Faculty of
Computer Science

universität
wien

Department of Distributed and
Multimedia Systems

Technical Report TR-20080202 February 2008

A Comparative Study of Mapping Solutions for Enabling Metadata Interoperability

Bernhard Haslhofer,

A Comparative Study of Mapping Solutions for Enabling Metadata Interoperability

Bernhard Haslhofer

University of Vienna
Department of Distributed and Multimedia Systems
bernhard.haslhofer@univie.ac.at

Abstract. A prerequisite for enabling uniform access to metadata in distributed, autonomous repositories is to deal with the heterogeneities among metadata information objects. Metadata mapping is one possible technique for handling incompatible metadata originating from distinct repositories and mapping solutions are its technical manifestation. Currently, there exists a variety of mapping solutions, with different conceptual characteristics and varying potential of solving solving specific types of interoperability problems. In this work, we give a comprehensive overview of existing mapping solutions and systematically compare their features and characteristics. Our analysis shows that the majority of mapping solutions are standalone-systems and support experts in mapping and integrating structured and semi-structured metadata. Solutions that conceive mapping as being a process and target broader metadata environments such as the Web are still rare.

1 Introduction

Metadata are information objects that describe some resource. Example resources are digital images, videos, or other multimedia content objects but also non-digital objects such as artefacts in museums or books in libraries. The nature of metadata information objects, i.e., their structure and the meaning of their elements (e.g., author, description, etc.), largely depends on the metadata creator's design choices and the characteristics of the repository they reside in (e.g., relational database, flat files).

For establishing access to multiple autonomous metadata repositories, one needs to deal with the distinct characteristics of the information objects stored therein. Hence, one must establish *metadata interoperability* and find an appropriate technique to deal with various heterogeneities among these objects.

Metadata mapping is such a technique. It allows domain experts to reconcile the various heterogeneities that impede metadata information objects from being interoperable. Systems can support experts in various mapping tasks: in discovering mappings, in representing mappings, in compiling mappings into executable code, and in maintaining and reusing existing mappings.

Currently there exists a variety of solutions that support experts in these tasks whereas some cover a broader, others a narrower spectrum of tasks. The

majority of solutions is implemented as standalone tools having a single domain expert as target user. Others, such as Yahoo Pipes, follow a novel collaborative, Web based approach and let ordinary Web users create and share their mappings via an intuitive, easy to use Web interface. In fact, each mapping solution has its individual features and potential of reconciling metadata heterogeneities.

The main contribution of this work is a comprehensive analysis and evaluation of existing solutions that can be applied for mapping incompatible metadata information objects. Based on the state-of-the-art literature on data integration and mapping, we set up a requirements framework against which we compare a representative selection of mapping solutions. Our analysis reveals which solutions cover which requirements and to what extend they support the mapping tasks required for establishing metadata interoperability.

The remainder of this work is organised as follows: in Section 2, we briefly introduce the notion of metadata and explain what metadata interoperability means in our context. In Section 3, we analyse requirements for metadata mapping solutions and set up the evaluation framework for our analysis. In Section 4, we present and categorise a representative selection of mapping solutions, which is then, in Section 5, evaluated against the evaluation framework. Finally, we conclude this paper with Section 6.

2 Metadata Mapping

Mapping is one possible technique for achieving metadata interoperability. Mapping solutions are the technical manifestation of this technique. Before further discussing and comparing the various characteristics of mapping solutions, we first introduce our conception of *metadata mapping*.

In Section 2.1, we first concentrate on the notion of *metadata* and regard its building blocks from a technical perspective. Then, in Section 2.2, we outline the heterogeneities that impede metadata information objects from being interoperable. In Section 2.3, we concentrate on the facets of *metadata mapping*.

2.1 Metadata — A Technical Perspective

Metadata are information objects that describe some resource, digital or non-digital. They consist of three main building blocks: schema definition languages, metadata schemes, and metadata instances. Schema definition languages provide the technical means to define metadata schemes and exist in various forms. The Unified Modelling Language (UML) [1], for instance, is a conceptual language for defining metadata schemes using a graphical notation. XML Schema (XSD) [2] and Java are languages on the programming/represenation level in information systems and permit the definition of schemes using other language primitives and a different syntax. A metadata schema is a set of elements, optionally connected by some structure, with a well-defined semantics, tailored to a certain application domain. Examples for metadata schemes in the digital libraries domain are the Dublin Core Element Set (DC) [3], the Visual Resource Association Core

(VRA Core) [4] elements, or the Learning Objects and Metadata (LOM) [5]. Besides defining elements such as "title", "creator", "description", etc., metadata schemes can also constrain the range of permitted values for each of its elements. Metadata instances comprise elements from the corresponding metadata schema and content values making up a metadata description. Figure 1 illustrates an example Dublin Core metadata description, hence an instance of the Dublin Core metadata schema, expressed in RDF/XML.

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:dc="http://purl.org/dc/elements/1.1/">

<rdf:Description rdf:about="http://media.example.com/audio/guide.ra">
    <dc:creator>Mahony, David (David James)</dc:creator>
    <dc:title>Sydney Olympics 2000, marathon runners cross Sydney
        Harbour Bridge [picture] /</dc:title>
    <dc:description>Photograph by David Mahony -- On reverse in pencil.;
        Condition: Good. Group of [marathon] runners feature eventual Gold Medal
        Winner Gezahgne Abero of Ethiopia (No. 1651) [Sydney, N.S.W., September 2000]
    </dc:description>
    <dc:date>2001-01-20</dc:date>
</rdf:Description>

