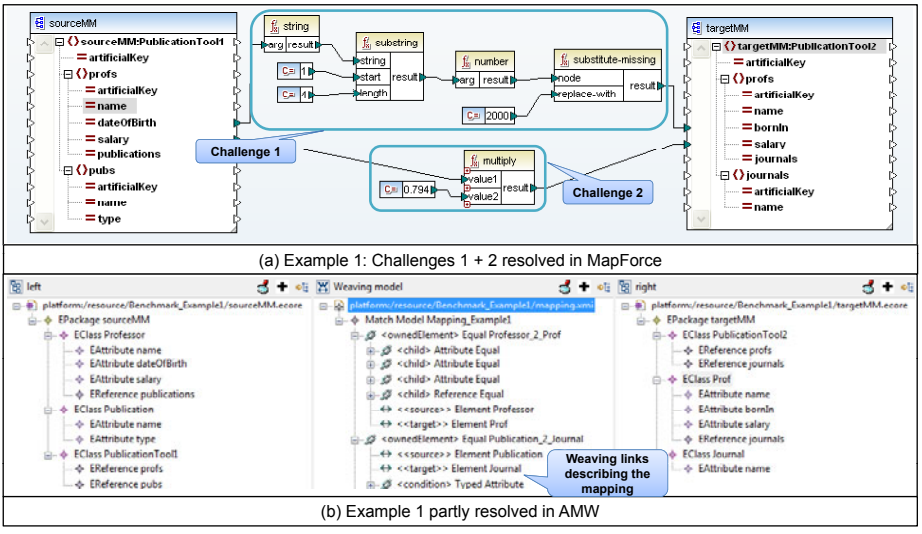

Fig. 5. Exemplary Solutions for Benchmark Example 1

the condition required for filtering journal instances (as in challenge 3) had to be duplicated in all mapping approaches due to insufficient trace model support. As for the second sub-challenge, *AMW* and *MAFRA* were not able to link newly generated objects due to insufficient trace model support. Although *MapForce* does not support a trace model, it resolved this example, since only one journal object with a fixed key value had to be generated, and this can be referred to by the foreign key. Both, *ATL* and *AGG* were able to resolve this heterogeneity using their trace models. *ATL* provides a built-in trace model which can be queried (`resolveTemp` mechanism), whereas in *AGG* the trace model must be maintained manually.

In summary, the fourth challenge appeared to be the most problematic one in this example for the approaches evaluated.

3.2 Benchmark Example 2

The second benchmark example belongs to the category of core concept heterogeneities when using different metamodeling concepts (cf. Fig. 3) and poses two main challenges. The example instances reveal that the intention is to create a `Kind` object only for distinct values of the attribute `Publication.kind`. Therefore, the RHS model contains only a single `Kind` object named `Journal` (cf. `K1` in Fig. 6), which is referenced by the `Publication` objects `P1` and `P2`.

Challenge 1: A2C. The first challenge in this benchmark example is the generation of `Kind` objects for distinct values of the `kind` attribute. A trace model is required to keep track of whether an object has already been created for a value. Since no explicit trace model is available in *AMW* and *MAFRA*, they were not able to resolve this heterogeneity. Although *MapForce* also does

