QuickMig - Automatic Schema Matching for Data Migration Projects

Christian Drumm
SAP Research, Karlsruhe, Germany
christian.drumm@sap.com

Matthias Schmitt
SAP AG, St.Leon-Rot, Germany
ma.schmitt@sap.com

Hong-Hai Do
SAP Research, Dresden, Germany
hong-hai.do@sap.com

Erhard Rahm
Universität Leipzig, Leipzig, Germany
rahm@informatik.uni-leipzig.de

ABSTRACT

A common task in many database applications is the migration of legacy data from multiple sources into a new one. This requires identifying semantically related elements of the source and target systems and the creation of mapping expressions to transform instances of those elements from the source format to the target format. Currently, data migration is typically done manually, a tedious and time-consuming process, which is difficult to scale to a high number of data sources. In this paper, we describe QuickMig, a new semi-automatic approach to determining semantic correspondences between schema elements for data migration applications. QuickMig advances the state of the art with a set of new techniques exploiting sample instances, domain ontologies, and reuse of existing mappings to detect not only element correspondences but also their mapping expressions. QuickMig further includes new mechanisms to effectively incorporate domain knowledge of users into the matching process. The results from a comprehensive evaluation using real-world schemas and data indicate the high quality and practicability of the overall approach.

Categories and Subject Descriptors

H.2.5 [**Database Management**]: Heterogeneous Databases

General Terms

Algorithms, Experimentation

Keywords

Schema Matching, Mapping Discovery, Schema Mapping, Data Transformation, Data Migration

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
CIKM'07, November 6–8, 2007, Lisboa, Portugal.
Copyright 2007 ACM 978-1-59593-803-9/07/0011 ...$5.00.

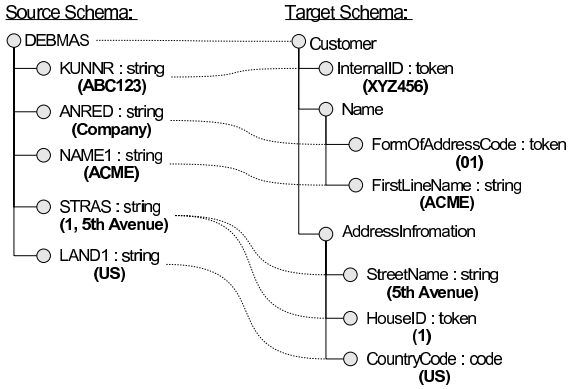

Figure 1: Running example showing a mapping of customer data

1. INTRODUCTION

Data migration is the task of transforming and integrating data originating from one or multiple legacy applications or databases into a new one. Whenever a new software application is introduced to replace existing legacy applications or whenever the application landscape is consolidated the requirement to migrate data between applications arises. During the migration process, data needs to be extracted from the source systems, transformed and loaded into the target system. This process requires solving two difficult tasks: i) *schema matching* to identify similar or semantically related elements between the source and target systems and ii) mapping discovery to determine *mapping expressions* capable of transforming instance data from the source format to the target format.

As an example, consider the schemas depicted in Fig. 1. The `DEBMAS` schema describes the data format used by the legacy system to represent a customer, whereas the `Customer` schema describes the data format used by the target system. The values in brackets show example element values for one customer instance. Furthermore, the dotted lines indicate corresponding elements in the two schemas. For example, the `ANRED` element in the source schema corresponds to the `FormOfAddressCode` element in the target schema as they contain the same semantic information namely the form of address. However, detecting

this correspondence and determining a mapping expression to transform values of the `ANRED` element to those of the `FormOfAddressCode` element remains a challenge as the former uses a textual representation whereas the latter utilizes specific codes. Another example is the complex correspondence between the source element `STRAS` and the target elements `StreetName` and `HouseID`, in which a value of the former needs to be split in order to form valid entries of the latter.

As surveyed in [7, 19], previous research work mostly focuses on the schema matching task to semi-automatically identify semantic correspondences between schema elements. To determine the similarity, i.e. degree of relatedness, between schema elements, proposed techniques exploit various kinds of information, including schema characteristics such as element names, data types and structural properties, characteristics of data instances, as well as background knowledge from dictionaries and thesauri. However, only a few approaches, e.g. [12, 21, 6], have addressed the task of semi-automatic discovery of mapping expressions. They typically use the available data instances in data sources and apply sophisticated machine learning techniques to determine mapping expressions between the matching schema elements.