</rdf:RDF>
```

Fig. 1. Sample Dublin Core metadata description.

When applying a technical view on these three blocks, we can regard both schema definition languages and metadata schemes as models. The former represents a set of language primitives (e.g., class, property, association), the latter a set of real world entities (e.g., person, name). Both define a precise semantics and syntax for their elements. We denote the model of a schema definition language as metadata *meta-model* and the model of a metadata schema as metadata *model*. Further, there is an instance-of relationship metadata instance, metadata model and metadata meta-model.

2.2 Heterogeneities impeding Metadata Interoperability

Metadata interoperability is a qualitative property of metadata information objects that enables systems and applications to work with or use these objects across system boundaries. In practice this means that a digital library system, for instance, can exchange bibliographic metadata descriptions with another system and both systems can correctly process these information objects regarding their syntax and semantics.

In reality, however, metadata interoperability is usually not given per default — especially when distinct systems have been developed independently from each other. As illustrated in Figure 2, we distinguish three main groups of heterogeneities that need to be considered for establishing metadata interoperability: *model-level structural*, *model-level semantic*, and *instance-level semantic heterogeneities*.

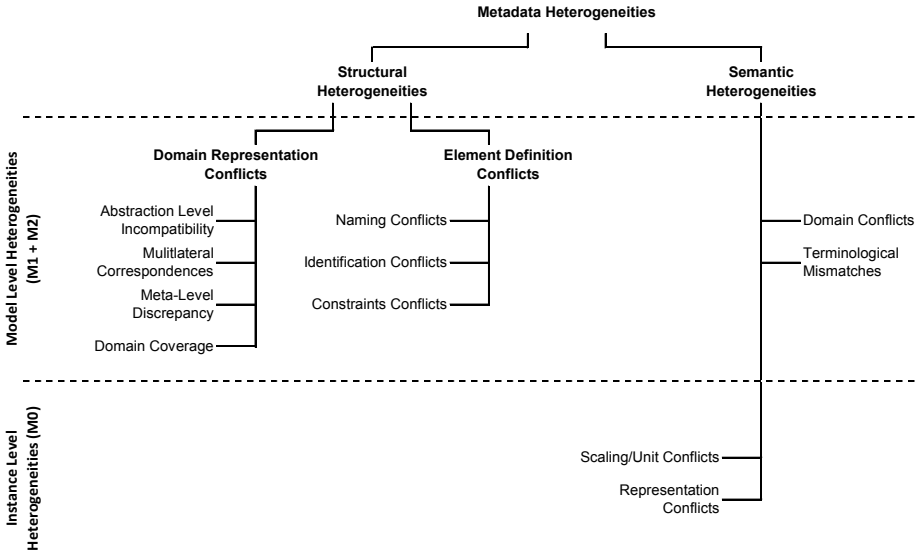

Fig. 2. Categorisation of metadata heterogeneities.

Model-level structural heterogeneities occur because the element of distinct models have assigned different names, identifiers, or conflicting constraints. We denote such heterogeneities as *element definition conflicts*. Model level structural heterogeneities also arise because of *domain representation conflicts*: domain experts represent the constituents of a domain in different generalisation hierarchies, using a different number and different types of elements.

Model-level semantic heterogeneities are caused by semantically overlapping, subsuming, or incompatible model elements (*domain conflicts*) and also by *terminological mismatches*, which occur whenever model elements with the same lexical name are mapped to distinct domain concepts (homonym) and vice versa (synonym).

Instance-level semantic heterogeneities appear when different scales (e.g., inch, centimetre) or different encoding representations (e.g., date-format) are used for content values.

2.3 Metadata Mapping

Metadata mapping is an interoperability technique for reconciling schema and instance level heterogeneities among metadata information objects, i.e. among distinct metadata models and their instance metadata information objects. In our context, mapping does not deal with heterogeneities among schema definition languages but assumes that metadata information objects are expressed in the

same language. If this is not the case, they must be transformed into the same language representation. On a technical level, metadata mapping has two facets:

First, a metadata mapping is a specification that relates the elements of two models in the same domain of discourse in a way that their schematic structures and semantic interpretation is respected on the metadata model and on the metadata instance level. A mapping is defined through a set of mapping relations between elements of a source and target schema, whereas a mapping relation is represented as a *mapping element*. The semantics of a mapping element is defined by a *mapping expression*. For reconciling instance level heterogeneities, a mapping element carries an appropriate *instance transformation* function. Figure 4 shows an example mapping specification consisting of a single mapping element that defines the semantic and structural correspondence between elements from a source schema S^s and a target schema S^t. The instance transformation function "concat()" concatenates the content values of the respective fields.

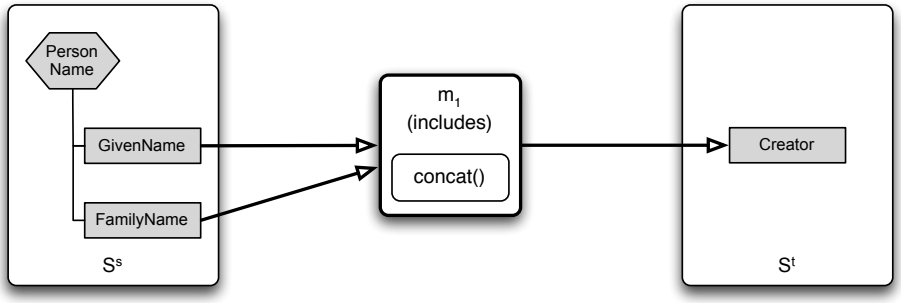

Fig. 3. Metadata mapping example.

Second, as illustrated in Figure 4, metadata mapping is a process consisting of a cyclic sequence of four phases: (i) mapping discovery, (ii) mapping representation, (iii) mapping execution, and (iv) mapping maintenance. We regard this sequence as being cyclic because through mapping maintenance it is possible to derive additional mapping between schemes. Throughout this work, we will concentrate in detail on each of these phases.

3 Evaluation Framework

In the subsequent presentation, we focus on requirements for metadata mapping solutions. They form the basis for the evaluation framework we use for our mapping tool analysis in Section 5.

We have organised the requirements into general requirements for mapping tools (Section 3.1) and requirements for each of the previously mentioned phases in the mapping process: mapping discovery (Section 3.2), mapping representation (Section 3.3), mapping execution (Section 3.4), and mapping maintenance

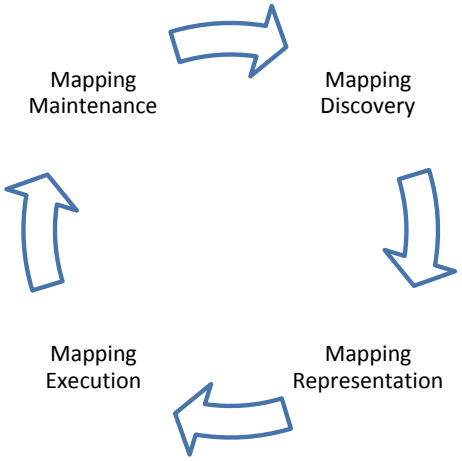

Fig. 4. Metadata mapping process.

(Section 3.5). Finally, in Section 3.6 we summarise the requirements in an evaluation framework for comparing existing mapping solutions.

3.1 General Requirements

Turning metadata mapping into practice requires the implementation of a mapping component, which is usually part of a larger metadata integration architecture. Like any other piece of software, such an integration architecture has some general requirements. First, it fulfils some architectural properties common to any metadata integration architecture. Second, it should be able to lift and normalise metadata represented in various schema definition languages to a common level; otherwise, if language mismatches are not resolved, metadata mapping is hardly possible. Third, and this is essential for creating mappings, it should provide a graphical user interface that supports domain experts in creating mappings.

Architecture Design The ultimate goal of any metadata interoperability technique is to achieve uniform access to metadata and digital media objects stored in multiple autonomous media repositories. Like other interoperability techniques, metadata mapping is just a prerequisite for achieving this goal. Therefore, the ultimate goal and the general requirement for any metadata integration architecture is *uniform accessibility* of metadata via a single interface. This requirement can be fulfilled by providing a query interface for a certain query language or by exposing a well-defined Application Programming Interface (API) for accessing the metadata.

Another basic requirement for an integration architecture is *modularity*. For each data source involved in an integration scenario, one has to implement an

adapter. Also the mappings and particularly the processing of the mappings must be embedded into a software component. An additional data source joining an integration scenario, should not affect the adapters of other data sources. Also the (re-)specification of mappings should not affect the implementation of software components; this should rather be a configuration task.

Domenig and Dittrich [6] give an overview of possible integration architectures. A very well known architecture for a modular integration approach are Mediated Query Systems, proposed by Wiederhold [7]. They consist of two types of software modules: mediators and wrappers. Each data source is encapsulated by a wrapper, which is program-specific for each data source. Its task is to accept queries from a mediator and to answer them on the basis of the underlying data source's technology. Mediators are modules that accept user queries formulated over a user-selected mediation model and reformulate them into subqueries according to previously defined metadata mappings. Then they disperse them to the local sources where they are executed, collect and combine the results and present them to the user in a certain format. Other architectural designs, especially Peer-to-Peer data management systems (e.g. Piazza [8], P-Grid [9], Edutella [10], Hyperion [11]), abstain from central mediators and mediation models and define point-to-point mappings directly between the models of the involved data sources (e.g. GridVine [12]). However, as Halevy et al. observe in [8], from a formal mapping perspective there is little difference between these two kinds of mappings.

Lifting and Normalisation As already mentioned in Section 2.3, metadata mapping postulates that all metadata information objects are expressed in the same schema definition language. This means, that all metadata models are internally represented as instances of the same metadata meta-model. Therefore, *lifting and normalisation* of metadata expressed in distinct languages to a common representation is another general requirement. A practical example from typical data integration scenarios, is the mapping of relational database schemes to XML Schema.

The LIFT tool which is part of the MAFRA ontology mapping framework [13] is an example for a component that fulfils this requirement. It provides means to lift DTDs, XML Schemes, and relational databases to a common structural ontology level.

Graphical User Interface Metadata mapping cannot be fully automated and will always depend on interactions with domain experts. Usually it is not the technicians who define semantic relations between schema elements, but expert users such as librarians, curators, or knowledge workers. Technicians are concerned with the technical implementation of mappings. Based on the semantic relations, they reconcile structural heterogeneities among schemes and concentrate on the implementation of instance transformation functionality. For both aspects, the domain expert view, and the technician view, a mapping solution should provide a *graphical user interface (GUI)* to support the domain experts

as well as the technicians in their mapping tasks. Especially the domain expert view must have a simple structure and guide the users' attention to the relevant places, especially when large metadata schemes need to be mapped.

Robertson et al. [14] present several advanced visualisation methods for larger schemes. Following and highlighting existing links when a model element is selected is one of the proposed techniques, auto-scrolling during typing and an incremental search mechanism are other examples.

Another useful design strategy is to separate between a schema and a data view, as it is implemented in Clio [15, 16]. The schema view represents the main perspective for mapping definition. To get further information on a schema element's semantics, the user can switch to the data view and retrieve sample data for that element.

The majority of mapping solutions is implemented as stand-alone desktop applications. However, recently they have also begun to emerge on the Web, Yahoo Pipes [17] being the prime example for this category. Via a simple, intuitive Web-interface users can aggregate XML data and define mappings between model elements. Provided that a Web-based mapping solution is simple enough, this category of mapping solutions can attract a large number of users, lead to collaborative mapping efforts, and give insights on the different semantic perceptions of model elements.

3.2 Mapping Discovery Requirements

The first metadata mapping step is to determine the relations between a source and a target schema. In the literature the task of discovering the mappings is also denoted as *matching*, *mapping*, and especially in combination with ontologies, *alignment*. Although finding the matching elements is an intellectual task, which is mainly carried out by humans, there are automated approaches to support users in mapping discovery. When many users are involved in determining the right mappings, building consensus on the defined mappings is essential.

Matching / Alignment Support The larger metadata schemes are, the more difficult it is to find the set of potential mappings between schema elements. Fully automatic mapping between schemes is considered to be an AI-complete problem, that is, as hard as reproducing human intelligence [18]. Although this problem is not yet solvable, there are many (semi-)automatic techniques that can support the user in matching tasks. Hence, a mapping solution should provide some mechanism for the automated resolution of semantic correspondences between the elements of heterogeneous metadata schemes; we call this requirement *alignment support*[1].

[1] On the Web, there are two sites which provide up-to-date information on the research topic of ontology alignment: the Ontology Alignment Source (`http://www.atl.lmco.com/projects/ontology/`) and OntologyMatching.org (`http://www.ontologymatching.org/`)

The term "alignment" frequently occurs in the context of "ontology alignment" and is used interchangeably with the term "ontology mapping", wherefore Kafoglou and Schorlemmer [19] provide a survey of (semi-)automatic techniques. Rahm and Bernstein [20] provide a survey of approaches in the database domain, Shvaiko and Euzenat [21] analyse both schema and ontology mappings, and Doan and Halevy [22] analyse mapping discovery solutions from a data integration perspective. Most of the approaches cited in these surveys are based on heuristic algorithms comparing the lexical and structural features of models (e.g., PROMPT [23]), or employ machine learning techniques to find mappings (e.g., GLUE [24]). Some approaches operate either on the schema level (e.g., Cupid [25]) or only on the instance level (e.g. SemInt [26]), recent developments (e.g., COMA++ [27]) include both levels in order to automatically discover mappings between models.