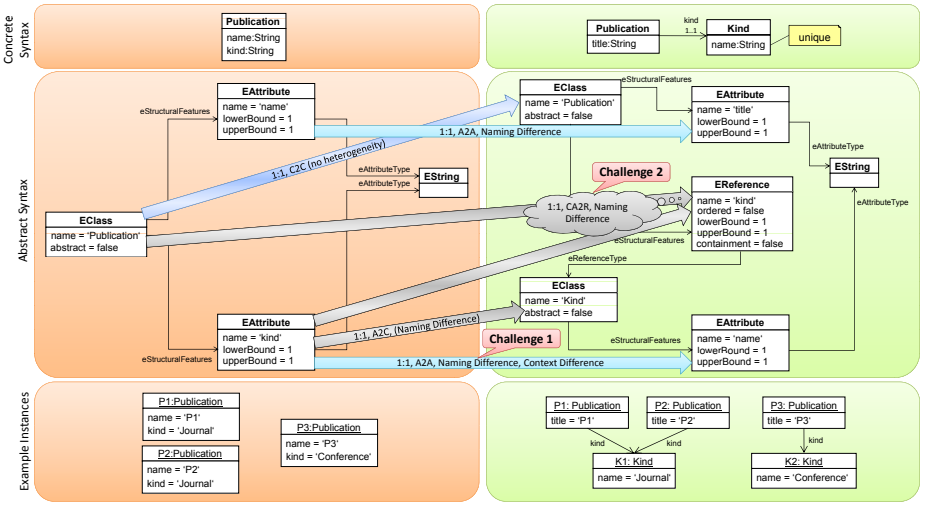

Fig. 6. Benchmark Example 2 – Different Metamodeling Concept Heterogeneities

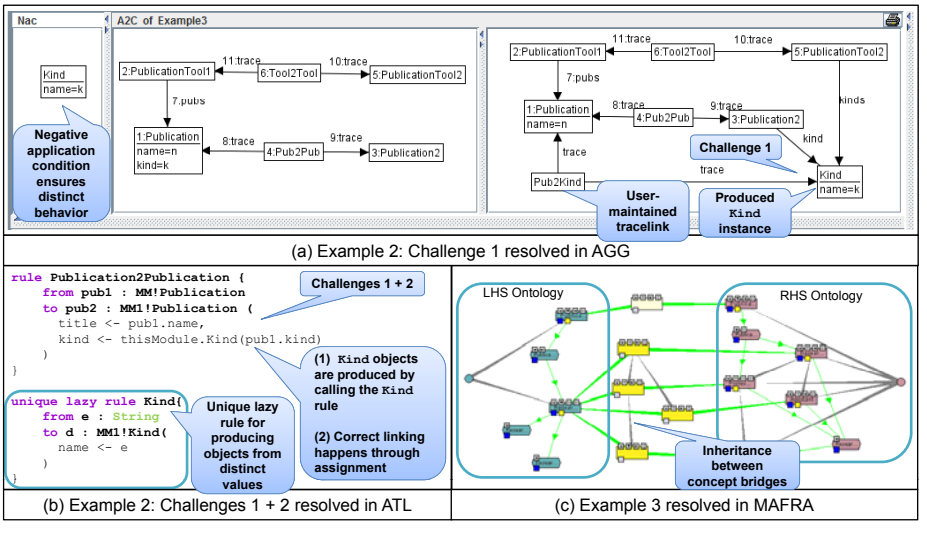

Fig. 7. Exemplary Solutions for Benchmark Examples 2 and 3

not provide explicit trace information, it offers a dedicated `distinct-values` component, which produces target elements for distinct input values only. In *ATL* a so-called *unique lazy rule* can be applied (cf. Fig. 7 (b)). Using the built-in trace model, this type of rule always generates and returns the same target object. *AGG* offers no dedicated support: the heterogeneity must be resolved by using user-defined graph transformation rules and a negative application condition that prevents multiple creation of `Kind` objects (cf. Fig. 7 (a)).

Challenge 2: CA2R. In addition to creating objects based on distinct LHS values, the second challenge in this example is to correctly link the generated target elements. Establishing such links requires information about the relationships between the concepts to be linked in the LHS model. In the LHS MM of this example, the source of the `EReference Publication.kind` is represented by the `EClass Publication` and the target of the `EReference` by the `EAttribute Publication.kind`. Therefore, this heterogeneity is classified as ***C**(lass)**A**(ttribute)**2R**(eference)*. To obtain the information needed to establish the links, the approaches must again support queries to the trace model. Since *AMW* and *MAFRA* could not cope with challenge 1, they were also not able to resolve this heterogeneity. *MapForce* was also unable to resolve this kind of heterogeneity, since the internal trace model of the `distinct-values` component cannot be queried. Although the trace model produced by the *unique lazy rule* in *ATL* also cannot be queried, the elements produced can be linked correctly by calling the *unique lazy rule* in the assignment (cf. Fig. 7 (b)). In *AGG* the user-maintained trace model can be used to resolve this heterogeneity.

In summary, the mapping tools evaluated provide only limited support for resolution of the various metamodeling concept heterogeneities. Detailed knowledge of the transformation languages is required when using them to resolve heterogeneities, which further emphasizes the need for direct support by dedicated components.