In this paper, we propose a new and integrated approach for schema matching and mapping discovery to support migration and transformation of data between heterogeneous sources. Compared to previous work, our approach exhibits the following improvements:

Novel use of sample instances: Instead of exploiting unrelated instances between source and target system, we propose to define and use a uniform set of standard instances. The availability of such instances in the source and target system makes it possible to use much more efficient methods to detect matching elements.

New instance-based matchers: Exploiting the availability of the same instances in the source and the target system, we developed a set of relatively simple instance-based matchers, which however are able to detect complex correspondences and mapping expressions in real-world schemas.

Comprehensive set of mapping categories: To capture such mapping expressions for data transformation, we have developed a comprehensive set of mapping categories, indicating how instance data has to be transformed from the source to the target format. The majority of the categories can be automatically detected by our new matchers.

Enhanced mapping reuse: Based on the mapping reuse idea of [7, 9], we have developed a new reuse matcher that is able to derive new mappings with mapping expressions from existing ones. The reuse matcher can be easily combined with the other matchers for improved match results.

Schema reduction based on domain knowledge: To deal with large and complex schemas, we developed a new questionnaire technique for schema reduction. Specified in domain nomenclature, the questionnaire allows the user to specify portions of the schemas relevant for data migration according to his understanding of the domain.

Real-world evaluation: We have implemented our novel concepts in a prototype and performed a comprehensive evaluation with large schemas taken from real SAP business applications. The evaluation results indicate the high quality and practicability of our approach for real-world scenarios.

The remainder of the paper is organized as follows. Section 2 briefly describes existing research related to our approach. Next, Section 3 presents an overview of the QuickMig approach. After this, sections 4 and 5 further detail important aspects of the approach before an evaluation is presented in Section 6. The paper ends by providing a summary and an outlook on future work in Section 7.

2. RELATED WORK

The work most closely related to ours is the recent schema matching tool COMA++ [7, 9]. COMA++ supports a large spectrum of schema- and reuse-based matchers and employs composition to combine the results of individually executed matchers. The result of a match operation in COMA++ is a structural mapping consisting of correspondences with a similarity value to indicate their plausibility. To deal with large schemas, COMA++ supports a fragment-based match approach, in which the input schemas are reduced to relevant schema fragments either chosen manually or identified automatically according to fragment similarity.

QuickMig is a further development of COMA++ and improves it in several aspects. First, it supports schemas with instances and includes a number of new instance-based matchers which can be combined with the existing matchers in COMA++. Second, it provides new techniques exploiting sample instances and domain-specific knowledge to detect mapping expressions for match correspondences and complex matches, such as string splitting and concatenation. Third, the mapping reuse approach of COMA++ has been enhanced to support not only mappings with similarity values, but also mappings with mapping expressions. Fourth, a new technique based on questionnaires in natural language is employed for schema reduction. The user can choose relevant fragments for matching according to his understanding of the business domain, and not based on the technical knowledge of the schemas as expected by the manual fragment selection in COMA++.

To determine corresponding schema elements, previous work typically exploits either schema information, such as element names, data types, and schema structure [1, 7, 9, 14, 15, 16, 18], or instance data [3, 2, 4, 6, 10, 11, 12, 13, 17]. So far, only a few approaches try to combine both schema- and instance-based techniques [6, 10, 21]. Inheriting the composite combination approach from COMA++, QuickMig is able to combine schema-based, instance-based and reuse matchers in a flexible way.

Previous instance-based approaches mostly rely on the availability of real instance data and apply sophisticated machine learning, statistical or information retrieval techniques to determine instance similarity of schema elements. However, the resulting quality of such approaches essentially depends on the quality of the available instance data. Our approach to provide and use sample instances makes it possible to achieve high match quality independently from the availability and quality of real instance data.