Consensus Building When metadata schemes are developed independent from each other, their structural and semantic properties are likely to be different and lead to the heterogeneities we have discussed earlier in this work. This is also the case when two schemes are mapped. As their semantic interpretation, also the mappings between two metadata schemes are always bound to a certain context. A mapping created by person A might not necessarily be true for person B. Therefore, *consensus building* is a principal requirement for any integration scenario. The prerequisite for consensus building is a precise definition and documentation of the semantics of the schemes to be mapped. If only one person is involved in establishing the mappings, no tool support is required for consensus building. Larger scenarios involving several persons and extensive metadata schemes to be mapped will benefit from tool support for consensus building.

Zhdanova and Shvaiko [28] propose a public, community-driven approach for mapping discovery where end users, knowledge engineers, and developer communities take part in the process of establishing mappings. The resulting mappings are handled as "subjective alignments", hence as mappings that are customised to a certain user, a community, or application requirements. From already existing mappings, and the information about users, communities, groups, and social networks, the system can determine valuable information for mapping discovery. For instance, if several users have mapped their user defined schemes with the same common mediated scheme, the system can leverage past experience and propose mappings for new mapping tasks.

Another approach which supports consensus building is the ESP game, proposed by von Ahn and Dabbish [29]. Although being developed for another purpose, which is the labelling of online images, it demonstrates how a game like approach can motivate people to build a consensus on the semantics of an online resource. If we consider metadata schemes and mappings as online resources, such game-like approaches could open the door for novel consensus building techniques.

3.3 Mapping Representation Requirements

After metadata mappings have been discovered, they must be represented in a machine read- and interpretable way. A semantically well-defined formalism ensures that mappings can actually be processed in the subsequent mapping phases. Since the decision whether a metadata mapping is semantically correct, depends on the context, the formalism must provide the means to represent the corresponding context. Finally, the properties of a metadata mapping largely depends on the schema definition language used for describing metadata schemes. Therefore, a mapping formalism should be flexible in its *language bindings*.

Mapping Formalism For reconciling heterogeneities by means of metadata mappings, one needs to formally declare mapping relations between the elements of two metadata schemes. A set of such relations between two schemes is denoted as *mapping specification*. A mapping formalism builds the basis for mapping specifications. It provides a machine read- and interpretable language for creating mappings and, as it is the case with schema definition languages, a concrete and an abstract syntax. The concrete syntax (e.g. a serialisation in XML or the graphical illustration of a mapping element) allows the serialisation, registration, and exchange of mapping specifications and also enables human readability. The abstract syntax, i.e., the meta-model of mapping specifications, represents a semantically well-defined corpus for mapping specifications and ensures the correct interpretation of mappings across machines and system boundaries.

The *strength* of a mapping formalism denotes its ability to express those relations that are required to reconcile the various kinds of heterogeneities, mentioned in Section 2.2. We have distinguished between two levels (model and instance level) and two main families of heterogeneity (structural and semantic). In combination with instance transformation, metadata mapping can reconcile a broad range of heterogeneities: naming, identification, and constraints conflicts as well as abstraction level incompatibilities, multilateral correspondences, domain coverage conflicts, terminological mismatches and meta-level discrepancies can be resolved through mappings on the schema level. Instance transformation, if it is an integral part of a mapping formalism, can resolve the remaining semantic heterogeneity conflicts on the instance level.

In literature we can find many approaches that support mappings among metadata schemes. Unfortunately, many of them do not consider the whole heterogeneity spectrum but focus mainly on schema level mappings and disregard the instance level. Observer [30], for instance, only allows the specification of "synonym", "homonym", "overlap", "disjoint", and "overlap" relations between entities of metadata schemes. Xiao and Cruz [31] have defined a mapping language for P2P systems, which provides one-to-one mappings such as "equivalent", "broader", "narrower", "union", and "intersection" between schemes. Even less expressive is GridVine [12] which relies on the very restricted set of built-in OWL mapping primitives (e.g. "owl:equivalentProperty"). Although these kind of mappings suffice for human interpretation, the question remains

how machines can provide uniform access to the sources. They need exact information about relations between concepts and precise processing instructions for dealing with the instances or data originating from heterogeneous sources.

The MAFRA ontology mapping framework [13] is an example for a system that covers the whole heterogeneity spectrum through the definition of "semantic bridges". Piazza [8] is a representative for the family of integration systems that uses queries (views) as representation mechanism for mappings. This approach is well known from the domain of relational databases and is, depending on the expressiveness of the query language, also suitable for other technologies (e.g., XML, XQuery).

Mapping Context Like metadata schema definitions, also metadata mappings are formal declarative specifications that are subject of interpretation. More specifically, metadata mappings define how heterogeneity conflicts, when they are detected, should be resolved. The problem is that even within a single metadata integration scenario, data sources as well as the mappings created between their metadata schemes and instances may embody different assumptions on how information should be interpreted. If, for instance, one schema A is mapped to two schemes B and C, which differ in their semantic domain, an element of A could be interpreted differently in relation with elements of B than it is interpreted in relation with elements of C. If mappings are created across integration scenarios, the importance of context grows. Therefore, it is necessary to capture the *mapping context*, hence the setting in which the interpretation of a metadata mapping is semantically correct. A mapping formalism should provide support for capturing mapping context.

COIN [32] was one of the early information integration system that has explicitly incorporated context into mapping definitions. With the "context interchange framework", it provides a formal, logical specification for modelling metadata models, axioms for identifying correspondences between model elements, and context axioms, that permit the specification of named contexts and therefore the definition of alternative interpretations of information objects. Based on this work, Wache [33] has provided an integration formalism which allows not only the representation of context but also the transformation of information objects between contexts, in order to reconcile semantic heterogeneities even across contexts.

Language Binding As described in Section 2.1, metadata information objects are instances of metadata models which, can be expressed in various schema definition languages. A metadata scheme for describing books, for instance, can be, depending on an application's needs, expressed as a model in Java or XML Schema. If mappings between schemes are established, the resulting mapping is always bound to a certain language.

For expressing and representing mappings, one needs a formalism which can either be language independent or bound to a certain schema definition language.

The advantages of a language independent or generic approach are its applicability for other schema definition languages. The main disadvantage is the increased complexity and additional development effort: if a mapping tool follows a language independent approach, hence is open to any schema definition language, a generic metadata meta-meta model must be defined and language-specific extensions must be built. However, we believe that *flexibility* in the language binding is necessary to support the variety of existing schema definition languages and for being open to future developments.

With the Ontology Definition Metamodel Specification (ODM) [34], the Object Management Group (OMG) has defined a set of metadata meta-models that reflect the abstract syntax of RDF, OWL, Common Logic (CL), and Topic Maps. In addition, mappings between a number of the meta-models, being expressed in the MOF QVT Relations Language [35] are provided. The goal of this approach is to support interoperability between MOF-based modelling tools independent of the schema definition language they support. Since metadata mapping tools can be regarded as a feature extension of modelling tools, it is worthwhile to consider MOF based strategies for implementing mapping solutions.

MAFRA [13], a mapping framework that enables the transformation of instances of source ontologies into instances of target ontologies is an example for a language dependent approach that expresses mappings as instances of a meta-ontology, which in this case is expressed in DAML-OIL [36]. We can find language dependent bridging axioms in ontology languages such as OWL which provide language primitives to define semantic relations (e.g. equivalent, sameAs) between ontology concepts. RuleML [37] is an effort to develop a rule language that is independent of any schema definition language. Semex [38] is an example for a system which formulates mappings between schemes in terms of queries. In other words, elements from one schema model are defined as a query over the elements in another schema model.

3.4 Mapping Execution Requirements

"We have the mappings. Now What?" With this question, Noy [39] points out that the definition and representation of mappings is the necessary precondition but not the goal in itself. The ultimate goal of metadata mapping is to achieve uniform access to metadata in multiple autonomous information systems.

In the mapping execution phase, mappings are used for reformulating queries over one schema into queries over another schema, for calculating query plans for query optimisation, and for generating stubs or wrappers to data sources.

Query Reformulation Users of an integrated and interoperable system environment should have the possibility to formulate queries over a user selected metadata schema and receive results from a set of integrated data sources, each potentially employing a different metadata schema. Hence, the integration system must convert the queries formulated over the user selected schema into terms of the data sources' metadata schemes. Metadata mappings are the technical

specifications that serve as input for this process, which is commonly referred to as *query reformulation*.

If we regard metadata mapping as definitions that describe how to construct the elements of a target schema from the elements in a source schema, they fulfil the same functionality as views. Previously, we have already mentioned that using views is a common formalism for representing mappings. Hence, if mapping specifications are not available in terms of views per se, as it is the case in systems such as Piazza [8], they can be transformed into such a representation. In principle, there are two ways of representing mappings using views: (i) the data sources, i.e. their schema elements, are described as queries (views) over the user selected schema — this is referred to as *Local as View (LaV)* — or (ii) the user selected schema is described as a set of views over the data sources — this is known as *Global as View (GaV)*. In the first case query reformulation means rewriting the queries similar to rewriting a query using a view [40]. In the second case, reformulation works analogously to view unfolding in traditional relational database systems.

Rajaraman et al. [41] took an approach similar to view-based query reformulation. They define *query templates* in a global query language consisting of a head and a body. The head contains a predicate denoting the view, arguments for the predicate, and binding patterns which indicate which arguments of the predicate are expected to be bound and which are free. The body is a program in some notation that produces a result to the query.

Query Plan For efficiently accessing integrated data sources the system must calculate a query plan and use it to optimise query reformulation. Usually, the main goal of a query plan is to reduce the execution time of queries on the data sources. There are two main tuning possibilities to achieve that: first, by avoiding redundant queries, i.e. queries that return a subset of results of a previously executed query. This is also known as *query containment* [42] and is a metric that can be calculated prior to query execution. Second, query time can be reduced by analysing the *capabilities* [43] of the involved data sources. The capabilities of a data source depend on its physical properties such as network connection speed, average response time, but also on logical properties, such as the availability of data in a data source[2]. Another optimisation goal that can be achieved by a query plan is the *minimisation of response time*. This reduces the time it takes until the first query response is returned from a data source, which is important especially in distributed environments, such as P2P systems.