3.3 Benchmark Example 3

The third benchmark example belongs to the category of inheritance heterogeneities with different metamodeling concepts (cf. Fig. 3) and poses one challenge. As the example instances show (cf. Fig. 8), the type of an LHS

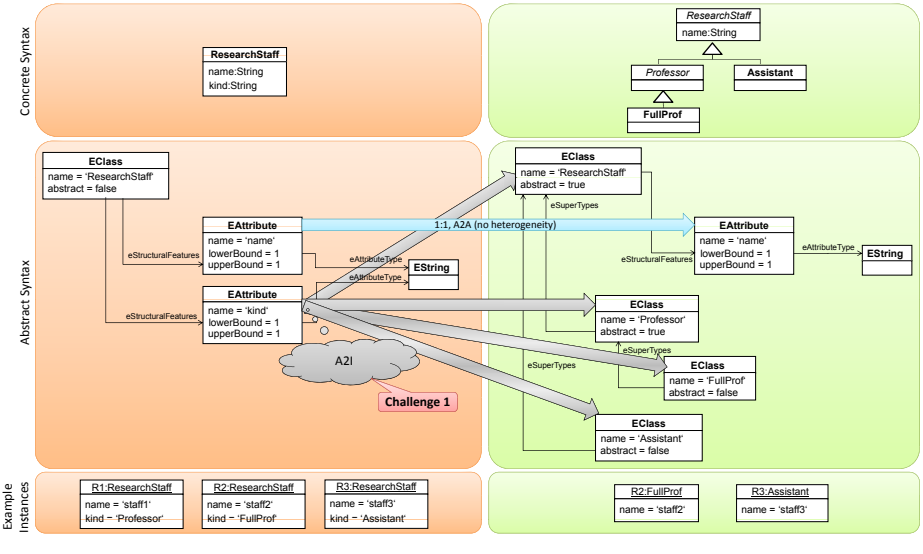

Fig. 8. Benchmark Example 3 – Different Metamodeling Concept Heterogeneities

`ResearchStaff` object is identified by the value of its `EAttribute Research-Staff.kind`, whereas the RHS MM provides an explicit type hierarchy. Thus, the problem may arise, that the `EAttribute` of the LHS MM contains values that do not correspond to any (concrete) `EClass` in the RHS MM. This is the case in the example with the instance `R1`, since the corresponding `EClass Professor` in the RHS MM is abstract and can thus not be instantiated, which causes information loss.

Challenge 1: A2I. To resolve this heterogeneity, objects must be filtered by using a certain attribute value and should provide means to deal with inheritance in order to reduce the specification overhead. With the exception of *MapForce*, which cannot display correctly XML schemas that make use of type derivation, all mapping tools were able to resolve this heterogeneity, although no dedicated components are available. In contrast to *AMW*, *MAFRA* allows for inheritance between mappings and thus reduces specification overhead (cf. Fig. 7 (c)). Of the transformation languages evaluated, only *ATL* supports inheritance between rules, whereas *AGG* does not.

In summary, although the resolution of this heterogeneity can be achieved in all approaches except in *MapForce*, no approach provides dedicated support. The approaches that enable resolution can be further divided into those supporting inheritance (*ATL*, *MAFRA*), which allow reuse in specifications, and those not supporting inheritance (*AMW*, *AGG*), which require duplication of parts of the specification.

Table 2. Comparison of Approaches

Challenges		Model Engineering			Data Engineering	Ontology Engineering
		AMW	**ATL**	**AGG**	**MapForce**	**MAFRA**
Example 1						
1	A2A, Multiplicity and Datatype Difference	~ User-defined extension	~ Conditional assignment with function	~ User-defined Java-function	✓ Function library (value translations and substitute-missing component for default-value generation)	~ Property bridge with a user-defined service
2	A2A, Semantic Heterogeneity	~ User-defined extension	~ User-defined Function	~ User-defined Java-function	✓ Function library available for diverse value translations	~ Property bridge with a user-defined service
3	C2C, Semantic Heterogeneity	✓ Equivalence component with condition	~ Condition	~ Condition	~ Condition	✓ Concept bridge with condition
4	R2R, Multiplicity Difference	✗ No reference to newly generated objects possible	~ Query of the trace model	~ User-maintained tracelinks	~ Has to be simulated by the generation of foreign-key-values according to a query	✗ No reference to newly generated objects possible
Example 2						
1	A2C	✗ No distinct semantics supported	~ Unique lazy rule	~ User-maintained tracelinks	✓ Distinct-values component	✗ No distinct semantics supported
2	CA2R	✗ No tracemodel available	~ Query of the trace model	~ User-maintained tracelinks	✗ No tracemodel available	✗ No tracemodel available
Example 3						
1	A2I	~ No inheritance support - simulation by code duplication	~ Inheriting rules	~ No inheritance support - simulation by code duplication	✗ No support for inheritance	~ Inheriting concept bridges

✓ supported ~ resolvable (no dedicated support) ✗ non-resolvable

4 Lessons Learned

In this section, we present the lessons learned from applying our examples.