Utilizing representative instances for schema matching was also proposed in the DUMAS prototype [4]. Unlike in QuickMig, such instances are determined using automatic duplicate identification, which may mislead the match operation with wrong duplicates. Furthermore, in contrast to DUMAS, QuickMig is able to determine mapping expressions for match correspondences and to identify different kinds of complex matches, such as string splitting and concatenation.

To date, only a few matching approaches address the task of finding mapping expressions or at least try to return correspondences with a higher semantics. DIKE [18] can detect semantic relationships, like synonymy and hypernymy, between schema elements. iMap [6] uses machine learning and [21] exploits a domain ontology to suggest complex matches, such as string concatenations and arithmetic operations. Starting from a set of correspondences provided by either the user or (semi-) automatic match, Clio [12] and HepTox [5] try to infer query mappings to query and transform instances of one schema to another one.

Unlike these prototypes, QuickMig employs several distinct techniques to identify mapping expressions and complex matches. First, sample instance data makes it possible to identify string split and concatenation using simple string comparison. Second, sample instance data combined with domain knowledge, such as standard formats and structures for modeling date, time, address, phone, fax data, enables the detection of complex matches between these formats and structures. Finally, QuickMig is also able to derive mapping expressions for match correspondences by reusing existing mappings and the associated mapping expressions.

3. OVERVIEW

This section gives an overview of the QuickMig approach for schema matching and mapping discovery. We start by discussing the observations driving the design of our approach (3.1) before presenting our migration process (3.2), the architecture (3.3) and the mapping categories (3.4).

3.1 Observations

The design of our approach to schema matching and mapping discovery is driven by several observations which we made by analyzing practical data migration scenarios of various SAP customers:

Limited value of schema information: We generally observed that legacy systems are mostly not optimized towards fostering system interoperability. The schemas typically make extensive use of technical names, abbreviations, and proprietary structure (cf. the running example). This makes it difficult to determine correspondences, not to mention mapping expressions, between schema elements. As a consequence, additional kinds of information, especially instance data, domain knowledge, and previously determined mappings, need to be considered in order to achieve reasonable quality in the schema matching and mapping discovery process.

Availability of domain knowledge: The migration process needs knowledge about both the source and the target system. Unfortunately, such knowledge is not always available at one place. The knowledge about the target system is available at its vendor, while only customers can provide detailed knowledge about their source systems. As a consequence, a migration solution developed by a vendor of a target system, e.g. SAP, should exploit the knowledge about the target system, and at the same time support effective mechanisms to incorporate the customers' knowledge about their source systems. In particular, the knowledge about the target system, e.g. field semantics, sample instances, data formats and code lists, can be specified in an ontology for automatic analysis. This manual effort is needed only once and can be quickly amortized over many migration projects.

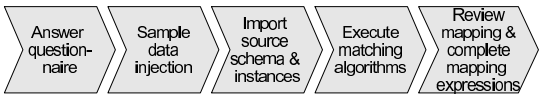

Figure 2: The QuickMig matching process

Scope of data migration projects: Typically, the data sources involved in data migration projects are complex, resulting in large schemas to be matched. Depending on the need of the particular customer, only certain parts of the target system need to be populated with source data, which results in different target schemas for different migration projects, even for the same target system. This potential for schema reduction needs to be exploited as much as possible in order to reduce the complexity of the data migration tasks; Schema reduction can be performed with suitable involvement of the customer at the beginning of the migration process.

Accumulated mapping knowledge: In a given migration project usually many different mapping tasks need to be solved, e.g. for customer data, supplier data, purchase orders, etc. As these schemas usually contain common parts, like address data, there is a high reuse potential for the results from previous mapping tasks, especially regarding complex mapping expressions which otherwise have to be developed manually. In our approach we consider this opportunity by the means of a dedicated matcher.

3.2 The QuickMig Migration Process

The problems and opportunities mentioned in the previous paragraphs led to the migration process depicted in Fig. 2.

1. Answering a Questionnaire. This first step is manually performed by a person with some knowledge of the capabilities of the source system. The purpose of the questionnaire is to collect as much information about the source system as possible. This information will be used to automatically reduce the complexity of the target schemas and thereby reduce the complexity of the matching process. A detailed discussion of the schema reduction will be given in Section 4.1.

