Automatic Data Transformation—Breaching the Walled Gardens of Social Network Platforms

Martin Wischenbart[2] **Stefan Mitsch**[1] **Elisabeth Kapsammer**[1]
Angelika Kusel[1] **Stephan Lechner**[3] **Birgit Pröll**[1] **Werner Retschitzegger**[1]
Johannes Schönböck[2] **Wieland Schwinger**[1] **Manuel Wimmer**[2]

[1] Department of Cooperative Information Systems
Johannes Kepler University, Linz, Austria, Email: {firstname.lastname}@jku.at

[2] Business Informatics Group, Institute of Software Technology and Interactive Systems
Vienna University of Technology, Vienna, Austria, Email: {lastname}@big.tuwien.ac.at

[3] Netural GmbH, Linz, Austria, Email: s.lechner@netural.com

Abstract

Although many social networks on the Web allow access via dedicated APIs, the *extraction of instance data* for further use by applications is often a tedious task. As a result, instance data transformation to *Linked Data* in the form of OWL, as well as the *integration* with other data sources, are aggravated. To alleviate these problems, this paper proposes a *model-driven approach* to overcome data model heterogeneity by *automatically* transforming schemas and instance data from JSON to OWL/XML, utilizing the semantic features of OWL and Jena inference rules. We present a prototypical implementation on the basis of the *Eclipse Modeling Framework*. This implementation is applied and evaluated on data from *Facebook*, *Google+*, and *LinkedIn*. Finally, we provide prospects for semantic integration and managing evolution, as well as a discussion of how to *generalize* our approach to other domains and transformations between *arbitrary technical spaces*.

Keywords: Schema and instance transformation, data model heterogeneity, model driven approach, social network data integration, JSON to RDF & OWL

1 Introduction

In recent years, online social networks have gained great popularity amongst internet users. These networks serve different purposes and communities, for instance, socializing on Facebook or Google+, or establishing professional networks in LinkedIn[1] (Kim et al. 2010). Since users are members of several social networks, integrated profiles from multiple networks are desired to achieve a *comprehensive view* on users, which would, for instance, increase the quality of personalized recommendations (Abel et al. 2011), or support users' search activities (Bozzon et al. 2012). In our research project TheHiddenU[2] we try to build such comprehensive user profiles in OWL, enriching them with machine learning and information extraction methods, for use in recommender applications. On our platform, non-experts shall express transformations of social network data to their preferred target format. Furthermore, prime goals are to make the system transparent and trustworthy, by (i) providing *provenance information* whenever demanded, and (ii) by respecting and supporting users' *privacy needs* at all times.

For building comprehensive user profiles, in principle, existing user models may be reused as target schemas for the user profiles to be integrated, such as GRAPPLE (Aroyo & Houben 2010) or others (Viviani et al. 2010). However, in contrast to social networks, these approaches often base on common ontologies expressed in OWL (FOAF, etc.[3]). Social networks would benefit from using such ontologies, Semantic Web technologies, and Linked Data, for instance, to solve portability issues and to enable data reuse (Razmerita et al. 2009). Thus, to ultimately extract Linked Data and breach the *walled gardens of social networks*, the resulting *difference in technical spaces* demands, as depicted in Fig. 1, that we first tackle *technical, syntactic* (cf. ①), and *data model* heterogeneity (cf. ②), before *structural and semantic* heterogeneity (cf. ③) can be resolved, to finally build integrated user profiles that are (i) complete, concise, and consistent (Bleiholder & Naumann 2009), (ii) within aligned ontologies (Parundekar et al. 2010), and (iii) in the form of Linked Data (Heath & Bizer 2011). Dividing these required transformation steps helps to cope with evolution, and facilitates reuse across data sources, because changes are kept local (e.g., JSON replaces XML, while semantic mappings remain unchanged).

For handling all these kinds of heterogeneities, existing tools in the research area of the Semantic Web, such as Virtuoso[4], Triplr[5], and Aperture[6], can be extended with components for user profile extraction. Typically, these components must be configured (i) on the *schema level* with respect to the *target schema*, which often is an existing ontology (e. g., FOAF) complemented with a manually created one, and (ii) on the *instance level* with respect to *transformation spec-*

This work has been funded by the Austrian Federal Ministry of Transport, Innovation, and Technology (BMVIT) under grant FIT-IT 825070 and BRIDGE 832160.

[1] www.facebook.com, plus.google.com, www.linkedin.com
[2] www.social-nexus.net
[3] FOAF: foaf-project.org, SIOC: sioc-project.org, Relationship Ontology: vocab.org/relationship
[4] virtuoso.openlinksw.com
[5] triplr.org
[6] aperture.sourceforge.net

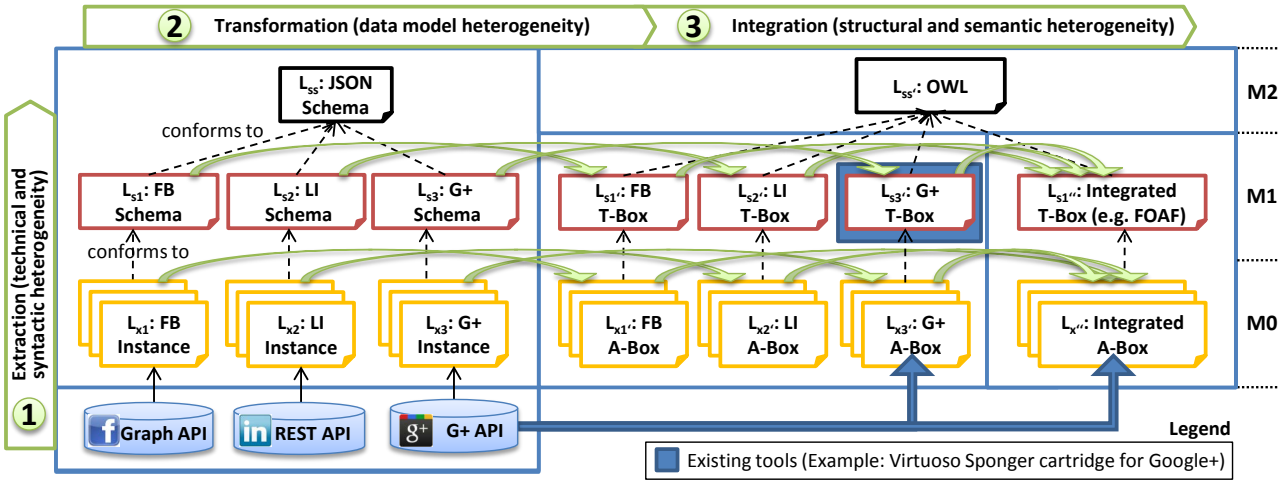

Figure 1: Overview of heterogeneities during user profile integration

ifications between the source schema of the extracted data and the desired target schema, in order to transform *instances*. Often, such components *mix* the resolution of data model, structural, and semantic heterogeneities in a single transformation step, thereby *aggravating maintenance and modifications*. Furthermore, in the face of new social networks arising frequently and *evolution* of existing ones (i. e., changes of schemas and APIs), manual creation of schemas and transformation specifications for all possible data sources is not an adequate option, not least since schemas may be extensive in size.