Absence of Trace Model Limits Applicability. Current mapping tools in the area of data engineering typically rely on the specification of simple correspondences between source and target elements, which may be refined by conditions or functions. However, these correspondences do not offer trace information, which would support the definition of dependent mappings. For instance, a value mapping always occurs in the context of a certain object mapping and is thus dependent on the element mapping. This deficiency leads to less expressive mapping specifications, as also discussed in [16]. The developers of mapping tools in the area of ontology engineering and model engineering recognized this need and thus implemented dependent mappings. However, simple dependencies between mappings, for instance, composition of mappings, are still insufficient, which leads to the problems, for example, in an `A2C` heterogeneity, in which explicit queries to the trace model are needed. Transformation languages can be divided into approaches providing automatic trace information, as in *ATL*, and approaches requiring manual generation of trace information, as in *AGG*. *ATL* provides trace information only for the declarative parts (`matched rules`) and not for the imperative parts of the language. Finally, a user-specified trace model leaves the entire tedious and error-prone process of setting up the trace information correctly to the transformation designer.

Transformation Languages Lack Reuse Facilities. Transformation languages such as *ATL* and *AGG* provide the expressivity to overcome the heterogeneities identified in our examples. Nevertheless, they lack adequate reuse facilities, which forces the transformation designer to respecify the resolution of recurring heterogeneities over and over. Especially in complex scenarios (e.g., when generating new target elements, as in the first example, or when dealing with unequal concept heterogeneities, as in the second example), the transformation designer must handle low-level intricacies of the transformation language. In order to avoid this tedious and error-prone task, transformation languages should provide idioms that resolve these structural heterogeneities, for instance, predefined, parameterizable rules in *ATL* or in *AGG*. A fact that hinders the provision of such predefined components is that transformation rules are based on the specific types defined in the corresponding metamodels. Thus, a notion of generic transformations which resembles the concept of templates in C++ or generics in Java is required.

Lack of Inheritance Support Encourages Code Duplication. Inheritance, which is heavily used in metamodels, supports the reuse of attribute and reference definitions. Thus, when a mapping is specified between subclasses, then it should be able to reuse attribute and reference mappings of mappings between superclasses; i.e., inheritance between mappings should be supported. The same holds true for transformation languages. Otherwise, duplicated mapping definitions or transformation rules induce both, a bigger specification overhead and maintenance problems in the future, as is the case in *AGG*.

Mapping Tools Struggle with Function/Condition Definitions. Mapping tools have the main advantage of providing predefined components for the resolution of heterogeneities, but the definition of functions and conditions poses a major problem. Each mapping tool provides a specific basic set of components. For instance, *MapForce* provides a library of low-level functions such as string conversion functions. However, such a library is naturally never complete, which leads to limitations in expressivity. Thus, incorporating an expressive language with which the transformation designer is familiar could resolve this problem.

Mapping Tools Lack Adequate Extension Mechanisms. Since mapping tools struggle with resolving certain kinds of heterogeneity, an adequate extension mechanism that allows addition of user-defined components should be offered. *MapForce* supports user-defined components but only on the basis of predefined ones. Although this enhances the scalability of the approach by composing several low-level components, expressivity is not increased. In contrast, both *AMW* and *MAFRA* allow increasing expressivity by user-defined components but require heavyweight programmatic extensions. In *AMW*, both, the metamodel describing the set of predefined components and the transformation generating *ATL* code from a mapping specification must be extended. *MAFRA* supports new components, but they must be coded manually in so-called `user-defined services`.

Mapping Tools Lack Comprehensive Validation Support. A major advantage of describing model transformations at a conceptual level by means of mapping tools is that comprehensive validations can be done at design time. To verify that the components are configured correctly, *structural validations* examine the required input and output parameters and *metamodel-based validations* check the interpretation of the mapped metamodels. For instance, a reference is only mapped correctly if both its source and target class have also been mapped. *MapForce* and *MAFRA* support only structural validations. *AMW* does not support validation at all, which results in potentially erroneous *ATL* code.

5 Related Work

Two threads of related work are considered: First, we compare our feature-based classification to existing classifications. Second, we relate the mapping benchmark to existing mapping benchmarks. We start with examining the most closely related area, model engineering, and then proceed to the more widely related areas of data engineering and ontology engineering.

5.1 Heterogeneity Classifications

Model Engineering. Although model transformations, and thus the resolution of heterogeneities between MMs, play a vital role in MDE, to the best of our knowledge no dedicated survey exists that examines potential heterogeneities.