2. Injection of Sample Instances. In the second step instances already existing in the target system are manually created in the source system by a user. These sample instances are used by the instance-based matching algorithms (see also Section 5.1) in order to determine correspondences between the source and the target schemas. A detailed description of the sample data creation in the source system is given in Section 4.2.

3. Schema and Instance Import. The third step in the migration process is the import of the source schemas as well as the corresponding sample instances into the QuickMig system.

4. Matcher Execution. In the fourth step the schema matching algorithms will be executed and automatically determine a mapping proposal using different matching algorithms. This mapping proposal includes similarities between elements of the source and target schemas as well as a proposal for the *mapping categories*.

5. Review. A developer reviews and corrects the mapping proposal in the final step. When the mapping proposal is accepted real mapping code is generated and the mapping is stored in a mapping repository for later execution or reuse.

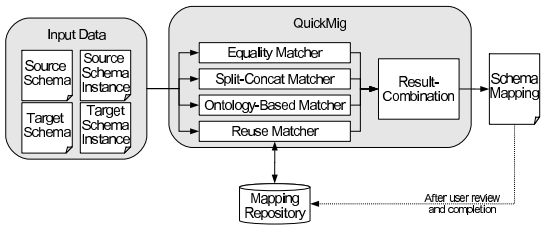

Figure 3: QuickMig architecture

3.3 The QuickMig Architecture

The QuickMig system is based on COMA++ [7, 9]. An overview of the architecture of the QuickMig system is given in Fig. 3. QuickMig extends COMA++ by implementing three new instance-based matching algorithms, namely the Equality, the Split-Concat and the Ontology-based matcher, and by improving the reuse matcher. A detailed description of these algorithms can be found in Section 5. Each matching algorithm creates a list of correspondences between the source and the target schema as well as the associated mapping category. The results of all algorithms are combined in order to create the final mapping presented to the user. Furthermore QuickMig adds functionalities to support instance data and the creation and management of mapping categories.

3.4 Mapping Categories

As described in the previous section the matching process not only returns correspondences between schema elements but also mapping categories associated to these correspondences. These mapping categories can be used to create parts of the necessary mapping expressions automatically or at least to provide a mapping expression template that can easily be completed by a developer.

As an example consider the `NAME1` element in the running example. This element corresponds to the element `FirstLineName`. The correct mapping category is *Move*. This means that the content of the source element can be copied into the target element without modification. Therefore the mapping expression for this category can be created automatically. In contrast to this consider the `KUNNR` element. This element corresponds to the `InternalID` element. The associated mapping category is *InternalID* meaning that an internal ID, which depends on the content of the source element, needs to be created in the target system. In this case only parts of the mapping expression can be generated automatically. A developer needs to complete the expression with the code to create a correct internal ID.

Based on an examination of mapping expressions common in migration projects, we identified 11 mapping categories. Table 7 in the appendix provides an overview of these mapping categories, together with a short explanation of each category. Furthermore, the table indicates in the third column if the mapping code for this category can be created automatically by the QuickMig system, and in the fourth column if the category can be identified automatically. The next paragraph briefly explains the most important mapping categories.

Move is the most simple and also the most frequent mapping category which we observe in a typical mapping. When this mapping category is assigned to a correspondence, instance data is simply copied from the source to the target schema element. Other common mapping categories are *Split* and *ValueMapping*. In the case of *Split*, the contents of a source schema element has to be split into several target schema elements. The *ValueMapping* category indicates that the source values need to be mapped to specific target values. This mapping can either be performed using a standard code list, if applicable, or using a custom translation table. Such a custom translation table maps certain source to certain target values and needs to be maintained manually. The mapping category *Complex* indicates the need for a complex function to translate the source instance data. In this case no automatic suggestion is provided by the QuickMig system. Instead, a user needs to create the necessary mapping expression manually.

For those mapping categories for which mapping expressions can be generated automatically, the QuickMig system creates the correct ones. In the other cases, a user either has to complete the mapping expression (e.g. in the case of *ValueMapping*) or has to create the whole mapping expression from scratch (in the case of *Complex*). With this approach no work at all is necessary for the simple cases, the user only needs to take care of the complex ones.