Calculating query plans and optimisation techniques are well studied in the domain of traditional database systems. Jarke and Koch [44] provide a survey of available techniques. Tatarinov and Halevy [45] propose optimisation techniques for Peer-to-Peer Data Management Systems and describe an algorithm for calculating XQuery query containment and optimisation algorithms for pruning of redundant queries and for minimising the required query reformulations.

[2] Even if a certain entity is represented in a data source's metadata schema, this doesn't guarantee that the instance data are available.

Integration Component Generation In order to enable a data source to be integrated with other sources, all the previously mentioned requirements must be packed into data source specific integration components. If a mapping solution follows a federated architecture, such components are typically *mediators* or *wrappers*. In other environments, *stubs* or *adapters* are required to access the metadata contained in third party data sources according to previously defined mappings.

To avoid manual implementation of integration components for each data source, it is desirable to have some at least semi-automatic mechanism that supports users in setting up such components. Although it will always be necessary to perform certain source specific adoptions, at least some generic aspects (e.g. query reformulation) can be realised automatically. Such a feature could, for example, be embedded in a mapping tool and allow the generation of source code skeletons based on mapping specifications.

TSIMMIS [46] is one of the early systems which automatically generates mediator as well as wrapper components based on high level descriptions of the information processing they need to do. Another approach, which is also relevant for capability based optimisation, is to transform the mapping specifications into *templates*, which represent the possible queries a mediator can ask. Wrapper Generators [47] can be applied to create a table holding the various query patterns contained in the templates; since it is not realistic to create a template for each possible form of a query, the wrapper must include a mechanism which works with query containment. It must determine if the result of a specific query pattern is included in a broader query and filter the results accordingly.

3.5 Mapping Maintenance Requirements

Together with the metadata schemes involved in an integration scenario, also mapping specifications can be registered in a kind of *metadata registry*. If a registry maintains several metadata schemes and mappings between these schemes are available, it is possible to verify mapping specifications by detecting potential conflicts with other mappings, to reuse existing mappings for the same or similar metadata schemes, or to infer potential, not yet explicitly expressed mapping relations.

Mapping Verification Although the "truth" of a mapping always depends on an interpretation bound to a certain context (e.g., the one of the mapping creator), it is possible to detect potential conflicts with other already existing metadata mappings. Although resolving conflicts automatically is difficult, it is at least possible to provide some kind of notification mechanism. In general, mapping verification can improve the quality of mappings and could also contribute to building consensus on mappings in the context of a certain application domain.

In Clio [15, 16], mappings are verified by presenting alternative mappings to the user. They see example data from a selected source and of the target as they

would appear under the current mapping. This should illustrate a given mapping and the perhaps subtle differences between other mappings.

Mapping Reusability Mappings can potentially be reused for future integration tasks. A metadata registry can analyse the relationships between existing schemes and mappings and use heuristics to identify similarities. Based on that the system can, for instance, propose possible mappings to domain experts. Especially with a growing number of schemes and mappings, mapping registries play an important role for determining the reuse potential of mappings.

Mapping Inference Based on existing mapping relations, a mapping registry could also infer not yet explicitly available mappings. Suppose we have three metadata schemes A, B, and C. If there is mapping between the elements of A and B, and B and C, it is possible to exploit these transitive relations and explicitly express mapping relations from A to C. In another setting, if for instance some elements of B are "subclasses", or "subproperties" of elements of A, and there exist mapping relations between the elements of A and C, the system can automatically infer relations between the elements of B and C.

GridVine [12] and Piazza [8], both belonging to the domain of Peer-to-Peer data management infrastructures, are systems that perform a kind of transitive mapping inference. Each node in the system has mappings to a small set of nodes. When a query is posed over a node, it transitively follows the nodes that are connected by semantic mappings (*chaining mappings*).

3.6 Requirements Summary

In this section, we have outlined the set of requirements which are, in our conception, the essential ingredients for building mapping solutions. We do not consider them as obligatory features that must be fulfilled by any mapping solution, but rather as complementary building blocks, some of which are more essential than others. A mapping solution could, for instance, exist without providing alignment support for end users; but not if there was no formalism for representing mappings.

Through the arrangement of the requirements along the four phases of the metadata mapping cycle, we want to emphasise the importance of considering metadata mapping as a process rather than a single step. Mapping discovery, for instance, is not enough to achieve the goal of uniform accessibility. It requires a formalism that can capture the various heterogeneity aspects and a mechanism for executing mapping specifications. Also having some kind of mapping registry gains great importance when the number of data sources and metadata schemes to be integrated grows.

Different from many other approaches in the field of metadata mapping, we discuss mapping discovery only superficially: we only distinguish between solutions that support users in discovering mapping relations and those that do

not. For further details on this topic we refer to the surveys of Rahm and Bernstein [20] as well as Kalfoglou and Schorlemmer [19]. As depicted in the requirements summary in Figure 5, the main focus of our analysis lies on the mapping process as whole and especially on the strength of the mapping formalisms.

	Requirement	Description
General	Uniform Accessibility	Provide access to a set of (distributed) metadata sources via a single access point
	Modularity	Adding additional sources and mappings without affecting/changing existing system components
	Lifting & Normalisation	Flexible means to convert metadata expressed in distinct schema definition languages to a common metadata meta-model
	Mapping GUI	A graphical user interface supporting domain experts and technicians in creating mappings
Mapping Discovery	Schema Matching / Alignment Support	(Semi-)automatic support for determining mappings; either on the schema, the instance, or on both levels
	Consensus Building Features	Providing features that support users in building consensus on conflicting mappings
Mapping Representation	Model-Level Structural Heterogeneity Reconciliation	The ability to represent and reconcile structural heterogeneities on the model level
	Model-Level Semantic Heterogeneity Reconciliation	The ability to represent and reconcile semantic heterogeneities on the model level
	Instance-Level Semantic Heterogeneity Reconciliation	The ability to represent and reconcile semantic heterogeneities on the instance level
	Context Representation	The ability to capture the interpreation context of a metadata mapping
	Flexible Language Binding	A generic, language independent mapping formalism that can easily be bound to a certain schema definition language
Mapping Execution	Query Reformulation	Reformulate queries over a user selected schema into queries over the target schema representation according to mapping definitions
	Query Plan / Optimisation	Algorithmic support for minimising the query execution / response time
	Integration Component Generation	(Semi-)automatic generation of data-source specific integration components (mediators, wrappers, adapters, etc.)
Mapping Maintenance	Mapping Verification	Detection of potential conflicts with other mappings
	Mapping Reusability	System support for reusing existing mapping definitions
	Mapping Inference	Infer not yet explicitly expressed mapping relations from existing mapping specifications

Fig. 5. Requirements framework for the evaluation of metadata mapping solutions.

4 Mapping Solutions

Mapping solutions are the means to achieve metadata interoperability in integration scenarios where *mapping* has been chosen as being the appropriate technique. We can regard them as technical manifestation of the previously mentioned mapping phases (discovery, representation, execution, maintenance) — though it depends on the mapping solution how and to what extent it supports a certain phase.

There exists a multitude of mapping solutions with varying mapping capabilities, different stages of stability and distinct underlying business models. In this section we present a categorisation for a representative while not complete selection of mapping solutions. It includes tools from major software vendors that are active in the domain of data integration, such as BEA, IBM, Sybase, Microsoft, Cape Clear, Altova, and Data Direct, as well as frequently cited research projects, and novel Web-based solutions such as Yahoo Pipes. The focus of our evaluation is on solutions and tools; theoretical mapping approaches lacking an at least prototypical implementation are therefore not part of this evaluation. We will describe each mapping solution in detail in Sections 4.2 to 4.5 and conclude with preliminary observations in Section 4.6.

4.1 A Categorisation of Mapping Solutions

For a coarse-grained categorisation of mapping solutions we fall back on their architectural properties, which are a direct result of the application domain they were designed for. In industrial, large scale environments metadata mapping solutions are an integral part of *Enterprise Information Integration (EII)* and *Enterprise Application Integration (EAI)* suites. EII and EAI systems are usually extensive and heavyweight software suites where mapping is usually only a small subset of the supported features. Besides that, we have also identified the category *mapping tool*, which contains lightweight standalone tools designed for the sole purpose of mapping. The category *other solutions* contains all mapping tools that cannot be assigned to any of the previously mentioned categories, such as XML editors or modelling tools that may also provide mapping support for certain kinds of schema definition languages, or Web applications.

Figure 6 gives an overview of available mapping solutions. For each solution we describe what type of software it is (commercial, research prototype) and to which solution category it belongs (EII suite, EAI suite, mapping tool, other). Furthermore, we analyse for what kind of schema definition languages a solution provides mapping capabilities and which software platforms are supported. Additionally we briefly sketch which mapping phases are covered by a certain mapping solution.

4.2 Enterprise Information Integration (EII) Suites

EII subsumes industrial solutions dealing with the problem of data integration. Generally, they aim at (i) identifying data sources, (ii) building virtual schemes,

		General Properties			Mapping Phases Support			
		License Type	Schema Definition Language Support	Supportted Platforms	Discovery	Representation	Execution	Maintenance
EII Suites	BEA Liquid Data 8.1	Commercial	XML Schema	HP-UX, MS Windows 2000/XP, Red Hat Linux, Sun Solaris, IBM AIX	□	■	■	■
	Sybase Data Integration Suite - Avaki (Studio/Server) 7.0	Commercial	Proprietary	Red Hat/Suse Linux, Windows 2003/XP, Sun Solaris, IBM AIX	□	■	■	■
EAI Suites	Microsoft BizTalk Server 2006	Commercial	XML Schema	Windows 2003/XP	■	■	■	■
	Cape Clear 7 (Studio/Server)	Commercial	XML Schema	Red Hat Linux, Sun Solaris, Windows 2003/XP	□	■	■	□
	IBM WebSphere Integration Developer	Commercial	Proprietary, XML Schema	Red Hat/Suse Linux, Windows 2000/2003/XP	■	■	■	□
Mapping Tools	Altova MapForce / SchemaAgent	Commercial	Proprietary	Windows 2000/2003/ XP/Vista	■	■	■	■
	COMA++	Research	Proprietary	Java	■	■	□	■
	Clio	Research	Proprietary	Java	■	■	□	□
Other Solutions	StylusStudio / DataDirect XML Converters	Research	XML Schema	Windows Platforms	□	■	□	□
	TopBraid Composer	Commercial	OWL	Java	□	■	□	□
	Yahoo Pipes	Commercial	XML	Windows 2000/XP	□	■	■	■

■ Supported
□ Not Supported

Fig. 6. Metadata Mapping Solutions — Categorisation and Overview.

and (iii) reformulating queries over a virtual schema into queries over multiple data source specific schemes. EII systems provide real-time information by integrating heterogeneous data sources on demand without moving or replicating them [48]. In such solutions, mappings are generally represented as views and executed through a query rewriting mechanism. To what extent mapping discovery and maintenance is supported depends on the solution.

BEA Liquid Data 8.1 BEA Liquid Data for WebLogic [49] is an Enterprise Information Integration suite allowing a real-time unified view over heterogeneous data sources such as relational databases, XML data, flat files (e.g., CSV-files),