Existing automated transformation approaches, for instance, by (Atzeni et al. 2005) or our previous work (Kapsammer et al. 2012), focus on resolving data model heterogeneity on the schema level. They extract *schemas* in the source technical space and transform them into schemas of the desired target technical space. Thus, in this paper, we complement the above approaches with a model-driven one, *automating* the configuration of *instance data transformation* processes, with a *focus on resolving data model heterogeneity* and a clear separation from other tasks during integration and from the resolving of other kinds of heterogeneities. We assume, that technical heterogeneities are already resolved, for instance, by building appropriate adaptors employing social network APIs (e. g., using HTTP, OAuth, etc.). For resolving structural and semantic heterogeneities later on, established integration techniques may be utilized, which are supported by general purpose modeling tools, such as Enterprise Architect[7], and by a multitude of dedicated integration tools, such as COMA++ (Massmann et al. 2011) for similarity matching, or MapForce[8] for mapping.

This model-driven approach is applied to instance data transformation between JSON data from social networks as source, and OWL, as proposed by the Linked Data Initiative (Heath & Bizer 2011), as target schema. JSON was used due to its popularity in social networks (also, existing approaches did not consider it much yet). OWL was chosen over RDF to be extensible for integration mappings to consolidate profiles. Exploiting the semantic features of OWL, *generic instance transformation specifications* may be applied, which are independent of concrete social network schemas and OWL ontologies.

[7] www.enterprisearchitect.at
[8] www.altova.com/mapforce

Structure of the paper. In the next section, we discuss related research, before in Section 3 and Section 4 we present our approach and an implementation thereof using the Eclipse Modeling Framework. In Section 5 we discuss results and lessons learned from the application on actual user profiles from Facebook, Google+, and LinkedIn. Finally, in Section 6 we give an outlook on *integration of user profile data*, present prospects for handling *evolution* of sources and the propagation of changes, and discuss the *generalization* of the approach for arbitrary source and target technical spaces beside JSON and OWL.

2 Related Work

In this section, we discuss closely related work with respect to instance transformation between JSON and OWL for overcoming data model heterogeneity, ontology engineering, as well as related approaches from the data and model engineering domains.

Concerning the specific transformation of JSON data into RDF graphs or OWL ontologies, most closely related work can be found in terms of so-called RDFizers. Such RDFizers are available for a plethora of different sources, ranging from specific websites over social networks to relational databases and plain files. RDFizers are implemented, for instance, as part of Virtuoso (an enterprise data integration server), Aperture for extracting and querying several information systems in the Semantic Desktop project (Dengel 2007), in D2R and R2RML for publishing relational databases to the Semantic Web (Bizer & Cyganiak 2006, Das et al. 2012), for transforming various microformats into RDF using, for instance, Triplr or Any23[9], as well as for extracting information from the Old REST API of Facebook (Rowe & Ciravegna 2008) and from Twitter (Mendes et al. 2010). All these frameworks either must be configured manually with respect to data extraction, transformation to RDF, and integration into common ontologies, or simply utilize hard-coded rules for this purpose (specified manually). Also, these approaches often mix the resolving of data model, structural, and semantic heterogeneities, incurring the disadvantages already discussed above. For example, in Virtuoso, so-called cartridges are responsible for extracting data from various sources, and for transforming them into RDF graphs. These graphs utilize the vocabularies pro-

[9] developers.any23.org

vided by various target ontologies, for instance, FOAF and a complementing Facebook ontology provided by OpenLink, which captures the concepts of Facebook not present in FOAF. For this, Virtuoso requires the target ontologies and XSLT definitions of transformations to these target ontologies, which are often specified manually. Semion (Nuzzolese et al. 2010) goes beyong such triplifiers by focusing on a *customizable* triplification process. Keeping in mind that data source schemas may frequently change, creating configurations, hard-coded rules, and target ontologies manually is a laborious and error-prone task. Our approach, in contrast, separates overcoming different kinds of heterogeneities into sequential transformation and integration steps, also aiming at *automating these steps*.

Addressing the challenges of *ontology engineering*, the application of model driven architectures for development of *Semantic Web ontologies* has been proposed by (Gašević et al. 2009). An in-depth discussion of meta-models in combination with ontologies for software engineering was done by (Henderson-Sellers 2011), including extensive related work. The method by (Cranefield & Pan 2007) employs Jena rules to create RDF from MOF-based models, whereas (Gašević et al. 2007) use XSLT to generate OWL from UML. The approach of (Glimm et al. 2010) uses OWL2 for meta-modeling, specifying class constraints and relationships as OWL axioms, and *synchronizing* them with individuals' role assertions. Our approach combines and *generalizes* these ontology engineering concepts into a model-driven approach that may be configured to *arbitrary* source and target *technical spaces*.

Concerning the individual steps in the transformation process, in particular with respect to schema and instance transformation, in the *data engineering domain* a considerable amount of research has been conducted. Existing generic approaches map different schemas of the same technical space and exchange data between these schemas (e. g., cf. (Doan & Halevy 2005, Fagin et al. 2009, Legler & Naumann 2007) for surveys on such approaches). More specific approaches (i) map between relational and XML schemas and instances, cf. (Yahia et al. 2004), (ii) map structured sources into RDF, such as (Knoblock et al. 2012, Speiser & Harth 2010), (iii) transform XML to JSON or XML to OWL/RDF (Bohring & Auer 2005, Cardoso & Bussler 2011, Kobeissy et al. 2007, Bischof et al. 2011), and (iv) align the individuals of different ontologies, such as (Noy 2004). To date, however, most approaches rely on *manually created* and *specifically tailored* transformation specifications and as a consequence, are vulnerable to evolution of source and target schemas. The requirements mentioned above (schema and metamodel evolution, platform-independent transformations, etc.) that drove our choice of a model-driven architecture pare of minor importance in these references.

Being applicable also in presence of schema evolution, especially interesting are the approach of (Bohring & Auer 2005) and CLIO (Fagin et al. 2009, Haas et al. 2005). These approaches generate transformation code in the form of XSLT, XQuery, and SQL queries (depending on the source and target technical space) in order to overcome technical and syntactic, data model, structural, and semantic heterogeneity in a single step. The ideas of such transformation code generation are the basis for the instance transformation step in our model-driven transformation approach for social network data integration, which, as already noted above, separates overcoming different kinds of heterogeneities into sequential steps.

Concerning schema and instance transformation in the *model engineering domain*, the general idea of bridging several modeling layers within one approach has been presented, for instance, in the well-known work by (Atzeni et al. 2005). For bridging the meta-model layer, a so-called supermodel has been proposed allowing to transform schemas between different technical spaces, such as OO, OR, ER, UML, and XSD. For actually transforming the corresponding instances thereof, so-called down functions have been realized (Atzeni et al. 2006), allowing to transform meta-model translation rules down to instances. In contrast, we exploit the inference capabilities of OWL, and thus, need not create instance transformations from schema transformations.

3 Architectural Overview

An initial overview of required steps for resolving all kinds of heterogeneities was shown in Fig. 1 above. In order to provide transformations independently of input (e. g., JSON) and output formats (e. g., OWL/XML), we propose a model-driven approach to bridging data model heterogeneity. In the following, the approach is detailed by means of a source JSON document, as extracted from a social network, and a target OWL/XML document, including the corresponding transformation of schemas. The proposed *model-driven transformation process* for resolving this data model heterogeneity is depicted in Fig. 2, and discussed in the following.