4. SCHEMA REDUCTION AND SAMPLE DATA

This section describes, in more detail, the two manual steps of the QuickMig process, i.e. the answering of a questionnaire in order to reduce the target schema and the creation of sample data in the source system.

4.1 Target Schema Reduction

Usually not all complexity supported by a target system is necessary in a given migration project. In order to deal with large schemas, we suggest to identify and leave out irrelevant parts of a schema and to simplify complex substructures by exploiting the domain knowledge of the user. In particular we propose to provide a (electronic) questionnaire for every target schema (or set of closely-related target schemas). Based on the answers to the questionnaire, the target schema(s) are reduced to the relevant parts, which in turn leads to a reduction of the complexity of the matching task. Examples of questions in such a questionnaire are: "Shall bank account data be migrated for a customer?" or "Shall multiple addresses be migrated for each customer?". If these questions are answered with no, the `BankDetails` sub-structure of the target schema can be removed and the `Address` sub-structure can be merged with the `CustomerT` sub-structure of the target schema. This approach is depicted in Fig. 4.

The mapping between the reduced target schema and the original target schema is generated automatically. As both the target schema and the questionnaire, including all possible answers, are known in advance, the necessary mapping expressions resulting from certain answers to the questionnaire are stored together with the questionnaire. The initial development of the mappings is performed during the development of the questionnaire. Consequently, no additional matching effort arises from the schema reduction in a given migration project. Furthermore, these mappings are reused in many migration projects, therefore the effort of initially creating them will be amortized over time.

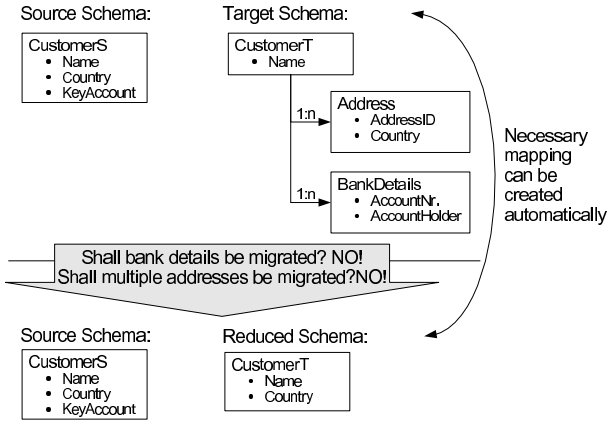

Figure 4: Reduction of the target schema

4.2 Sample Data

Various previous schema matching approaches already make use of instance data [3, 6, 10, 11, 12, 13, 17]. However, they mostly apply sophisticated statistical, machine learning approaches to unrelated instances available in source and target systems in order to identify matching elements. Unfortunately, similar instances, such as phone/fax numbers typically result in wrong matches, e.g. between customer and supplier phone and fax numbers as reported in [10].

Therefore we propose to deliver one sample instance for every target schema instead of analyzing sets of unrelated instances. This sample instance contains data for all relevant fields of the target schema. In the second step of the migration process relevant sample instances will be automatically selected based on the answers to the questionnaire and presented in a user-friendly way. A business user will then manually create the same instances in the source system. Note that creating these instance in these legacy systems can be done quickly by business users, as entering this kind of information into the system is their daily business.

Using sample data this knowledge can be exploited by the matching algorithms. By injecting sample data into the source system the matching algorithms do not only have arbitrary instances available but one dedicated instance which maps exactly to a specific instance of the target schema. This feature is later exploited by the matchers. Instead of comparing unrelated instances, the matchers use instances based on the same data to identify related schema elements.

The idea of injecting sample data into the source system also helps to verify concrete mapping expressions. After a complete mapping has been created, the sample instance of the source system can be translated using this mapping. If the result of this translation differs from the existing sample instance in the target system, it is very likely that the mapping function is wrong and needs to be revised. As an example consider the `ANRED` element of the running example. If execution of a mapping does not result in the value `01` for the target schema element `FormOfAddressCode`, the mapping expression is wrong.

However, providing sample data may not make sense for all schema elements. The reasons for this are twofold. First, the sample data needs to be easily understandable by a business user, who has to create the data in the source system. Second, the sample data should not contain similar values for different schema elements as this leads to wrong matching proposals. Both features can only be guaranteed for a subset of elements.