and third party application data. It leverages XML standards throughout the mapping phases and also delivers the query results structured in XML. Liquid Data is available for all major software platforms.

While not supporting the discovery phase of the mapping process, BEA Liquid Data offers a view-based approach for the mapping representation phase: all supported schema definition languages are lifted to the level of XML Schema. The mappings between source and target schemes are expressed in XQuery. For executing mappings, BEA provides the Liquid Data Server for deploying mapping specifications and a distributed query processor for translating queries according to the mappings. To a certain extent, BEA Liquid Data also supports the mapping maintenance phase: schemes as well as mappings between schemes are stored in the Liquid Data Repository, which enables the reuse of mappings.

Sybase Data Integration Suite — Avaki Studio / Server 7.0 The Sybase Data Integration Suite [50] with its components Avaki Studio and Avaki Server 7.0 is a data federation solution which provides standardised access to distributed heterogeneous data through a single layer. It supports the integration of the same types of data sources as BEA Liquid data, but follows a mapping approach based on the relational model. Both Avaki Studio and Server are available for all major software platforms.

Mappings are created using the Avaki Studio application — support during the mapping discovery phase is not provided. For mapping representation Avaki defines a mapping model based on the relational model. One or more input sources, a single result element, and a set of operators which can be arranged sequentially to combine or transform data from one or more input sources, are the main constituents of that mapping model. For the mapping execution phase, the mapping models are deployed as data services exposing SQL-DDL (view) schema definitions. All data services and mapping models are registered and maintained in a data catalogue.

4.3 Enterprise Application Integration (EAI) Suites

EAI systems deal with the problem of integrating applications through business processes. They are well known in the context of *Service Oriented Architectures (SOA)* where they connect and integrate loosely coupled, distributed software components usually by using Web Services in order to fulfil a certain business task (e.g. booking a flight). Since the applications involved in a business process usually provide metadata corresponding to various, incompatible schemes, also EAI systems require mapping techniques for resolving these discrepancies.

Different than EII systems, which can query a set of heterogeneous sources via a virtual unified target schema, the focus of EAI systems is on the exchange of data residing at multiple sites. Hence, during the execution phase, mappings do not serve as input for query reformulation but for transforming data from a source into a target schema.

Microsoft Biztalk 2006 Microsoft BizTalk Mapper [51] is part of the BizTalk Server Enterprise Application Integration suite and allows to create and edit mappings to translate or transform messages within business processes from one format into another. Since BizTalk is a pure EAI suite and XML has evolved as de facto standard for structuring messages within business processes, mapping is supported among XML schema definitions. Like the whole BizTalk suite, also the mapper is available only for Microsoft Windows 2003 and XP platforms.

The mapping discovery phase is not supported by the Microsoft BizTalk Mapper, meaning that expert users must find the mapping relations among the source and target schemes without technical assistance. Schemes as well as mappings are represented in a BizTalk-specific mapping model and are transformed to XSL style-sheets during the mapping executing phase. For maintaining mappings, BizTalk relies on a simple WebDav repository where mappings can be published and reused in other integration tasks.

Cape Clear Studio and Server 7 The Cape Clear EAI suite [52] comprises two main products: the Cape Clear Server and the Cape Clear Studio. The former provides the environment for deploying business processes and Web Service components. The latter is a design tool for creating Web Services and BPEL processes. Data Transformation Web Services are a special kind of service: they permit the integration of non-XML data sources that represent data as structured text (e.g., CSV, EDI, SWIFT). However, they first must be lifted to the level of XML Schema in order to be mappable with other message formats within a business process. Cape Clear Studio and Server are both Java applications — the Studio is an Eclipse Rich Client Application — and therefore run on all major software platforms.

Like all mapping solutions mentioned so far, also the Cape Clear Studio does not support mapping discovery. Mappings must be defined manually between XML Schema definitions using the Cape Clear Studio XSLT mapper. This implies that mappings are represented as XSL style-sheets. For the execution of mappings at run-time, a Data Transformation Service can be deployed on the Cape Clear Server and be incorporated into any business process. So far the mapping maintenance phase is not supported because there is no possibility to publish or share the created mappings.

IBM WebSphere Integration Developer 6.0.2 IBM's contribution to the market of EAI suites is the WebSphere platform, an environment for deploying reusable business processes on a Service Oriented Architecture (SOA) foundation. The IBM WebSphere Integration Developer [53], a tool for modelling business processes, also has mapping capabilities. Since *business objects* are the WebSphere internal representation of application data, the Integration Developer supports mappings among instances of that proprietary model. Additionally it also supports the development of mediation services, which are services that intercept and modify messages passed between existing services within a business process. For this kind of service, the Integration Developer provides XML

Schema mapping capabilities. As the whole WebSphere EAI suite, also the IBM WebSphere Integration Developer is available for all major software platforms.

Although the extent is minimal, we can categorise the IBM WebSphere Integration Developer as solution that supports the mapping discovery phase: it can automatically create mapping relations among attributes having the same lexical name. Mappings are represented either in a proprietary model (business object maps) when they are created between business objects, or as XSL style-sheet when XML Schema definitions are mapped. They can be deployed as software modules on a WebSphere Process Server or a WebSphere Enterprise Service Bus. Mapping maintenance is not supported.

4.4 Mapping Tools

Different from EAI and EII suites, where mapping solutions are only part of a broader application infrastructure, mapping tools are lightweight standalone systems created for the sole purpose of mapping. The market for this category of mapping solutions is still sparsely populated; Altova with its products MapForce [54] and SchemaAgent [55] turned out to be the only well-known commercial representative.

Altova MapForce and SchemaAgent 2008 At the time of this writing, Altova MapForce in combination with Altova SchemaAgent is the most powerful mapping solution on the market. MapForce supports mapping between any combination of SQL-DDL definitions in relational databases, XML Schema and DTD declarations, and flat files formats such as CSV or EDI. The main drawback is that these tools are currently available only for Microsoft Windows platforms.

MapForce assists the user during the mapping discovery phase by automatically matching child elements of already mapped elements. In contrast to any other mapping solution mentioned so far, MapForce allows the definition of mappings among several kinds of schema definition languages (XML Schema, SQL-DDL, etc.); this implies that internally MapForce has a common, generic representation for these kind of schemes and also for the mapping between them. For the mapping executing phase, it provides the possibility to generate code from these proprietary mapping representations; a mapping specification can be compiled into XSLT code, XQuery, Java, C++, and C#. The mapping maintenance phase has been completely outsourced to Altova SchemaAgent, which is another standalone product that works in combination with Altova MapForce. Besides capabilities for registering schemes and mappings, it provides graphical means to view and analyse dependencies between them.

COMA++ The COMA++ [27] research prototype, an extension of COMA [56], is a generic schema mapping tool. It supports the user in the mapping discovery phase by matching schemes expressed in SQL-DDL, XML Schema, XML Data Record (XDR), or OWL. The tool is written in Java and is therefore platform-independent.

The focus of COMA++ is on the mapping discovery phase; it allows the combination of a variety of matching algorithms in order to find appropriate mappings between schemes. Internally, the schemes are uniformly represented as directed graphs, i.e., also the mappings are represented in a proprietary format. Currently, COMA++ does not support the mapping execution phase, neither for query rewriting nor for transforming models from one schema to another. For the mapping maintenance phase, COMA++ provides a repository component which centrally stores schemes, matching results, and mappings between schemes. An outstanding feature in this phase is the ability to derive new mappings from previously determined matching results.

Clio Clio [15, 16] is an IBM research prototype for creating and executing mappings among schemes expressed in SQL-DDL or XML Schema. It is implemented in Java and therefore platform independent.

For the mapping discovery phase, Clio relies on a semi-automatic tableaux-based algorithm which calculates all the possibilities in which schema elements relate to each other and prompts the user to select the semantically correct relation. Internally, mappings are represented as an abstract query graph which can be serialised into specific languages such as XQuery, XSLT, and SQL/XML. The execution of queries in the mapping execution phase is left to the user. Clio does not provide any mapping maintenance support.

4.5 Other Solutions

Mapping support may also be a feature of solutions that do not belong to one of the above mentioned categories. In the following we briefly discuss tools and applications offering mapping capabilities.

Stylusstudio 2007 / DataDirect XML Converters Stylusstudio [57] is an XML Integrated Development Environment (IDE) which supports mappings among schemes expressed in SQL-DDL, XML instance documents, XML schemes, DTDs, and EDI documents. In combination with DataDirect XML Converters, it allows the conversion of any legacy data format into XML. At the moment, Stylusstudio is available for Windows platforms only.

The mapping discovery phase is not supported by Stylusstudio. For representing mappings, Stylusstudio relies on standardised XML technologies and compiles mapping relations drawn on the user interface directly into XSLT or XQuery; hence it does not define a proprietary mapping model. Executing the XQuery code on a certain data source or transforming XML documents using the resulting XSL style-sheet is left to the user. With Stylussstudio it is currently not possible to deploy mapping services covering the mapping execution phase. The mapping maintenance phase is also unsupported.

TopBraid Composer TopBraid Composer [58], which is the commercial extension of the Protégé OWL editor[3], is a Semantic Web ontology development platform which also supports the creation of mappings between ontologies. The tool has been implemented in Java, using the Eclipse Rich Client Platform and therefore runs on any Java-enabled software platform.

In version 2.3.0, TopBraid does not support the mapping discovery and maintenance phases. However, it is possible to create mappings between ontologies, which can be compiled either into SPARQL[4] query construct statements or into SWRL[5] rules. Support for the mapping execution phase is currently not provided.

Yahoo Pipes Yahoo Pipes [17] has introduced a novel facet into the topic of metadata mapping. Through a Web application, users can aggregate data from various sources such as XML feeds, online CSV files, or other online applications (e.g., Flickr, Google and Yahoo search results) and deploy new data services (mashups) that provide uniform access to these sources. This also includes mapping between the various source formats — a task which is supported by Yahoo Pipes by an intuitive, easy-to-use online interface. Since all Yahoo Pipes features are part of a Web application, this mapping solution can run in an ordinary Web browser on any platform.

Mapping discovery is the only phase not supported by Yahoo Pipes; the users have to identify the semantic and structural correspondences on their own. For representing metadata (e.g., XML tags of feeds), Yahoo Pipes relies on a very simple internal model consisting only of elements and sub-elements. Thus, it does not regard any schema definitions, which might not even exist for certain data sources, but concentrates on the instance level. Users only create mappings between elements without being confronted with complex schema definitions. For representing the mappings between the data elements, Yahoo Pipes employs its own proprietary model which defines a collection of modules for assembling data transformation pipes. That Yahoo Pipes also supports the mapping execution phase becomes obvious when creating a pipe: during the modelling process, each mapping module automatically executes itself and presents the resulting instance data to the user. Each pipe created is stored and optionally published on Yahoo Pipes — the maintenance phase is therefore supported. Other users can copy existing pipes and adapt them to their needs or include the resulting instance data of an existing pipe as data source in their own pipe. These features enable even non-expert users to collaboratively assemble pipes and create mappings in a trial-and-error manner, a frequently applied strategy[6] in the Web context and also a main factor for its success.