Data Engineering. In the area of data engineering, in contrast, extensive literature exists, over decades, highlighting various aspects of heterogeneities in the context of database schemata. Batini et al. [2] presented a first classification of

semantic and structural heterogeneities that arise when two different schemas are integrated. Kim et al. [13] introduced a systematic classification of possible variations in an SQL statement, detailing *Table-Table* and *Attribute-Attribute* heterogeneities (e.g., with respect to cardinalities). The classification of Kashyap et al. [12] provides a broad overview of potential heterogeneities in a data integration scenario with semantic heterogeneities and conflicts that occurr between the same modeling concepts. Blaha et al. [4] described patterns that resolve syntactic heterogeneities, both between the same and different MM concepts. Finally, the classification of Härder [9] and Legler [15] presented a systematic approach to attribute mappings by combining attribute correspondences with potential cardinalities.

Ontology Engineering. In ontology engineering, both pattern collections and classifications exist. A pattern collection by Scharffe et al. in [17] presented correspondence patterns for ontology alignments, but on a rather coarse-grained level. For instance, their conditional patterns dealing with attribute differences and transformation patterns deal only vaguely with different metamodeling concept heterogeneities. Visser et al. [20] and Klein [14] provided classifications in the form of comprehensive lists of semantic heterogeneities but neglected syntactic heterogeneities.

In summary, although there are several classifications available, none focuses explicitly on the domain of MDE. Since the benchmarks in the area of data engineering base on the relational data model, they do not include potential heterogeneities stemming from the explicit concepts of references and inheritance in object-oriented metamodels. Although in ontology engineering references and inheritance are explicit concepts, their interest is to resolve semantic heterogeneities rather than syntactic heterogeneities. Finally, current classifications fail to explicate how types of heterogeneity relate to each other. We formalized these relationships in a feature model.

5.2 Mapping Benchmarks

Model Engineering. To the best of our knowledge, no benchmark for mapping systems in the area of MDE exists. However, a benchmark for evaluating the execution performance of graph transformations [19] has been proposed.

Data Engineering. In the area of data engineering Alexe et. al. [1] proposed a first benchmark for mapping systems that focuses on resolving syntactic and semantic heterogeneities in information integration. Although the benchmark provides a first set of mapping scenarios, it remains unclear how the scenarios were obtained and whether they provide full coverage in terms of expressivity. Even though XQuery expressions are given to define the semantics, some of the XQuery functions assume the availability of custom functions which are not provided. Since RHS models are also not given, it is hard to know the actual outcome of the transformation. A further benchmark called THALIA was presented by Hammer et. al [8], which provides researchers with a collection of twelve benchmark queries expressed in XQuery. They focus on the resolution of syntactic and semantic heterogeneities in an information integration scenario. For each query a so-called

reference schema (i.e., global schema) and a *challenge schema* (i.e., the schema to be integrated) are provided together with corresponding instances. Although the authors claim to provide a systematic classification of semantic and syntactic heterogeneities resulting in the queries, the rational behind the systematic is not explained further.

Ontology Engineering. In ontology engineering, no dedicated mapping benchmark exists. However, there have been efforts to evaluate matching tools, i.e., tools for automatically discovering alignments between ontologies, which resulted in an ontology *matching benchmark*[7]. Although the goal of the evaluation is different, the examples could also be of interest for a dedicated mapping benchmark.

In summary, although both benchmarks from the area of data engineering provide useful scenarios in the context of XML, they do not provide a systematic classification that results in a systematic set of benchmark examples for evaluating the expressivity of a mapping tool.

6 Conclusion and Future Work

In this paper we have introduced a systematic classification of heterogeneities between Ecore-based MMs. This classification can also be applied to other domains, that use the same core concepts on which this classification is based, i.e., classes, attributes, references and inheritance. Furthermore, three of the proposed benchmark examples were used to evaluate mapping tools from diverse engineering domains and to compare solutions realized with the transformation languages *ATL* and *AGG*. Further work includes the completion of the benchmark examples to fully cover the classification and the evaluation of further approaches.