5. NEW MATCHERS

We implemented a number of new instance-based matchers which have been added to the matcher library of COMA++. This allows combining the matchers in a flexible way. Furthermore, we enhanced the reuse matcher of COMA++ to support mappings containing similarity-based correspondences as well as correspondences with mapping categories. The following sections explain the new instance-based and reuse matchers (5.1, 5.2), and the combination of their results (5.3).

5.1 Instance-based Matchers

In the QuickMig system three instance-based matchers were developed, namely i) an *Equality* matcher, ii) a *Split-Concat* matcher and iii) an *Ontology-based* matcher. In the following we will briefly introduce the Equality and the Split-Concat matcher and then focus on the more complex Ontology-based matcher.

5.1.1 Equality Matcher

The equality matcher is the most simple instance-based matcher. This matcher tries to identify equal instance values in the source and target schema. By this approach only matches with the mapping category *Move* can be identified. Although this matcher is rather simple, it fits nicely to the sample data injection as this approach leads to many identical instance values in the source and the target schema.

5.1.2 SplitConcat Matcher

In addition to the equality matcher the SplitConcat matcher checks the instance data for splitting or concatenation relationships. As an example consider the `STRAS` element of the running example. Given the sample instance data shown, the SplitConcat matcher would be able to identify that the element containing `1, 5th Avenue` matches to the elements containing the substrings `1` and `5th Avenue` because the first value can be split into the latter two. Furthermore the Split-Concat matcher would identify *Split* as the correct mapping category. In general the SplitConcat matcher is able to identify the *Split* and *Concatenate* mapping categories depending on the direction of the identified substring relationship.

5.1.3 Ontology-Based Matcher

The Ontology-based matchers exploit background knowledge provided in a domain ontology in addition to instance data in order to identify corresponding schema elements. The Ontology-based matcher currently exploits the following types of background knowledge:

Modeling alternatives for common data structures. For some forms of data a set of well known modeling alternatives exist. As an examples consider telephone numbers. They could either be represented as `<CountryCode> (<AreaID>) <SubscriberID>` or simply as `<Number>`. Other data with well known modeling alternatives are date and time information and address data.

Available standard code lists. For certain information a set of standard, pseudo-standard and proprietary

code lists exist. As an example of such a code list consider the SAP code list of form-of-address codes. Using this code list the correspondence between `Company` and `01` in the running example can be identified based on the given sample data.

The following paragraphs describe how these types of background knowledge are modeled in a domain ontology and how they are exploited by QuickMig.

The goal for the usage of a domain ontology was not only to support the mapping process but also to make the information understandable for a human user. Therefore, the ontology was not modeled according to either of the used schemas. Instead we first conceptually modeled the domain (information related to business partners in this case) using OWL DL [20] and annotated the target schema using the resulting ontology. The background knowledge was added to the domain ontology using annotations of the respective concepts or properties. As an example consider the `FormOfAddressCode` element in the running example. This element would be annotated with the object property `hasFormOfAddressCode` of the concept `Name`. The object property `hasFormOfAddressCode` has an additional annotation property `usedStandardCodeList` which is linked to an instance of a `StandardCodeList` representing the specific SAP code list for the form of address codes. Listing 1 in the appendix shows this example in abstract owl syntax.

The Ontology-based matcher uses the information modeled in the ontology to find additional correspondences based on the available instance data. For each target schema element the Ontology-based matcher checks the annotation properties of the ontology entity it is annotated with. If the annotation property contains information regarding a code list, the Ontology-based matcher uses this code list to translate the sample data related to the element. If a description of a modeling alternative is found, the matcher also checks for this alternative based on the sample data related to the element.

In the example presented previously, the specific SAP code list for form of address codes would be used to translate the string `Company` to its code representation `01`. Thereby the Ontology-based instance matcher is able to identify matching elements based on the set of instances consisting of `Company` and `01`.

It is important to note that the ontology, as well as the questionnaire and the sample data, only depends on the target system. Therefore it only needs to be developed once per system and can be delivered as part of QuickMig.