[3] Protégé OWL editor: `http://protege.stanford.edu/overview/protege-owl.html`
[4] SPARQL Query Language for RDF: `http://www.w3.org/TR/rdf-sparql-query/`
[5] Semantic Web Rule Language (SWRL): `http://www.w3.org/Submission/SWRL/`
[6] Many users create Web sites by simply copying and adapting other already existing Web sites. This is possible because the source code of any Web site is available.

4.6 Preliminary Observations

In this section, we have given an overview of existing mapping solutions and categorised them according to their application domain. We have outlined their basic architectural properties, their mapping capabilities in respect to schema definition languages, and the software platforms they support. Additionally, we have outlined what mapping solutions support which metadata mapping phases. Before we perform an in-depth analysis of each mapping solution in respect to the supported mapping phases, we conclude this section with some preliminary observations.

First, only Altova MapForce in combination with Altova SchemaAgent as well as Microsoft BizTalk Server support all four mapping phases. Other tools lack at least one phase. Especially mapping discovery, an issue which has gained much attention in scientific literature, has not yet found its implementation in commercial mapping solutions — and if at all, then only to a minor, rather trivial extent such as the comparison of schema element names.

The second observation is that many solutions use their own proprietary models for representing schemes and mappings among them. Additionally, we notice that there is a strong support for XML technologies (XML Schema and DTD) and the relational model (SQL-DDL); support for other technologies is rather minor.

Third, we can observe that most mapping solutions support the mapping execution phase by compiling mapping specifications into executable code (e.g., XSLT, XQuery) and providing the possibility to deploy this code as an executable service.

Our last observation is that the majority of mapping solutions support the mapping maintenance phase, but in a varying degree: in Microsoft BizTalk, for instance, users can publish their mappings in a web-accessible WebDav repository which does not provide any further sophisticated functions. Altova's SchemaAgent, COMA++, and Yahoo Pipes mark the opposite end of the spectrum of possibilities.

5 Analysis

After we have set up the evaluation framework in Section 3 and presented a representative selection of mapping solutions in Section 4, we now analyse these solutions according to the imposed requirements.

In the previous section, where we briefly introduced the mapping solutions, we already outlined which mapping phases a solution supports. Here we analyse in detail how and to what extent a solution supports the requirements of a certain mapping phase.

5.1 General Requirements

In the evaluation framework, this category contains all requirements that must be supported by any mapping solution but cannot be assigned to any of the four mapping phases.

Uniform Accessibility Uniform accessibility denotes the possibility of deploying a single access point to a set of heterogeneous sources from previously defined metadata mappings. A single access point could be a query interface or a service that provides transparent access the metadata provided by other services.

By nature, Enterprise Information Integration (EII) Suites fulfil exactly that requirement. BEA Liquid Data, for example, provides the means to create data views over a set of data sources that can be queried in a uniform manner using XQuery. Also the Sybase Data Integration Suite allows the deployment of a unified data layer.

Different from EII Suites, the mapping solutions categorised as Enterprise Application Integration Suites (Microsoft BizTalk Server, Cape Clear, and IBM Websphere Integration Developer), do not provide uniform access via a single query interface but rather allow transparent access to data from incompatible data sources through deployed business processes. In Service Oriented Architectures, a business process typically exposes the aggregated and converted data via a Web Service interface.

Among the "pure" metadata mapping tools we cannot find any solution that supports uniform accessibility. Although it is possible to deploy mappings with Altova MapForce in terms of services that provide integrated access to a single source, this is not possible for multiple sources. COMA++ does not support mapping execution at all and therefore cannot provide uniform access. Clio has the the facilities to convert mappings into queries or transformation style-sheets without considering query execution.

From the group of non-categorised solutions, StylusStudio enables the deployment of so called XML Pipelines[7]. An XML Pipeline allows to aggregate metadata from a set of XML sources and define various processing steps to transform them into a single format exposed by a single endpoint. StylusStudio therefore supports uniform accessibility. Yahoo Pipes acts in a similar manner, with the difference that all pipelines may be created and shared on the Web. TopBraid Composer does provide means to deploy a uniform access interface for various sources.

Modularity Assuming that a mapping solution already provides access to a set of data sources, often the need arises to add additional or remove already integrated data sources without changing any existing implementation. Depending on the modularity of a mapping solution's architecture, this requirement can optionally be supported. Obviously this is only relevant for solutions that do fulfil the previous requirement and provide uniform access to multiple sources.

When using EII Suites such as BEA Liquid Data or the Sybase Data Integration Suite, removing or adding metadata sources requires the redefinition and redeployment of previously created views. We regard this task as a minor modification of an integration definition rather than a change in the implementation and therefore categorise EII Suites as mapping solutions that support modularity.

[7] `ttp://www.w3.org/TR/xml-pipeline/`

With EAI Suites the required effort is similar: BizTalk Server, Cape Clear Server, and the IBM WebSphere Server require the redefinition and redeployment of existing business processes and service orchestrations whenever new sources are added. Therefore they partially fulfil the requirement of being modular.

Mapping solutions that follow a pipe-line approach for providing uniform access to metadata sources (StylusStudio, Yahoo Pipes) require the redefinition of the processing steps within the pipes, a task requiring only minor modification effort.

Flexibility in Lifting and Normalisation Lifting and Normalisation denotes the ability to lift metadata expressed in distinct schema definition languages to a common metadata meta-model. As already mentioned in Section 2.3, it is commonly agreed that this is a pre-condition for metadata mapping. Therefore mapping solutions should provide the flexibility to lift metadata of any kind to that common meta-model.

All analysed Enterprise Information Integration as well as all Enterprise Application Integration Suites fulfil this requirement. BEA Liquid Data can lift any data available in Relational Databases (RDB), XML files, delimited files, Web Services, and third party applications (e.g. Siebel, SAP) to the level of XML Schema. The Sybase Data Integration Suite offers so called "Data Services" to transform data available in specific source formats to its internal, proprietary representation. Microsoft BizTalk and IBM WebSphere both provide extensible and customisable adapter frameworks[8] for lifting external data to their internal representation. Also CapeClear provides a fixed set of data transformers for common formats such as SOAP, CSV, structured text (e.g. EDI, SWIFT), or Excel spread-sheets.

Since Altova MapForce supports mapping between any combination of SQL-DDL, XML Schema, Flat Files, and EDI messages, internally it also must have the facilities to lift these meta-models to a common representation. However, it is not possible to extend MapForce with adapters for custom data formats. Although not providing the flexibility to lift any proprietary format, COMA++ and Clio can both lift SQL-DDL and XML Schema definitions to their internal representation. COMA++ also supports lifting of OWL ontologies and XML Data Records (XDR).

The DataDirect XML Converters give StylusStudio the flexibility to lift practically any format to the level of XML Schema, which is its internal meta-model. DataDirect already contains a large number of converters for widely-used formats and give the users the means to easily build their own converters. TopBraid supports the lifting of a fixed set of other meta-models (e.g., UML) to the level of OWL. The same is the case for Yahoo Pipes which provides data sources adapters for XML, RDF, JSON, iCal, and CSV files.

[8] Microsoft BizTalk Adapter Framework and IBM WebSphere Integration Framework

Mapping GUI Metadata mapping is a complex task, both for domain experts and technicians. Therefore any mapping solution must support these user groups by providing a Graphical User Interface (GUI).

Since all analysed solutions fulfil this requirement, its importance becomes obvious. The EII Suites provide graphical means for building data views (BEA Data View Builder, Sybase Avaki Studio), and the EAI Suites provide orchestration design tools (BizTalk Orchestration Designer, Cape Clear Studio, WebSphere Integration Developer). Also all other solutions offer graphical means for creating mapping specifications. Compared to all other solutions, Yahoo Pipes provides an outstanding and easy to use Web-based mapping GUI.

5.2 Mapping Discovery Requirements

This category of the evaluation framework lists the common requirements that occur during the first mapping phase, which is mapping discovery. From the previous section we already know that this phase is supported only by a few tools.

Schema Matching / Alignment Support The requirements of Schema Matching or Alignment denotes the ability to (semi-)automatically support users in determining metadata mappings.

Among the commercial solutions, IBM WebSphere Integration Developer supports the user in creating mappings by automatically aligning model elements with the same lexical representation. This could be the case, if, for instance, two attributes in two distinct schemes have the same label (e.g., "Person"). Altova MapForce extends this feature and automatically aligns lexical equivalent child elements (e.g., "firstName") of already mapped elements. Since this kind of mapping support is trivial, these solutions support the schema matching requirement only partly.

Microsoft BizTalk, or more specifically the BizTalk-Mapper, offers advanced schema matching capabilities and therefore supports this requirement. Besides having the capability of matching schema elements based on their lexical names, it can also "autolink" elements based on their structure (e.g., their sub-elements).

COMA++ and Clio are research prototypes whose main feature is in fact schema matching. While the Clio matching algorithm operates only on the schema level, COMA++ uses a composite approach to combine different schema and instance level matching algorithms.

Consensus Building Features A mapping solution offering consensus building features supports users or user communities in building consensus on conflicting mappings.

From all solutions under investigation, only Yahoo Pipes partly supports this requirement. It has built-in user and community management features, which are an important prerequisite for building consensus. Further it is built upon Web technology, which lowers the entry barriers for building communities. For existing

mappings Yahoo Pipes offers search and browsing as well as ranking features. Mappings can be cloned and reused for other integration tasks; one can assume that cloning demands at lest a minimal degree of agreement and consensus on a certain mapping.

5.3 Mapping Representation Requirements

By their nature, all mapping solutions require a formalism for representing mappings. As we have seen in Figure 6 in the previous section, this is indeed the case for all solutions under consideration. At this point we will analyse the strengths of each formalism, i.e., its ability to capture the various kinds of heterogeneities and interoperability conflicts.

Model-Level Structural Heterogeneity Reconciliation Structural conflicts on the model level fall into two categories: *element definition conflicts* occur because the elements of distinct models might have assigned different names, identifiers, or conflicting constraints. *Domain representation conflicts* arise because domain experts reflect the constituents of a domain in different generalisation hierarchies, using a different number and different types of elements. While element definition conflicts are easily resolvable by renaming elements, dealing with domain representation conflicts is a more complex task. They can be reconciled by providing the ability to relate, for instance, a general entity in one model with more concrete entities in another model.

All EII Suites under investigation can resolve element definition conflicts and relate elements with different names, identifiers, or data type constraints. BEA Liquid Data can also resolve a majority of domain representation conflicts. Although not directly reflected in the Data View Builder GUI, one can create XQuery expressions to relate a source element to multiple target elements. Further it is possible to deal with different generalisation hierarchies and relate concrete with more general model elements by defining conditions that filter out relevant data values. By providing a rich set of operators, Sybase Avaki Studio can also resolve these heterogeneity conflicts.