Meta-modeling layers and transformations. Our approach anchors the transformations of schemas and instances along the *four* meta-modeling layers of MOF (Object Management Group 2011). The bottom layer (M0) describes actual *instances* (e. g., user profile instance data from Facebook in JSON or in an OWL ontology A-Box). These instances conform to *models* at the model layer (M1, e. g., the Facebook Graph API's schema and JSON in general, or its representation as Facebook T-Box). In turn, these models conform to so-called *meta-models* (M2, e. g., JSON Schema as a language for defining the schemas of JSON documents or OWL as language for defining ontologies in the Semantic Web technology stack). Finally, meta-models are described in terms of a *meta-meta-model* (M3), such as Ecore[10].

In this four-layer representation, transformations for bridging technical spaces on a particular layer are always specified on the superior layer: thus, transformations on the M1 layer (e. g., from Facebook schema to a Facebook T-Box) are *specified on the* M2 *layer*, and transformations on the M0 layer (e. g., from Facebook instances to individuals in a Facebook A-Box) are *specified on the* M1 *layer*, as depicted in Fig. 2.

Schema and instance transformation. For automatically transforming schemas, a transformation $tss : L_{m2} \rightarrow L_{m2'}$ has to be specified, for instance, from JSON Schema to OWL T-Box axioms, which allows to transform the corresponding schemas. For instance, a Facebook schema may be transformed to an according T-Box by executing the transformation specification, i. e., $L_{m1s'} = tss(L_{m1s})$.

For actual instance data, existing approaches require *specific* instance transformation specifications $tsi_s : L_{m1s} \rightarrow L_{m1s'}$ between pairs of source L_{m1s} and target $L_{m1s'}$ models. Thus, to automatically transform Facebook instance data into Facebook A-Box axioms, a transformation specification between the Facebook Schema and the Facebook T-Box would be required, and for transforming data from LinkedIn

[10] Ecore is the realization of MOF in the Eclipse Modeling Framework (EMF) `www.eclipse.org/modeling/emf`

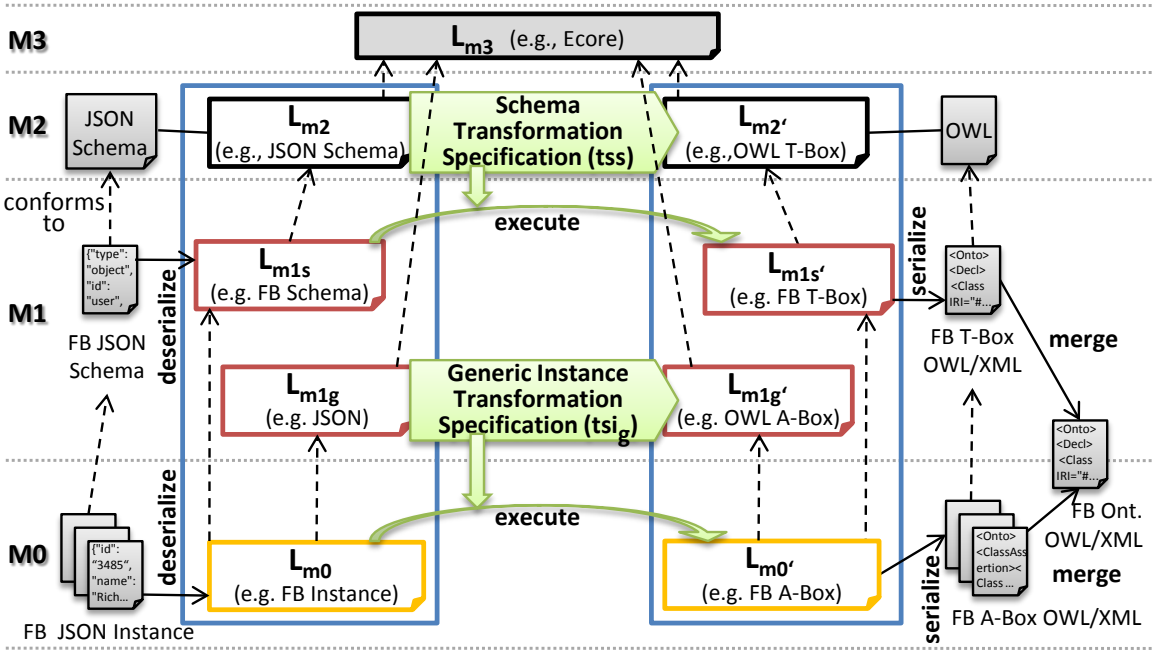

Figure 2: Overview of the model driven transformation process

and Google+, another two specifications would be needed. We argue, that instead of multiple specific instance transformation specifications tsi_s, generic ones $tsi_g : L_{m1g} \rightarrow L_{m1g'}$ can be defined, if the source instances carry certain partial schema information when being interpreted in terms of a generic source language L_{m1g}, and the target language $L_{m1g'}$ supports DL and rule inference. Markup languages, such as JSON and XML, used in today's online social network APIs, and OWL satisfy this requirement.

In the next section, we discuss our transformation approach for resolving data model heterogeneity between JSON Schema and OWL T-Box axioms on the *model layer* (M1), as well as between JSON and OWL A-Box axioms on the *instance layer* (M0). This transformation approach makes use of OWL inference capabilities to enable the specification of *schema transformations* on the meta-model layer, as well as *instance transformations* on the model layer, which replace manually created transformation specifications for each kind of data source.

4 JSON to OWL Transformation

In principle, our approach as depicted in Fig. 2 comprises two steps on the four-layer meta-modeling stack, transforming (i) JSON Schema to OWL T-Box axioms, and (ii) JSON instances to OWL A-Box axioms. Both these steps can be realized using model transformation techniques. The execution of a schema transformation specification, first, processes a source model (e. g., a Facebook schema deserialized from its textual representation) to create a corresponding target model (e. g., a Facebook T-Box, serializable into a textual representation). Second, when executed, a generic instance transformation specification (i. e., independent of Facebook Schema and T-Box) processes source instances (e. g., Facebook instances deserialized from JSON responses of Facebook) to create the target instances (e. g., a Facebook A-Box), which are serialized into a textual representation, such as OWL/XML. Finally, the result of the schema transformation execution is merged with the instance transformation execution result into a coherent Facebook ontology. The specifics of JSON and OWL, which are exploited in these generic transformations, are detailed below. The actual transformation specifications are given in Sect. 4.2.

4.1 JSON to OWL by Example

Let us consider user information from a social network (e. g., Facebook) extracted in JSON, as depicted in Fig. 3. This snippet, which is rather simple for the sake of understandability, shows a JSON object with a single property `name`, whose value is `'Jane Doe'`. The JSON object conforms to a simplified Facebook schema, defining that every `User` is of type `object` and comprises a property `name`.

Transformation of JSON Schema to OWL T-Box. In order to provide the Facebook schema in a T-Box (e. g., to support querying and reasoning), in a first transformation step, the schema is transformed into corresponding OWL T-Box axioms. These axioms define that `User` is equivalent with the class of things that have a `name` property of type `String` (in OWL, this may be specified by the domain and range of a data property).