References

1. Alexe, B., Tan, W.-C., Velegrakis, Y.: STBenchmark: Towards a Benchmark for Mapping Systems. VLDB Endow 1(1), 230–244 (2008)
2. Batini, C., Lenzerini, M., Navathe, S.B.: A Comparative Analysis of Methodologies for Database Schema Integration. ACM Comp. Surv. 18(4), 323–364 (1986)
3. Bézivin, J.: On the Unification Power of Models. Journal on SoSyM 4(2), 31 (2005)
4. Blaha, M., Premerlani, W.: A catalog of object model transformations. In: Proc. of the 3rd Working Conf. on Reverse Engineering (WCRE 1996), pp. 87–96 (1996)
5. Czarnecki, K., Helsen, S.: Feature-based Survey of Model Transformation Approaches. IBM Systems Journal 45(3), 621–645 (2006)
6. Czarnecki, K., Helsen, S., Eisenecker, U.: Staged Configuration Using Feature Models. In: Proc. of Third Software Product Line Conf., pp. 266–283 (2004)
7. Del Fabro, M., Bézivin, J., Valduriez, P.: Model-driven Tool Interoperability: an Application in Bug Tracking 1. In: Proc. of the 5th Int. Conf. on Ontologies, DataBases, and Applications of Semantics (ODBASE 2006), pp. 863–881 (2006)

[7] `http://oaei.ontologymatching.org/2010/`

8. Hammer, J., Stonebraker, M., Topsakal, O.: THALIA: Test harness for the assessment of legacy information integration approaches. In: Proc. of the Int. Conf. on Data Engineering (ICDE 2005), pp. 485–486 (2005)
9. Härder, T., Sauter, G., Thomas, J.: The intrinsic problems of structural heterogeneity and an approach to their solution. The VLDB Journal 8(1), 25–43 (1999)
10. Harel, D., Rumpe, B.: Meaningful modeling: What's the semantics of "semantics"? IEEE Computer 37, 64–72 (2004)
11. Hull, R., King, R.: Semantic Database Modeling: Survey, Applications, and Research Issues. ACM Comp. Surv. 19(3), 201–260 (1987)
12. Kashyap, V., Sheth, A.: Semantic and schematic similarities between database objects: A context-based approach. The VLDB Journal 5(4), 276–304 (1996)
13. Kim, W., Seo, J.: Classifying Schematic and Data Heterogeneity in Multidatabase Systems. Computer 24(12), 12–18 (1991)
14. Klein, M.: Combining and relating ontologies: an analysis of problems and solutions. In: Proc. of Workshop on Ontologies and Information Sharing (IJCAI 2001), pp. 53–62 (2001)
15. Legler, F., Naumann, F.: A Classification of Schema Mappings and Analysis of Mapping Tools. In: Proc. of the GI-Fachtagung für Datenbanksysteme in Business, Technologie und Web (BTW 2007), pp. 449–464 (2007)
16. Raffio, A., Braga, D., Ceri, S., Papotti, P., Hernández, M.A.: Clip: a visual language for explicit schema mappings. In: Proc. of the 24th Int. Conf. on Data Engineering (ICDE 2008), pp. 30–39 (2008)
17. Scharffe, F., Fensel, D.: Correspondence Patterns for Ontology Alignment. In: Gangemi, A., Euzenat, J. (eds.) EKAW 2008. LNCS (LNAI), vol. 5268, pp. 83–92. Springer, Heidelberg (2008)
18. Sheth, A.P., Larson, J.A.: Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases. ACM Comput. Surv. 22(3), 183–236 (1990)
19. Varro, G., Schürr, A., Varro, D.: Benchmarking for graph transformation. In: Proc. of the 2005 IEEE Symposium on Visual Languages and Human-Centric Computing (VLHCC 2005), pp. 79–88 (2005)
20. Visser, P.R.S., Jones, D.M., Bench-Capon, T.J.M., Shave, M.J.R.: An analysis of ontological mismatches: Heterogeneity versus interoperability. In: Proc. of AAAI 1997 Spring Symposium on Ontological Engineering, pp. 164–172 (1997)
21. Wimmer, M., Kappel, G., Kusel, A., Retschitzegger, W., Schoenboeck, J., Schwinger, W.: Surviving the Heterogeneity Jungle with Composite Mapping Operators. In: Tratt, L., Gogolla, M. (eds.) ICMT 2010. LNCS, vol. 6142, pp. 260–275. Springer, Heidelberg (2010)
22. Wimmer, M., Kappel, G., Kusel, A., Retschitzegger, W., Schönböck, J., Schwinger, W.: Towards an Expressivity Benchmark for Mappings based on a Systematic Classification of Heterogeneities. In: Proc. of the First Int. Workshop on Model-Driven Interoperability (MDI 2010) @ MoDELS 2010, pp. 32–41 (2010)