Among the EAI Suites, Microsoft BizTalk and Cape Clear Studio both rely on the power of XSLT to transform objects from a source schema to a target schema. The creation of XSL style-sheets is supported by mapping GUIs, which in both cases do not reflect the full power of XSLT. In Biz Talk Mapper for instance, it is not possible create 1:n mapping relationships between model elements. However, when manipulating the XSL style-sheets directly, all element definition and domain representation conflicts can be resolved. The IBM WebSphere Integration Developer does not rely on standardised technologies such as XQuery or XSLT, but allows the definition of so called "business maps" among business objects. Using a predefined set of "Transform Type" objects, domain experts can relate different model structures and resolve structural heterogeneities.

Altova provides a powerful mapping model for reconciling any structural heterogeneity and can transform such representations into XSLT or XQuery.

Clio does not provide these capabilities on the GUI, but also relies on XQuery and can therefore represent the required mapping information. Since COMA++ is a schema matching tool with the primary intent to discover mappings, it can represent a mapping relationship between two attributes but cannot relate an entity with one or more attributes.

Stylusstudio also relies on XQuery and can therefore deal with any structural heterogeneity. TopBraid Composer uses SPARQL, or more specifically, the SPARQL CONSTRUCT statement, which like XQuery gives the freedom to design and construct any target model from a set of source models. Finally, also Yahoo Pipes can indirectly represent structural mappings through its operators.

Model-Level Semantic Heterogeneity Reconciliation The two main classes of semantic conflicts on the model-level are *domain conflicts* (e.g., semantically overlapping, subsuming, or incompatible model elements) and *terminological mismatches* (e.g., synonyms and homonyms). A mapping formalism should provide means to define the type of heterogeneity between two model elements (e.g., "element X and element Y semantically overlap", "element X and element Y are synonyms") and the ability to reconcile that conflict, if possible.

From the category of EII Suites, all solutions provide means to resolve domain conflicts and terminological mismatches. Of course, only if they are resolvable — mapping two models from incompatible domains (e.g., billing and gardening), for instance, is not feasible for any tool. However, it is possible, for instance, to map semantically subsuming elements (e.g., "author" and "person") by filtering and transforming instances according to specific conditions ("authors = persons that have written books"). Sybase Avaki provides a library of operators and expressions, BEA Liquid Data the expressiveness of XQuery to perform that task. However, none of them provides the means to explicitly define the type of heterogeneity (e.g., "subsume", "overlap", "synonym", etc), which is remarkable because many scientific approaches in literature concentrate on this kind of semantic representation (see Section 3.3).

EAI Suites support semantic heterogeneity reconciliation on the model level in a similar way as EII Suites and neither offer means to represent the semantics of a mapping relation. Microsoft BizTalk provides an extensible set of functions that fulfil the same task as operators in the above mentioned EII Suites. WebSphere relies on the definition of *maps* that are made of a series of transformation steps, which define how to transform source into target business objects. Since each map is in fact Java code, one has the full expressiveness of a programming language for semantically reconciling model elements.

For Altova our analysis bears the same results as for EII and EAI Suites: semantic reconciliation is possible through a set of operators but there is no mechanism to represent the nature of a mapping relation. Clio does not provide such operators or functions (sort, join, rename) in its mapping model and therefore does not have any advanced capabilities for resolving semantic heterogeneities, if we disregard the fact that one could manually edit the XQuery interpretation of the mapping generated by Clio. The same is the case for COMA++ — it has

no reconciliation operators and does not offer means to represent the semantics of a mapping relationship.

Stylusstudio utilises the full power of XQuery and its functions for heterogeneity reconciliation; TopBraid represents mappings in SPARQL which also uses the set of XQuery function primitives for model reconciliation. Yahoo Pipes is not extensible with respect to its operators but provides a fixed set, which is sufficient to deal with semantically conflicting model elements.

Instance-Level Semantic Heterogeneity Reconciliation Instance transformation is the means for reconciling semantic heterogeneities on the instance level. It specifies how an instance value can be transformed from one representation into another. A transformation is usually implemented in terms of functions, which can define simple operations such as data type conversion (e.g., string to integer) but also more complex tasks such as converting scales (e.g., weight:pound to weight:kg). For the complex tasks, a mapping formalism should support the implementation of custom, domain-specific functions.

BEA Liquid Data provides a set of standard functions for creating data views and is also extensible by creating custom functions to perform specialised tasks. The same is the case for the Sybase Data Integration Suite.

Also the EAI Suites provide such means: Microsoft BizTalk and Cape Clear rely on XSLT which in fact is a fully functional language for implementing a broad range of transformation scenarios. IBM WebSphere does not rely on standard technologies but allows to implement custom transform types for converting data values within business objects from one representation to another.

Altova offers a variety of standard functions for transforming data and also allows the implementation and registration of custom functions. Clio implicitly supports this feature by relying on XQuery but does not provide the necessary means on the GUI level. COMA++ does not have any data transformation capabilities.

Also Stylusstudio and TopBraid Composer support instance level reconciliation of semantic heterogeneities. The former allows the definition of user-defined XQuery functions. The latter relies on SPARQL which also uses the XQuery function set. Yahoo Pipes currently provides only a fixed set of functions that cover a wide, but not extensible spectrum of possible transformations.

Context Representation Since the semantic correctness of mappings depends on the *mapping context*, i.e., the setting in which a mapping has been created, a mapping formalism should be able to capture such information.

A simple form of context representation is to capture user information, i.e., the user-name of the domain expert that has created a mapping. From all mapping solutions under consideration, only Yahoo Pipes binds established mappings to a certain user.

Flexible Language Binding If a mapping solution relies on a generic formalism, it is open for mappings between schemes expressed in a variety of schema

definition languages. The drawback of such an approach is the increased complexity and additional implementation effort for each single language. As already illustrated in Figure 6, some solutions have mapping capabilities for specific languages (e.g., XML Schema, OWL) while others rely on proprietary meta-models.

Solutions bound to a specific language do not have the flexibility to map metadata expressed in other languages without lifting them to a common representation. BEA Liquid Data, all EAI Suites, Stylusstudio, and also Yahoo Pipes are bound to XML Schema or XML respectively and therefore cannot be considered as having a flexible language binding. This also the case for TopBraid composer, which is bound to OWL.

Sybase Avaki relies on a proprietary model and has specific bindings for other types of models. Also Altova MapForce, Clio and COMA++ rely on a generic approach and have bindings for languages such as XML, OWL, or SQL-DDL.

5.4 Mapping Execution Requirements

In this section, we analyse to what extent the analysed mapping solutions fulfil common requirements that occur when mappings are executed during run-time.

Query Reformulation Mapping solutions that are part of *virtually integrated* systems, hence systems that leave the data in their data sources without replicating them to a central store, require a mechanism that reformulates queries according to a mapping definition. A common way to implement such a mechanism is to work with views, i.e., the user selected schema is described as a set of views over the data sources.

Regarding our representative selection of mapping solutions, we can further divide them into two categories: those that use mappings to generate views, and those that use them to transform metadata from one representation to another. Only the Enterprise Information Integration (EII) Suites fall into the first category: in BEA Liquid Data, the domain experts create views over a set of data sources using XQuery. Sybase Avaki follows a hybrid approach: it supports the definition of so called "view models", which are sequences of operations that combine or transform data from one or more sources. Other solutions, such as Altova MapForce or Stylussstudio, allow to generate XQuery code from mapping definitions, but do not provide means to execute these queries. Therefore we consider them to not support query reformulation in the mapping execution phase.

Query Plan / Optimiser Obviously, only metadata solutions that provide capabilities for query reformulation, can offer means for optimising query access to data sources.

BEA Liquid Data offers a variety of features for query optimisation. First it allows to view query plans and to analyse the execution times for each part of a query. Users can either rely on a built-in optimiser or perform manual optimisation by, for instance, changing the order of the data sources to be queried or by

giving optimisation hints that override the default behaviour of that optimiser. Sybase Avaki does not provide any query optimisation features but relies on a built-in caching service, which stores (temporary) query results for a definable time span.

Integration Component Generation A mapping solution should provide at least some semi-automatic means to compile mapping specifications into executable integration components, such as data source adapters, stubs, wrappers, or mediators.

With BEA Liquid Data on can deploy mapping specifications on the Liquid Data Server and Sybase Avaki allows to deploy mapping models as data services. On Microsoft BizTalk Server and all other EAI Suites under investigation, mapping specifications are deployed as part of executable business processes. Altova MapForce provides the possibility to generate executable Java or C# source code from a mapping specification and also in Yahoo Pipes users can deploy and execute their pipes.

5.5 Mapping Maintenance Requirements

The mapping maintenance phase is is usually supported by a kind of *mapping registry*, which stores information about available schemes and mappings between schemes.

Mapping Verification If schemes and mappings between schemes used in a certain integration context are available, an automated mechanism could detect conflicting mapping specifications.

None of the mapping solutions under investigation provides such a fully automatic mechanism. Only Clio and Altova SchemaAgent partly support this requirement: Clio verifies mappings by presenting alternative mappings to the user during the mapping discovery phase. SchemaAgent provides a GUI that allows users to browse available schemes and already established mappings between those schemes.

Mapping Reusability Reusing existing schema definitions is a very simple way of achieving interoperability; the same is the case with schema mappings. If there is already a mapping specification which reconciles the heterogeneities among two schemes, and the mapping could also fit for other integration scenarios, one should reuse and possibly modify that mapping.

As soon as mappings are available in a mapping repository, they can be discovered and reused in other scenarios. Additional repository features, for instance searching and browsing mappings, could even improve the reuse potential. However, only Altova SchemaAgent and Yahoo Pipes provide such advanced functionality. In SchemaAgent users can browse existing schemes and mappings and Yahoo Pipes provides a faceted search interface which guides users through

the bulk of already created pipes. All other solutions offer rather unsophisticated repository features such as storing mapping specifications in a central WebDAV repository.

Mapping Inference Deriving new mapping relationships from existing ones currently seems to be an invariably scientific topic because it has not yet been implemented in any of the commercial mappings solutions or research prototypes we have analysed in this section.

5.6 Analysis Results

Regarding the results of our analysis, summarised in Figure 7, from a high level perspective, we can make the following observations:

Most mapping solutions fulfil the general requirements we have set up for our analysis; heavyweight EII and EAI Suites fulfil them better than standalone mapping tools. This is because the latter are designed solely for the task of mapping and produce mappings that can be deployed in other systems. Heavyweight suites cover the whole spectrum from creating mappings to providing uniform access to data sources.