Transformation of JSON to OWL A-Box. In a second transformation step, the Facebook instance is transformed into corresponding OWL A-Box axioms, which again are specified in DL notation. As basic schema information, the format of the `name` property's value in our sample JSON snippet allows us to derive the property's type: JSON distinguishes between string, number, boolean, object, and array. Further schema information, such as the concrete type of object (e. g., a person vs. an address), is not available in the instances. Anyhow, this is where OWL, implementing the family $\mathcal{SROIQ}$ of description logic (Grau et al. 2008), is a perfect fit on the target side: description logic reasoners are specifically designed to classify objects according to their role assertions, and hence, are able to infer the schema information that is not explicitly present in JSON instances (e. g., given a sample T-Box axiom specified in description logic notation $User \equiv \exists name.String$, a description logic reasoner infers that anything with

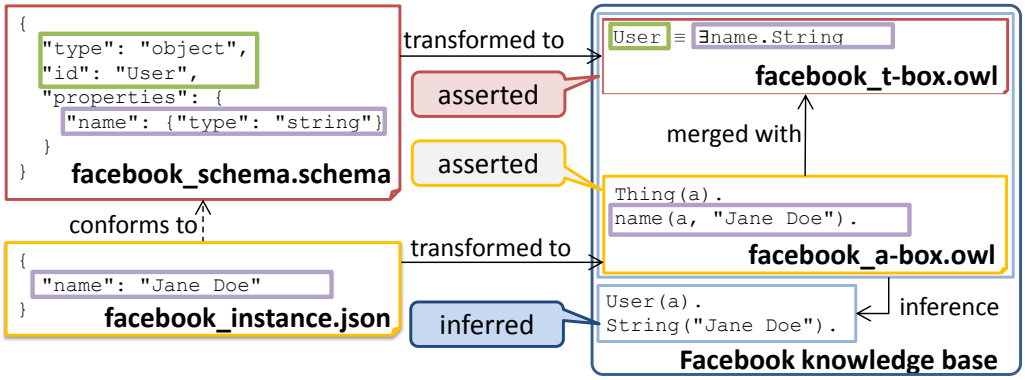

Figure 3: Sample JSON to OWL transformation

at least one name is a user). Solutions to cope with potential ambiguities will be discussed in Section 4.4 below. As a result, the instance transformation specifications do not necessarily need information from a concrete model (e. g., a Facebook schema) for transformation. Thus, all JSON objects can be transformed into generic concept assertions of the kind `Thing(a)` (instead of specific ones, such as `User(a)`). In a similar manner, every value of a primitive property (e. g., `'Jane Doe'`) can be transformed into a concept assertion of the kind `Literal`. Finally, the connections between objects and the values of their primitive and complex properties (e. g., the fact that our sample user has the name `'Jane Doe'`) have to be transformed into role assertions (e. g., `name(a, 'Jane Doe')`). In case that a primitive or complex property allows an array of values, every array element must be represented with a corresponding role assertion.

Merging of OWL T-Box and A-Box. When being merged with the T-Box axioms, a description logic reasoner, such as HermiT[11], infers concept assertions, as explained above. As a result, we can specify all instance transformations $tsi_g : L_{m1g} \rightarrow L_{m1g'}$ between JSON as source model and OWL A-Box axioms as target model in a generic manner. The specifications for both, generic schema and instance transformation, are detailed below.

4.2 Transformation Specifications

For specifying the actual schema and instance transformations, one may resort to existing (model) transformation languages, such as ATL (Jouault et al. 2008), QVT (Object Management Group 2009), or XSLT. Since existing tools for publishing social network data in RDF format use various transformation languages, we utilize our Mapping Operator language (MOps) (Wimmer et al. 2010*b*), which is designed as a suitable basis for creating transformation specifications in different kinds of transformation languages. Also, in our prototype, we use an executable implementation of MOps to perform the actual transformation. The basic building blocks of the MOps language for specifying transformations are so-called *kernel MOps*, which are composed to reusable higher-level transformations, denoted as *composite MOps*. Kernel MOps, their interplay, as well as the composition to composite MOps are explained below, as part of their application to schema and instance transformation specification. For a detailed description of kernel and composite MOps we refer to (Wimmer et al. 2010*a*,*b*).

[11] `hermit-reasoner.com`

Schema transformation specification. The specification of schema transformations from JSON Schema to OWL T-Box using MOps is depicted in Fig. 4. Each MOp has *input ports* for accepting input on the left side and *output ports* for producing target objects on the right side. Ports are typed to classes (C) or attributes (A).

The JSON Schema meta-model is a subset of the JSON Schema Internet Draft[12], which was selected for ease of presentation in a straightforward manner. For the OWL T-Box meta-model, we based on OMG's Ontology Definition Metamodel[13]. Note, that for presentation purposes, the OWL T-Box meta-model was simplified, i. e., axioms for domain and range of properties are modeled as relationships (instead of subclasses of `Axiom`).

In principle, every source element (`Schema` and `Property`) results in a target `Declaration`, as indicated by two `Copier` MOps in Fig. 4. The kind of declared entity depends on the `type` of the transformed source `Schema`: (i) complex schemas (type has value "object" or "array") can best be represented as instances of `Class` in OWL, while (ii) primitive schemas naturally become instances of `Datatype`. Since the topmost complex schema must be additionally transformed into an instance of `Ontology` (every OWL ontology is represented with one such instance), a composite MOp in terms of a horizontal partitioner is used. Such an `HPartitioner` comprises several `CondCopiers`, which restrict the output of a `Copier` to a subset satisfying a certain condition—in our case an ontology is only output for the topmost schema, and classes are only output for those schemas with type "array" or "object". For transforming primitive schemas (i. e., simple types), we create an instance of `Datatype` for each distinct *value* of the `type` property in the source model (i. e., one datatype for "string", one for "boolean", and another one for "number"). Hence, the composite MOp `ObjGenerator` is used, which is depicted in white-box view, exposing the comprised kernel MOps. It includes (i) an $\mathtt{A_2C}$ kernel MOp (transforms the value of an attribute of the source meta-model into a class instance of the target meta-model) for creating the instance of `Datatype`, and (ii) an $\mathtt{A_2A}$ kernel MOp (copies an attribute value into another attribute value) for setting the IRI (Internationalized Resource Identifier) of the newly created datatype. Since there are dependencies between these kernel MOps (i. e., the IRI can only be set after the datatype has been created), the $\mathtt{A_2C}$ MOp is additionally linked to the $\mathtt{A_2A}$ MOp through a *trace port* (T), which provides context information about

[12] `tools.ietf.org/html/draft-zyp-json-schema-03`
[13] `www.omg.org/spec/ODM/1.0`