5.2 Reuse Matcher

The reuse of existing mappings represents a promising approach for solving new matching tasks, which are similar to already solved matching tasks, in an efficient way. COMA++ [7, 9] already includes a matcher applying mapping reuse. The proposed approach assumes the transitivity of the similarity relationships and uses a special compose operation to aggregate the similarity of the identified transitive match correspondences [9]. The Reuse Matcher in QuickMig is based on the same assumption of transitivity of match relationships. However, we have extended it in two ways. First, we provide an optimized pre-selection of the mapping path. Second, the representation of mappings and the compose operation are enhanced in order to support not

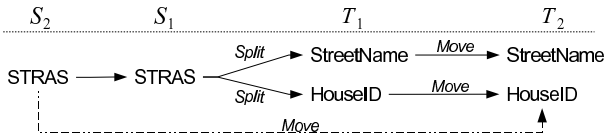

Figure 5: Combination of mapping categories

only similarity-based correspondences but also correspondences with mapping categories.

The quality of the reuse matcher essentially depends on the quality of the mapping path selected for the compose operation. In our migration scenarios we observed that migration projects typically cover different kinds of data such as customer data, supplier data, which, despite their difference in semantics, often have a very similar structure. However, this structural similarity typically exists only within the source and target systems, respectively, as they have been designed and evolved independently from each other. This intra-system similarity can be exploited in order to determine a good mapping path for our reuse matcher. In particular, to derive a mapping between a source schema S_2 and a target schema T_2, we first match the source schema S_2 to a previously matched schema S_1 to obtain a mapping m_1. Assuming that the mapping m_2 between S_1 and another target schema T_1 was constructed in a previous migration task, and the mapping m_3 between the target schemas T_1 and T_2 has been specified and maintained during the development of the target system, we can compose m_1, m_2 and m_3 to obtain a mapping between S_2 and T_2.

The implementation of the compose operation is based on Table 8, which describes how a mapping category can be composed with another one in case of transitive correspondences. In this table the column headings are the mapping categories of the mapping between T_1 and T_2 and the row headings the mapping categories between S_1 and T_1. Between S_2 and S_1 no mapping categories are calculated as only schema based matching approaches are used. Furthermore *LookUp*, *Code2Text*, *Default*, *Split*, *Concatenate* and *Complex* should not occur between the target schemas as in the target system similar data will be stored consistently (e.g. using the same codes). As an example for the combination of mapping categories consider Fig. 5. The mapping category resulting from the combination of *Split* and *Move* is the category *Split* again. Consequently *Split* is the mapping category that is assigned to the correspondence between `STRAS` in schema S_2 and `StreetName` and `HouseID` in schema T_2 respectively.

Our reuse approach is particularly promising due to three reasons. First, compared to directly matching S_2 and T_2, only one, potentially much simpler match operation is required, in particular to obtain the mapping m_1 between the source schemas S_1 and S_2. With the expected structure similarity within the source system, efficient schema-level matchers considering schema information, such as element names, data types, and structural neighborhood, may already be sufficient to derive m_1 with a high quality. Second, the inter-system heterogeneity, which actually makes it difficult to directly match S_2 and T_2, is intelligently solved by reusing the previously determined mapping m_2 between S_1 and T_1. m_2 contains only confirmed correspondences and mapping expressions, also in cases which are difficult or impossible to detect automatically. Third, the mapping m_3 between the target schemas can be specified and maintained

along with the development of the target system. Such mappings can be delivered as part of the data migration solution and are thus readily available for reuse.

5.3 Matcher Combination

In this step, the individual results of the Equality, the SplitConcat, the Ontology-based and the Reuse matcher are combined. The results of the instance-based matchers can be simply merged as the matchers focus on identifying correspondences with different mapping categories and thus complement each other. For the Reuse Matcher, we observe that it still delivers some wrong matches due to two reasons. First, the schema based matching of the source schemas might result in wrong matches. Second, the transitivity property does not hold, as already reported in regard to COMA in [9]. Using sample data we can easily identify potentially wrong matches. Element pairs with provided but unequal sample instances are most likely mismatches if they are not involved in some complex matches identified by the SplitConcat and Ontology matchers. Such element pairs are obtained by inverting the merged results of the three instance-based matchers, and are then removed from the results of the Reuse matcher. Finally, the cleansed result of the Reuse matcher is merged with the results of the instance-based matchers.