The mapping discovery phase is weakly supported, which leads us to the conclusion that the field of automatic schema matching, which has extensively been studied in literature, is still not mature enough for being deployed in practice. Only the category of research prototypes includes sophisticated schema matching support; if a commercial solution supports this phase, then only in a very unsophisticated way, such as comparing the lexical representation of model elements.

Mapping representation is well supported: almost all solutions provide the means to reconcile heterogeneities among schemes and their instances. However, none of them puts an emphasis on the semantic nature of a mapping relationship. Furthermore, none of them has strong means to represent a mapping context and most mapping solutions rely on a single, specific schema language (e.g., XML Schema). This implies that these tools must support lifting and normalisation in order to provide support for metadata expressed in other schema definition languages.

The mapping execution phase is very weakly supported. From the analysed EII Suites, only BEA Liquid Data executes queries, reformulates them according to previously defined mappings and optimises the queries using a tailorable query optimisation algorithm. EAI Suites do not provide such means but rely on the deployment of business processes, which also transform data from one format into another using pre-defined mappings. In a similar way, in Yahoo Pipes, mappings are an integral part of pipes definitions, which can be deployed and executed on the Web.

The potential of mapping maintenance has not yet been considered by most mapping solutions. Only Altova SchemaAgent can be regarded as a suitable and useful schema and mapping repository, which enables (manual) mapping

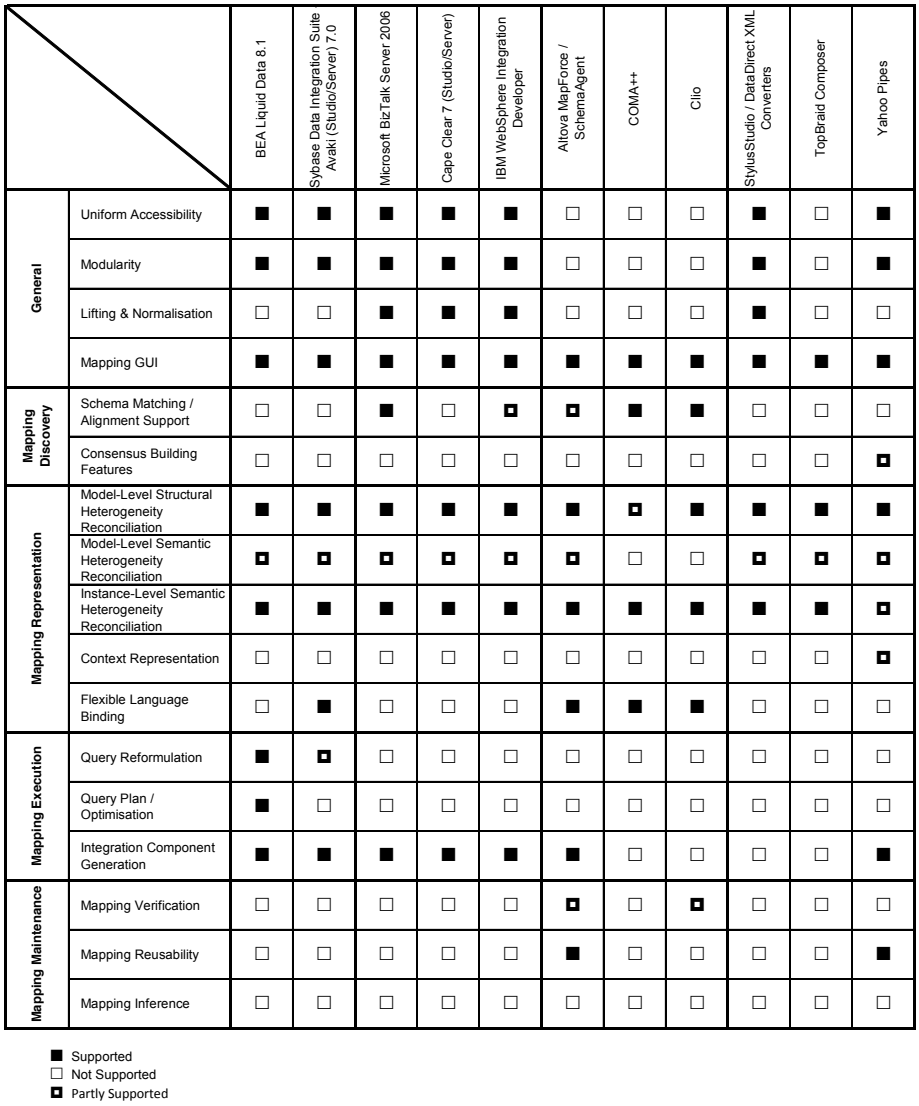

Fig. 7. Metadata mapping solutions evaluation summary.

verification and reuse. Due to its Web-based nature Yahoo Pipes, implicitly enables mapping maintenance. Users can search existing pipes, i.e., also existing mappings, for specific sources, and tailor them to their specific needs. This leads to high reusability of existing mappings and thereby to higher interoperability.

6 Summary and Conclusions

In this paper we have analysed the characteristics of a representative set of metadata mapping solutions against an evaluation framework we have derived from the state-of-the-art literature. Different from other surveys, we have conceived metadata mapping as being a cyclic sequence of phases rather than a single task (e.g., mapping discovery). We believe that this viewpoint is essential for domain experts that want to employ these solution also in real-world scenarios.

One outcome of this study is that many solutions concentrate only on a specific mapping task: research prototypes concentrate mainly on mapping discovery and disregard how these mappings could be executed in a real-world system. Commercial solutions, in contrast, have a completely different focus. They support the mapping representation and execution phases, and disregard the discovery phase. Both research and commercial solutions have in common, that the support for mapping maintenance is rather weak.

So far, the majority of mapping solutions targets domain experts working in closed environments on the basis of metadata available in structured data sources. Yahoo Pipes has revolutionised this view and provides a tool that integrates with the Web architecture. It allows users — also non-experts — to create and share mapping specifications (pipes) that integrate metadata from metadata sources exposed on the Web. We believe that this shift of mapping solutions toward the Web will continue and that it requires a lightweight metadata integration solution that can operate with common Web protocols as well as Web-enabled metadata schemes and schema definition languages. One challenge will be to provide intuitive user interfaces that allow users to create mappings without confronting them with the whole complexity of metadata schema definitions and mapping features. Also in this regard, Yahoo Pipes has set an important milestone.

References

1. OMG: Unified Modelling Language (UML). Object Management Group (OMG). (2007) Available at: `http://www.uml.org/`.
2. W3C: XML Schema 1.1 Part 1: Structure. W3C XML Core Working Group. (2006) Available at: `http://www.w3.org/TR/xmlschema11-1/`.
3. DC: Dublin Core Metadata Element Set, Version 1.1. Dublin Core Metadata Initiative. (2006) Available at: `http://dublincore.org/documents/dces/`.
4. VRA: VRA Core 4.0. Visual Resources Association's (VRA) Data Standards Commitee. (2007) Available at: `http://www.vraweb.org/projects/vracore4/index.html`.
5. IEEE WG-12: IEEE Standard for Learning Object Metadata: 1484.12.1-2002. IEEE Inc. (2002) Available at: `http://ltsc.ieee.org/wg12`.
6. Domenig, R., Dittrich, K.R.: An overview and classification of mediated query systems. SIGMOD Rec. **28**(3) (1999) 63–72
7. Wiederhold, G.: Mediators in the architecture of future information systems. Computer **25**(3) (1992) 38–49

8. Halevy, A.Y., Ives, Z.G., Mork, P., Tatarinov, I.: Piazza: data management infrastructure for semantic web applications. In: WWW '03: Proceedings of the 12th international conference on World Wide Web, New York, NY, USA, ACM Press (2003) 556–567
9. Aberer, K.: P-grid: A self-organizing access structure for p2p information systems. In: CooplS '01: Proceedings of the 9th International Conference on Cooperative Information Systems, London, UK, Springer-Verlag (2001) 179–194
10. Nejdl, W., Wolf, B., Qu, C., Decker, S., Sintek, M., Naeve, A., Nilsson, M., Palmér, M., Risch, T.: Edutella: a P2P networking infrastructure based on rdf. In: WWW '02: Proceedings of the 11th international conference on World Wide Web, New York, NY, USA, ACM Press (2002) 604–615
11. Rodríguez-Gianolli, P., Kementsietsidis, A., Garzetti, M., Kiringa, I., Jiang, L., Masud, M., Miller, R.J., Mylopoulos, J.: Data sharing in the hyperion peer database system. In: VLDB '05: Proceedings of the 31st international conference on Very large data bases, VLDB Endowment (2005) 1291–1294
12. Aberer, K., Cudre-Mauroux, P., Hauswirth, M., van Pelt, T.: GridVine: Building internet-scale semantic overlay networks. In: International Semantic Web Conference (ISWC). Volume 3298 of LNCS. (2004) 107–121
13. Maedche, A., Motik, B., Silva, N., Volz, R.: Mafra — an ontology mapping framework in the semantic web. In: Proceedings of the ECAI Workshop on Knowledge Transformation, Lyon, France, 2002. (2002)
14. Robertson, G.G., Czerwinski, M.P., Churchill, J.E.: Visualization of mappings between schemas. In: CHI '05: Proceedings of the SIGCHI conference on Human factors in computing systems, New York, NY, USA, ACM Press (2005) 431–439
15. Miller, R.J., Hernández, M.A., Haas, L.M., Yan, L., Ho, C.T.H., Fagin, R., Popa, L.: The Clio project: managing heterogeneity. SIGMOD Rec. **30**(1) (2001) 78–83
16. Haas, L.M., Hernández, M.A., Ho, H., Popa, L., Roth, M.: Clio grows up: from research prototype to industrial tool. In: SIGMOD '05: Proceedings of the 2005 ACM SIGMOD international conference on Management of data, New York, NY, USA, ACM Press (2005) 805–810
17. Yahoo! Inc.: Yahoo Pipes (2007) Available at: `http://pipes.yahoo.com`.
18. Bernstein, P.A., Melnik, S., Petropoulos, M., Quix, C.: Industrial-strength schema matching. SIGMOD Record **33**(4) (2004) 38–43
19. Kalfoglou, Y., Schorlemmer, M.: Ontology mapping: the state of the art. Knowl. Eng. Rev. **18**(1) (2003) 1–31
20. Rahm, E., Bernstein, P.A.: A survey of approaches to automatic schema matching. The VLDB Journal **10**(4) (2001) 334–350
21. Shvaiko, P., Euzenat, J.: A survey of schema-based matching approaches. Journal of Data Semantics **3730** (2005) 146–171
22. Doan, A., Halevy, A.Y.: Semantic-integration research in the database community. AI Magazine **26**(1) (2005) 83–94
23. Noy, N.F., Musen, M.A.: The prompt suite: interactive tools for ontology merging and mapping. Int. J. Hum.-Comput. Stud. **59**(6) (2003) 983–1024
24. Doan, A., Madhavan, J., Domingos, P., Halevy, A.: Learning to map between ontologies on the semantic web. In: WWW '02: Proceedings of the 11th international conference on World Wide Web, New York, NY, USA, ACM Press (2002) 662–673
25. Madhavan, J., Bernstein, P.A., Rahm, E.: Generic schema matching with cupid. In Apers, P.M.G., Atzeni, P., Ceri, S., Paraboschi, S., Ramamohanarao, K., Snodgrass, R.T., eds.: VLDB, Morgan Kaufmann (2001) 49–58

26. Li, W.S., Clifton, C.: Semint: a tool for identifying attribute correspondences in heterogeneous databases using neural networks. Data Knowl. Eng. **33**(1) (2000) 49–84
27. Aumueller, D., Do, H.H., Massmann, S., Rahm, E.: Schema and ontology matching with COMA++. In Özcan, F., ed.: SIGMOD Conference, ACM (2005) 906–908
28. Zhdanova, A.V., Shvaiko, P.: Community-driven ontology matching. In Sure, Y., Domingue, J., eds.: ESWC. Volume 4011 of Lecture Notes in Computer Science., Berlin, Heidelberg, Springer (2006) 34–49
29. von Ahn, L., Dabbish, L.: Labeling images with a computer game. In: CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems, New York, NY, USA, ACM Press (2004) 319–326
30. Mena, E., Illarramendi, A., Kashyap, V., Sheth, A.P.: Observer: An approach for query processing in global information systems based on interoperation across pre-existing ontologies. Distrib. Parallel Databases **8**(2) (2000) 223–271
31. Xiao, H., Cruz, I.F.: Ontology-based query rewriting in peer-to-peer networks. In: In Proceedings of the 2nd International Conference on Knowledge Engineering and Decision Support, 2006. (2006)
32. Goh, C.H., Bressan, S., Madnick, S., Siegel, M.: Context interchange: new features and formalisms for the intelligent integration of information. ACM Trans. Inf. Syst. **17**(3) (1999) 270–293
33. Wache, H.: Semantische Mediation für heterogene Informationsquellen. PhD thesis, University of Bremen (2003)
34. OMG: Ontology Definition Metamodel Specification (ODM). Object Management Group (OMG). (2006) Available at: `http://www.omg.org/docs/ptc/06-10-11.pdf`.
35. OMG: Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification. Object Management Group (OMG). (2005) Available at: `http://www.omg.org/cgi-bin/apps/doc?ptc/05-11-01.pdf`.
36. W3C: DAML+OIL. (2001) Available at: `http://www.w3.org/TR/daml+oil-reference`.
37. The Rule Markup Iniative: RuleML – version 0.91 (2006) Available at: `http://www.ruleml.org/`.
38. Cai, Y., Dong, X.L., Halevy, A., Liu, J.M., Madhavan, J.: Personal information management with semex. In: SIGMOD '05: Proceedings of the 2005 ACM SIGMOD international conference on Management of data, New York, NY, USA, ACM Press (2005) 921–923
39. Noy, N.F.: Semantic integration: a survey of ontology-based approaches. SIGMOD Rec. **33**(4) (2004) 65–70
40. Halevy, A.Y.: Answering queries using views: A survey. The VLDB Journal **10**(4) (2001) 270–294
41. Rajaraman, A., Sagiv, Y., Ullman, J.D.: Answering queries using templates with binding patterns (extended abstract). In: PODS '95: Proceedings of the fourteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems, New York, NY, USA, ACM Press (1995) 105–112
42. Millstein, T., Levy, A., Friedman, M.: Query containment for data integration systems. In: PODS '00: Proceedings of the nineteenth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, New York, NY, USA, ACM Press (2000) 67–75
43. Papakonstantinou, Y., Gupta, A., Haas, L.: Capabilities-based query rewriting in mediator systems. In: Proceedings of 4th International Conference on Parallel and Distributed Information Systems, Miami Beach, Flor. (1996)

44. Jarke, M., Koch, J.: Query optimization in database systems. ACM Comput. Surv. **16**(2) (1984) 111–152
45. Tatarinov, I., Halevy, A.: Efficient query reformulation in peer data management systems. In: SIGMOD '04: Proceedings of the 2004 ACM SIGMOD international conference on Management of data, New York, NY, USA, ACM Press (2004) 539–550
46. Chawathe, S., Garcia-Molina, H., Hammer, J., Ireland, K., Papakonstantinou, Y., Ullman, J.D., Widom, J.: The TSIMMIS project: Integration of heterogeneous information sources. In: 16th Meeting of the Information Processing Society of Japan, Tokyo, Japan (1994) 7–18
47. Ullman, J.D., Widom, J., Garcia-Molina, H.: Database Systems - The Complete Book. Prentice Hall, Inc. (2002)
48. Halevy, A.Y., Ashish, N., Bitton, D., Carey, M., Draper, D., Pollock, J., Rosenthal, A., Sikka, V.: Enterprise information integration: successes, challenges and controversies. In: SIGMOD '05: Proceedings of the 2005 ACM SIGMOD international conference on Management of data, New York, NY, USA, ACM Press (2005) 778–787
49. BEA: BEA Liquid Data for WebLogic 8.1. BEA Systems, Inc. (2007) Available at: `http://edocs.bea.com/liquiddata/docs81/index.html`.
50. Sybase: Sybase Data Integration Suite. Sybase Inc. (2007) Available at: `http://www.sybase.com/products/dataintegration/dataintegrationsuite`.
51. Microsoft: Microsoft BizTalk Mapper. Microsoft Inc. (2007) Available at: `http://www.microsoft.com/biztalk/techinfo/tips/mapper/default.mspx`.
52. Cape Clear: CapeClear Studio / Server. Cape Clear Software Inc. (2007) Available at: `http://www.capeclear.com/products/index.shtml`.
53. IBM: IBM WebSphere Integration Developer. IBM Inc. (2007)
54. Altova: Altova MapForce. Altova, Inc. (2007) Available at: `http://www.altova.com/products/mapforce/data_mapping.html`.
55. Altova: Altova SchemaAgent. Altova Inc. (2007) Available at: `http://www.altova.com/schemaagent_mapforce.html`.
56. Do, H.H., Rahm, E.: COMA - a system for flexible combination of schema matching approaches. In: VLDB, Morgan Kaufmann (2002) 610–621
57. DataDirect Technologies: Stylus Studio 2007 and DataDirect XML Converters (2007) Available at: `http://www.stylusstudio.com/`.
58. TopQuadrant Inc.: TopBraid Composer (2007) Available at: `http://www.topbraidcomposer.com/`.