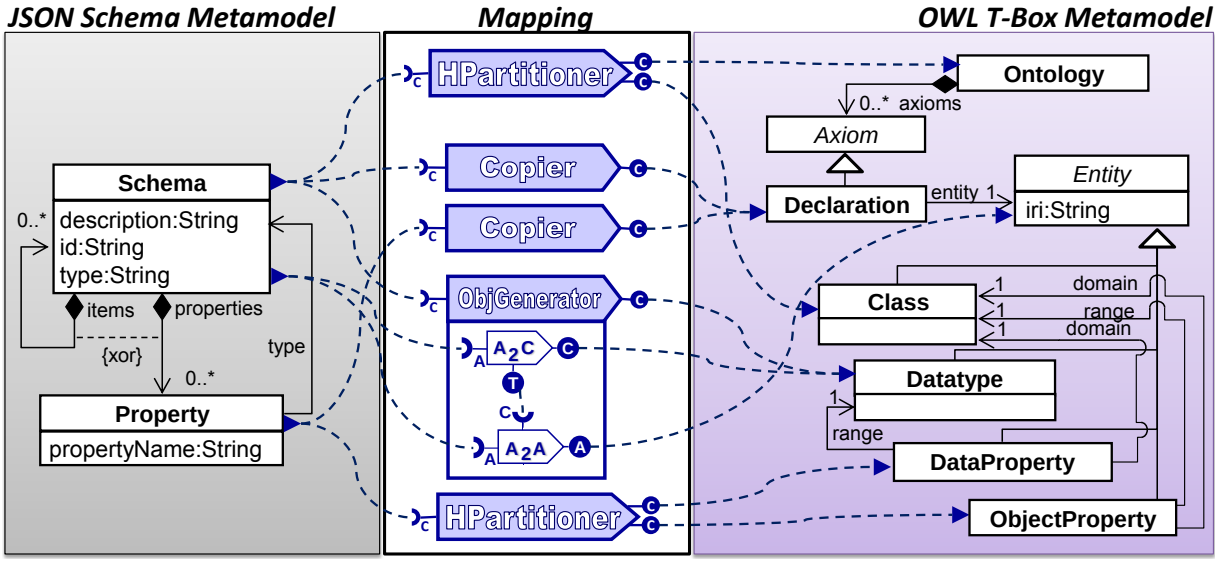

Figure 4: Transformation of JSON Schema to OWL T-Box using MOps

the produced output objects. Finally, instances of `Property` must either be transformed to instances of `ObjectProperty` (in case they reference a complex schema) or to `DataProperty` (in all other cases). Analogously to transforming instances of `Schema`, an `HPartitioner` can be used for that.

Instance transformation specification. Fig. 5 summarizes the instance transformations between JSON and OWL. The meta-models were built on basis of the same specifications as those for schema transformation.

As already discussed in the transformation example above, JSON objects are transformed in a straightforward manner to OWL individuals of unspecified type (i. e., `Things`). Hence, one `Copier` creates an `Individual` for each `Object`, while another one creates a `ClassAssertion` referring to the entity `Thing`. Node identifiers (IRI) are generated, if present, from nested key properties ("id", "key"), or, otherwise, from the filename (for root objects) or employing a simple heuristic, which uses the hash value of all nested members. Thereby, same objects can be matched (i. e., get equal identifiers), as long as they do not contain another layer of nested objects. In that case, distinct IRIs are generated.

Analogously to the schema transformation specification, on the instance level we distinguish between members of complex and of primitive type. Consequently, we utilize an `HPartitioner` to create an `ObjectPropertyAssertion` for members with complex values and a `DataPropertyAssertion` for members with primitive values. These assertions must reference the corresponding entities that were created during schema transformation (e. g., a "firstName" member must be transformed into an assertion of the corresponding "firstName" data property[14]). Another `Copier`, again depicted in white-box view, creates new `Entity` instances from `Member` instances (the `name` of the member serves as the entity's IRI). During merging of T-Box and A-Box, IRI equivalence ensures that the A-Box entities can be connected to their counterparts in the T-Box. Finally, every primitive value (`Boolean`, `Number`, and `String`) must be copied to an instance of `Literal`, but only if it was not created before (hence, the `HMerger` contains in fact three `CondCopiers`).

4.3 Implementation in EMF

As implementation platform, we chose the *Eclipse Modeling Framework* (EMF), including *Xtext* and *Xtend* for text (de-)serialization and transformation, due to its maturity and large community support. For implementing MOps and transformations in Xtend, meta-models in Ecore are required. The meta-models for JSON, JSON Schema, and OWL were generated from Xtext grammars according to the specifications introduced above. These Ecore meta-models are automatically translated by EMF into Java classes, which can then be used in Xtend. To switch to a different source technical space — in the past Facebook and the Twitter streaming API switched from XML to JSON — our implementation would only require Xtext grammars to generate the new meta-models for de-serialization to Ecore.

In order to utilize existing ontology tools and description logic reasoners, the Ecore representations of T-Box and A-Box resulting from applying the Xtend implementation of our MOps, then, must be serialized (e. g., as RDF/XML or OWL/XML) and merged. Again, Xtend is used for this purpose. For increased compatibility we implemented serializers for both, OWL/XML, e. g., for loading in Protégé[15], and RDF/XML, as required for Apache Jena[16].

4.4 Type Inference & Reasoning

Loading the generated files in Protégé enables the application of the included HermiT reasoner for type inference. Generic instance transformation specifications not taking into account schema information, however, may result in ambiguities during classification by a description logic reasoner. First, classes having the same mandatory, but different optional properties, cannot be matched unambiguously, in case that an object thereof is described in terms of the mandatory properties, only. Second, equally named and typed properties with different constraints in two different classes cannot be distinguished (e. g., a

[14] Keeping in mind our focus on resolving data model heterogeneity, both the source and target technical space have the same structure, and, thus, we can safely assume name equivalence between JSON members and OWL properties.

[15] `protege.stanford.edu`, used version: 4.2 beta (build 276)

[16] `jena.apache.org`, used version: 2.7.0

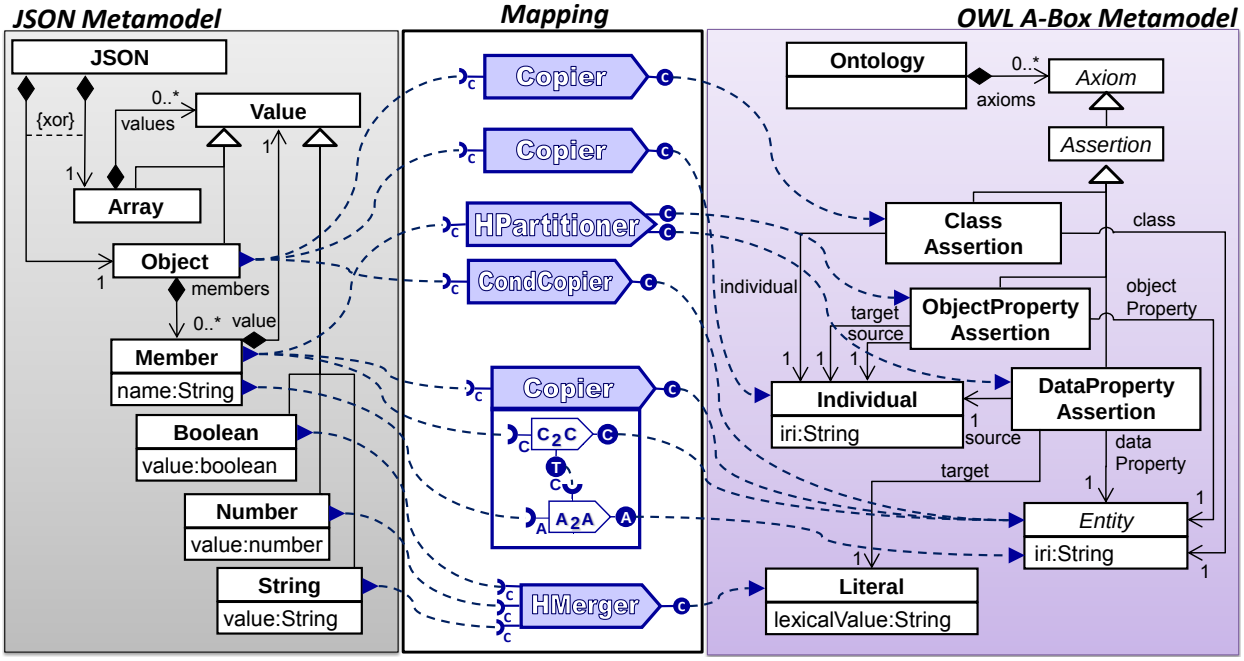

Figure 5: Transformation of JSON to OWL A-Box using MOps

class `AustrianAddress` with a 4-digit zip code vs. a `GermanAddress` with 5 digits).

To alleviate this problem, we propose to encode additional meta information from instance data in an OWL T-Box — in a similar manner as previously shown for schema extraction (Kapsammer et al. 2012). For example, the property `category` of Facebook pages (a key concept in Facebook data, used universally) may be used to define equivalences with specific classes: $Restaurant \equiv Page \sqcap \exists category.\{restaurant\}$. Previously (Kapsammer et al. 2012), various heuristics for tackling this problem have been proposed: for example, `IdFromValue` infers a class name from property values (e. g., "type" or "category" in Facebook, and property "kind" in Google+), while `IdFromReferenceName` infers a class name from the names of references to nested individuals (e. g., used in LinkedIn). These different heuristics can be exploited during instance transformation as well. Therefore, each heuristic is encoded as a generic declarative Jena rule with accompanying built-ins for imperative computations. For instance, regarding connections in Facebook, the API's JSON response, if requested, contains an object `metainfo`, containing an array `connections`, which in turn contains multiple properties, having URIs as values. These are links for other requests to receive further data, but they would actually resemble a connection between the response's root object, and the root object of another JSON object (e. g., a list of friends). This `LinkPatternFromValue` strategy can be expressed as a Jena rule, thereby providing a direct edge between such objects (resulting in one edge per connection, instead of two intermediate nodes and three edges). As an alternative to built-ins, such definitions can be automatically generated by our previously proposed schema extraction process (Kapsammer et al. 2012) in terms of T-Box axioms, or as reasoning rules for semantic web rule reasoners (e. g., Jena inference rules).

5 Results & Evaluation

In this section, we *evaluate* our prototypical implementation by means of transforming data from Facebook, Google+, and LinkedIn to corresponding OWL ontologies. Thereby, we will discuss several aspects: completeness, consistency, conciseness, as well as performance and scalability.

Evaluation Setup. In order to obtain comparable results, *equal test user profiles* were created in each of the selected social networks and extracted via their API. These profiles contain basic user information, jobs, education, as well as a connected friend for direct communication and interaction within a group. Note, that these data sets, since they were created manually, do not reflect the complete information available in real social network profiles. Nonetheless, as they were created in a consistent manner, they are suited for a first evaluation. (cf. (Kapsammer et al. 2012) for details on user profiles and generated schemas). The input data sets from Facebook, Google+, and LinkedIn as JSON files, as well as a simple introductory example, are available online[17]. Furthermore, the files include generated JSON Schema files, as explained by (Kapsammer et al. 2012), T-Box and A-Box in Ecore format, the merged serialization thereof (in RDF/XML & OWL/XML), as well as (for the three social networks) Jena rules and reasoning results.

Completeness. The completeness of the extracted data from social network APIs was already discussed previously (Kapsammer et al. 2012). The requirement of *all information* from the JSON input to be present in the output is *fulfilled* by the proposed transformation approach: all JSON objects, arrays, and simple types are transformed to OWL (i. e., all input is present in output). This was evaluated by manually comparing the input to the generated instances, object properties, and datatype properties, on multiple samples from all four data sets.

Consistency. To evaluate the consistency of the transformations, we compared the outputs of *multiple runs* on the same input data. First, the more or less *random serialization order* of assertions is not a problem, since Protégé shows classes and individuals in alphabetical order anyway. Second, as discussed

[17] `social-nexus.net/publications`

Table 1: Description of input, output, and execution times—including count of files and individuals, file sizes (plain/zip-compressed), generated IDs and ID mismatches, as well as average execution times for transformations and reasoning.

Input	Simple Ex.	Facebook	Google+	LinkedIn
Number of input instance files (JSON)	**1**	**192**	**11**	**21**
Size of input files (JSON Schema + JSON) plain/compressed	1/1 kB	399/128 kB	31/12 kB	41 (14) kB
Output	Simple Ex.	Facebook	Google+	LinkedIn
Number of individuals (in A-Box)	**4**	**1549**	**104**	**260**
Size of output files (T-Box, A-Box & inferred triples in RDF/XML)	13/2 kB	626/63 kB	58/6 kB	113/9 kB
Number of individuals without unique ID from JSON input	2	1078	64	187
Mismatches of generated IRI for objects with equal content	0	53	4	25
Transformation & Reasoning Time	Simple Ex.	Facebook	Google+	LinkedIn
Sum of transformation & reasoning time	**196 ms**	**8621 ms**	**2198 ms**	**3186 ms**
JSON Schema to OWL T-Box Ecore	45 ms	107 ms	64 ms	67 ms
JSON to OWL A-Box Ecore	62 ms	2541 ms	282 ms	559 ms
OWL Ecore to RDF/XML & OWL/XML	90 ms	2322 ms	397 ms	774 ms
Jena Rule Reasoning on RDF/XML	-	3651 ms	1455 ms	1785 ms

earlier, for individuals without a unique ID from the JSON input, an *identifier* has to be generated. These generated IRIs may *differ between runs*, but are always consistent within a generated ontology. In this context, Table 1 also counts mismatches of generated IRIs for objects with equal content (i. e., the heuristic fails, if JSON objects contain nested objects, as discussed in Sect. 4.2).

Conciseness. Comparing the file size for JSON input and OWL output, Table 1 shows that plain RDF/XML files are larger than the JSON counterparts. Zip-compression, however, works much more efficiently for the RDF/XML format.

Regarding the conciseness of transformed output itself, we compare the generated serialization to the *minimal representation of the same facts*. Such a minimal OWL A-Box would contain one assertion for each individual, datatype property, and object property. In this sense, the generated RDF/XML is not minimal, as we assert all individuals as `Things`, and the reasoner adds inferred assertions with more specific types. For properties, however, the transformation is minimal. Concerning the JSON tree structure, in which data is provided by social network APIs, the *length of paths* is crucial for navigation. Our implementation converts these trees, being in fact a special case of graphs, to tree-like ontology graphs, which then allow the introduction of *shortcuts* (e. g., for Facebook connections, as discussed in Sect. 4.4). On one hand, these shortcuts represent *additional assertions*, but, on the other hand, materializing these shortcuts enable *faster queries* that are also *easier to express*.

Performance & Scalability. To evaluate the *time complexity* of our approach, we measured the execution times[18] for the different data sets in our prototype, as shown in Table 1. From these measurements we can observe that the size of the JSON Schema correlates with its transformation time to Ecore. The same applies for the transformation of JSON instances to Ecore. Concerning the Jena inference step, clearly the reasoning takes most of the time in the overall process, with the benefit of being able to infer information that is not present in the source data explicitly. Note, that the times in the above table include file I/O. In a separate run without those JSON files from the Facebook input, which contain almost no members (i. e., empty objects and arrays), the transformation of instances to Ecore was sped up by 37%, with remaining 92% of individuals within one third of files. Thus, to make the transformation more efficient, all steps may be integrated into a single application with reduced file system access for input and intermediate files. Concerning scalability, the average transformation times for larger inputs grow, from a certain point, linearly (transformation & serialzation), and exponentially for the reasoning step. However, using specific rules that do not require imperative computations (cf. Sect. 4.4), the magnitude would probably be reduced. On average, for 62.000 individuals plus properties, the transformation to Ecore took 37 seconds, the Jena inferencing took 184 seconds.

Finally, *memory complexity* was measured in terms of RAM consumption. Whereas for transformation of these 62.000 individuals plus properties the Java process consumed 1025 MB, growing linearly, for the reasoning it peaked at 335 MB, almost constantly (i. e., almost no growth).

[18] average of at least 5 runs, on a notebook PC (Intel i7-Q740, 8 GB RAM, running Windows 7 64-bit)

6 Conclusion and Future Work

We presented an approach for model driven *transformation of schemas and instances* between different technical spaces. Our method requires the transformation *specification* to be done *only once*, for instance, from JSON Schema to OWL T-Box axioms. This specification can then be *executed for different data sources*, such as different social networks providing JSON data via their APIs. Neglecting semantics, JSON's tree-like structure can be transformed to OWL in a straightforward manner (e.g., JSON slots of simple datatype as datatype properties, complex types and arrays as object properties). To go beyond syntactic equivalence, and to consider *semantics of* APIs, some configuration was required, which obviously depends on the data source. For instance, semantic equivalence from JSON slots is not generally possible, but in Facebook ID slots can be used to define semantic equivalence.

In the *evaluation* section we applied our approach to *comparable user profiles* from Facebook, Google+, and LinkedIn. Not surprisingly, *time and memory complexity* were relatively high (especially for reasoning and for larger user profiles), but clearly there is potential for optimizations, and an *evaluation on extensive data sets* would be interesting for future work.

Our focus was on the *pre-requisite steps for data integration* and consolidation from different social networks: the goal was to *overcome data model heterogeneity*, in order to facilitate structural and semantic *integration later on*. In contrast to related transformation approaches, using model driven architectures allows to build graphical editors and to cope with *evolution*, for instance when APIs change. Also, they enable source/target formats exchange without influencing transformation rules, and platform-independent MOps allow replacing the transformation platform.

Generalization to arbitrary source and target models. For generalizing the presented approach to technical spaces other than JSON as source and OWL as target, several modifications need to be taken into account. Without the reasoning capabilities of OWL and Jena rules, instances need to be loaded according to their concrete source schema (instead of some generic meta-model), therefore, preventing generic instance transformations that are independent of source models. As a consequence, firstly, dedicated (de-)serializers for every single source and target schema are required, and, secondly, the generic instance transformation specification (specified on the model layer) must be replaced with specific ones for each social network schema. In order to automate this whole process, the deserializers for source schemas should be generated automatically. Furthermore, the *instance transformation specifications on the model layer* are foreseen to be created automatically as artifacts as well—just like the target model classes are generated—during the execution of the transformation from source to target model (specified on the meta-model layer). This means, a sole transformation specification on the meta-model layer may result in potentially many transformation specifications on the model layer. However, as a result of the limitations of current model engineering software frameworks (specifically, the fact that Eclipse modeling spans three of the four meta-modeling layers, only), the models created during schema transformation must be *lifted* to the meta-model layer first. This means, that *instances* in the schema transformation specifications must *become models* in the instance transformation specification.

Semantic integration of schemas. Having resolved data model heterogeneity between different social networks by transformation to OWL, instance based schema matching tools, such as COMA++ (Massmann et al. 2011), may be used to support humans in defining semantic correspondences. Such tool support is especially helpful for large or unknown schemas. However, unlike transformations for resolving technical heterogeneity, which were shown in this paper to be specifiable in a generic manner, transformations resolving semantic heterogeneity still need manual intervention. Therefore, we can resort to Semantic Web technologies to integrate the transformed user models in OWL/XML, for instance, by defining equivalences between the OWL classes `User` from Facebook and `Person` from FOAF. Again, employing reasoners allows to retrieve instances materialized as Facebook A-Box from queries using the FOAF vocabulary.

Source Schema Evolution. Once such mappings are specified, an especially interesting question therefore is, how to *automate* or support *evolution* of these transformations. A first idea in this direction is the design of a *meta-model of possible schema changes* in EMF, including generic operations to *propagate changes* to dependent artifacts, such as queries and integration rules.

References

Abel, F., Araújo, S., Gao, Q. & Houben, G. J. (2011), Analyzing cross-system user modeling on the social web, *in* 'Proc. of the 11th int. conf. on Web engineering', ICWE'11, Springer, pp. 28–43.

Aroyo, L. & Houben, G.-J. (2010), 'User modeling and adaptive Semantic Web', *Semantic Web* **1**(1), 105–110.

Atzeni, P., Cappellari, P. & Bernstein, P. A. (2005), ModelGen: Model Independent Schema Translation, *in* 'Proc. of the 21st Int. Conf. on Data Engineering', ICDE '05, IEEE Computer Society, pp. 1111–1112.

Atzeni, P., Cappellari, P. & Bernstein, P. A. (2006), Model-Independent Schema and Data Translation Advances in Database Technology - EDBT 2006, Vol. 3896 of *LNCS*, Springer, pp. 368–385.

Bischof, S., Decker, S., Krennwallner, T., Lopes, N. & Polleres, A. (2011), Mapping between RDF and XML with XSPARQL, Technical report, DERI.

Bizer, C. & Cyganiak, R. (2006), D2R Server – Publishing Relational Databases on the Semantic Web, *in* 'Proceedings of the 5th International Semantic Web Conference'.

Bleiholder, J. & Naumann, F. (2009), 'Data Fusion', *ACM Comput. Surv.* **41**(1), 1–41.

Bohring, H. & Auer, S. (2005), Mapping XML to OWL Ontologies, *in* K. P. Jantke, K.-P. Fähnrich & W. S. Wittig, eds, 'Leipziger Informatik-Tage', Vol. 72 of *LNI*, GI, pp. 147–156.

Bozzon, A., Brambilla, M. & Ceri, S. (2012), Answering search queries with CrowdSearcher, *in* 'Proceedings of the 21st international conference on World Wide Web', WWW '12, ACM, New York, NY, USA, pp. 1009–1018.

Cardoso, J. & Bussler, C. (2011), 'Mapping between heterogeneous XML and OWL transaction representations in B2B integration', *Data & Knowledge Engineering* **70**(12), 1046–1069.

Cranefield, S. & Pan, J. (2007), 'Bridging the gap between the model-driven architecture and ontology engineering', *Int. J. Hum.-Comput. Stud.* **65**(7), 595–609.

Das, S., Sundara, S. & Cyganiak, R. (2012), 'R2RML: RDB to RDF Mapping Language', W3C Working Draft, http://www.w3.org/TR/r2rml/.

Dengel, A. R. (2007), *Knowledge Technologies for the Social Semantic Desktop*, Vol. 4798 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, Berlin, Heidelberg, chapter 2, pp. 2–9.

Doan, A. & Halevy, A. Y. (2005), 'Semantic-integration research in the database community', *AI Mag.* **26**(1), 83–94.

Fagin, R., Haas, L. M., Hernández, M., Miller, R. J., Popa, L. & Velegrakis, Y. (2009), Conceptual Modeling: Foundations and Applications, Springer-Verlag, Berlin, Heidelberg, chapter Clio: Schema Mapping Creation and Data Exchange, pp. 198–236.

Gašević, D., Djurić, D. & Devedžić, V. (2007), 'MDA-based Automatic OWL Ontology Development', *Int. J. Softw. Tools Technol. Transf.* **9**(2), 103–117.

Gašević, D., Djurić, D. & Devedžić, V. (2009), *Model Driven Engineering and Ontology Development*, 2nd edn, Springer Publishing Company, Incorporated.

Glimm, B., Rudolph, S. & Völker, J. (2010), Integrated metamodeling and diagnosis in OWL 2, *in* 'Proceedings of the 9th international semantic web conference on The semantic web - Volume Part I', ISWC'10, Springer-Verlag, Berlin, Heidelberg, pp. 257–272.

Grau, B. C., Horrocks, I., Motik, B., Parsia, B., Patel-Schneider, P. & Sattler, U. (2008), 'OWL 2: The next step for OWL', *Web Semantics: Science, Services and Agents on the World Wide Web* **6**(4), 309–322.

Haas, L. M., Hernández, M. A., Ho, H., Popa, L. & Roth, M. (2005), Clio grows up: from research prototype to industrial tool, *in* 'Proceedings of the 2005 ACM SIGMOD international conference on Management of data', SIGMOD '05, ACM, New York, NY, USA, pp. 805–810.

Heath, T. & Bizer, C. (2011), *Linked Data: Evolving the Web into a Global Data Space*, Vol. 1, Morgan & Claypool Publishers.

Henderson-Sellers, B. (2011), 'Bridging metamodels and ontologies in software engineering', *J. Syst. Softw.* **84**(2), 301–313.

Jouault, F., Allilaire, F., Bézivin, J. & Kurtev, I. (2008), 'ATL: A model transformation tool', *Science of Computer Programming* **72**(1-2), 31–39.

Kapsammer, E., Kusel, A., Lechner, S., Mitsch, S., Pröll, B., Retschitzegger, W., Schönböck, J., Schwinger, W., Wimmer, M. & Wischenbart, M. (2012), User Profile Integration Made Easy - Model-Driven Extraction and Transformation of Social Network Schemas, *in* 'International Workshop on Interoperability of User Profiles in Multi-Application Web Environments (MultiA-Pro) at WWW 2012', ACM.

Kim, W., Jeong, O.-R. & Lee, S.-W. (2010), 'On social Web sites', *Information Systems* **35**(2), 215–236.

Knoblock, C. A., Szekely, P. A., Ambite, J. L., Goel, A., Gupta, S., Lerman, K., Muslea, M., Taheriyan, M. & Mallick, P. (2012), Semi-automatically Mapping Structured Sources into the Semantic Web, *in* E. Simperl, P. Cimiano, A. Polleres, O. Corcho & V. Presutti, eds, 'ESWC', Vol. 7295 of *Lecture Notes in Computer Science*, Springer, pp. 375–390.

Kobeissy, N., Girod Genet, M. & Zeghlache, D. (2007), Mapping XML to OWL for seamless information retrieval in context-aware environments, *in* 'IEEE International Conference on Pervasive Services', IEEE, pp. 361–366.

Legler, F. & Naumann, F. (2007), 'A Classification of Schema Mappings and Analysis of Mapping Tools', *Proceedings of Datenbanksysteme in Business, Technologie und Web (BTW 2007) (Germany, Aachen, March 7-9)*.

Massmann, S., Raunich, S., Aumüller, D., Arnold, P. & Rahm, E. (2011), Evolution of the COMA Match System, *in* 'The Sixth International Workshop on Ontology Matching'.

Mendes, P. N., Passant, A. & Kapanipathi, P. (2010), Twarql: tapping into the wisdom of the crowd, *in* 'Proceedings of the 6th International Conference on Semantic Systems', I-SEMANTICS '10, ACM, New York, NY, USA.

Noy, N. F. (2004), 'Semantic integration: a survey of ontology-based approaches', *SIGMOD Rec.* **33**(4), 65–70.

Nuzzolese, A. G., Gangemi, A., Presutti, V. & Ciancarini, P. (2010), 'Fine-tuning triplification with Semion', *EKAW workshop on Knowledge Injection into and Extraction from Linked Data (KIELD2010)*.

Object Management Group (2009), 'Meta object facility (MOF) 2 query/view/transformation specification', www.omg.org/spec/QVT/1.1/Beta2/PDF/.

Object Management Group (2011), 'Meta object facility (MOF) 2 core specification', www.omg.org/spec/MOF/2.4.1/PDF.

Parundekar, R., Knoblock, C. A. & Ambite, J. L. (2010), Linking and Building Ontologies of Linked Data, *in* 'Proceedings of the 9th international semantic web conference on The semantic web - Volume Part I', ISWC'10, Springer-Verlag, Berlin, Heidelberg, pp. 598–614.

Razmerita, L., Firantas, R. & Jusevičius, M. (2009), Towards a New Generation of Social Networks: Merging Social Web with Semantic Web, *in* '5th International Conference on Semantic Systems (I-Semantics 2009)'.

Rowe, M. & Ciravegna, F. (2008), Getting to Me - Exporting Semantic Social Network Information from Facebook, *in* '1st Social Data on the Web workshop (SDoW2008)'.

Speiser, S. & Harth, A. (2010), Taking the LIDS off data silos, *in* 'Proceedings of the 6th International Conference on Semantic Systems', I-SEMANTICS '10, ACM, New York, NY, USA.

Viviani, M., Bennani, N. & Egyed-Zsigmond, E. (2010), A Survey on User Modeling in Multi-Application Environments, *in* 'Proceedings of the 3rd International Conference on Advances in Human-Oriented and Personalized Mechanisms, Technologies and Services', IEEE, Nice, France, pp. 111–116.

Wimmer, M., Kappel, G., Kusel, A., Retschitzegger, W., Schönböck, J. & Schwinger, W. (2010*a*), Plug & Play Model Transformations - A DSL for Resolving Structural Metamodel Heterogeneities, *in* 'Proceedings of the 10th Workshop on Domain-Specific Modeling (DSM'10) @ Splash 2010'.

Wimmer, M., Kappel, G., Kusel, A., Retschitzegger, W., Schönböck, J. & Schwinger, W. (2010*b*), Surviving the Heterogeneity Jungle with Composite Mapping Operators, *in* 'Proc. of the 3rd Int. Conf. on Model Transformation', Springer, pp. 260–275.

Yahia, S. A., Du, F. & Freire, J. (2004), A comprehensive solution to the XML-to-relational mapping problem, *in* 'Proceedings of the 6th annual ACM international workshop on Web information and data management', WIDM '04, ACM, New York, NY, USA, pp. 31–38.