6. EVALUATION

In order to assess the effectiveness of the developed approach, an evaluation with real-world schemas and data was performed. The schema mapping process was executed automatically and results were compared against the correct mapping. The correct mappings were extracted from existing code implementing these mappings. To quantify the quality of the automatically obtained matching result, the standard measures *Precision*, *Recall* and *F-Measure*[8] were used. The precision with which the correct mapping categories are proposed was also evaluated.

Note that even though the evaluation of QuickMig was only performed using XML schemas, the system is not limited to them. As it is based on COMA++, it is, for example, also capable of supporting relational schemas.

6.1 Evaluation Set-up

To evaluate our approach the following SAP schemas were used as examples of possible source schemas[1] i) SAP R/3, Release 4.0, Customer Master Data IDoc and Vendor Master Data IDoc, ii) SAP ERP, Release 2005, Customer Master IDoc and Vendor Master IDoc and iii) SAP B1, Business Partner Data Model. The target schema in all our experiments was the Business Partner schema of new SAP solutions. Note that in the cases where two or more schemas are mentioned in the list above, the information regarding business partners is split across several schemas. Consequently all of these schemas need to be matched against the target schema. For the evaluation we will in the following treat each set of schemas as one evaluation scenario.

These schemas were chosen to evaluate QuickMig because they exhibit most of the characteristics expected in complex migration projects. The naming of the elements ranges from cryptic eight letter names in the first schema to very verbose naming in the target schema. Also the structure of the schemas ranges from flat structures in the case of the first schema to deeply nested structures in the case of the target schema. In addition, each schema groups the information differently. Finally, the selected schemas are quite large with respect to the number of schema elements. The size of the schemas used in the evaluation is given in table 1. The table shows the number of elements in each scenario as well as the number of elements that can occur multiple times.

[1]The schemas used for evaluating QuickMig can be downloaded at: http://tinyurl.com/2f99vk

Table 1: Complexity of schemas used for evaluation

Scenario	Number of elements	Number of elements w. cardinality 0..*
Target Schema	4639	4111
R/3 4.0	953	648
SAP ERP	2150	1691
SAP B1	480	293

Table 2: Complexity of the target schema after the reduction

Scenario	Number of elements	Number of elements w. cardinality 0..*
Target schema	4639	4111
reduced to R/3 4.0	645	395
reduced to SAP ERP	612	494
reduced to SAP B1	639	279

6.2 Target Schema Reduction

As described in Section 3.2 the first step in the QuickMig process is to reduce the target schema by answering a questionnaire. We answered the questionnaire according to the capabilities of the different source schemas. The results of our evaluation can be found in table 2. It shows the complexity of the reduced target schema based on source system capabilities. Generally speaking the schema reduction approach reduces the target schema to about 10-15% of its original size and many of the complex cases can be removed.

It is not possible to state exactly what elements of the target schema were removed as this largely depends on the scenario. However, a typical example is that the used target schema supports storing multiple, time dependent addresses per business partner. Some of the source systems in our experiments do not support that. Consequently these parts of the target schema could significantly be reduced.

As the original target schema contains about 4600 elements which turned out to be difficult to handle, we only executed the following experiments with the reduced schemas. Therefore we can only point out that the complexity of the target structure can be reduced significantly with the schema reduction approach, but we do not have absolute numbers on the increase of the mapping quality. Even if the automated mapping process would not be improved by the schema reduction approach, at least the benefits for the manual review of the resulting mappings are obvious.

Table 3: Evaluation results using only sample data

Scenario	Prec.	Recall	F-Meas.
R/3 4.0	1	0.27	0.43
SAP ERP	1	0.38	0.55
SAP B1	1	0.58	0.7
Average	1	0.41	0.56

Table 4: Evaluation results using domain-specific background knowledge

Scenario	Prec.	Recall	F-Meas.
R/3 4.0	1	0.5	0.66
SAP ERP	1	0.49	0.65
SAP B1	1	0.65	0.79
Average	1	0.55	0.70

6.3 Schema-level matching approaches

In order to see how our approaches perform in comparison to existing schema matching approaches we tested COMA++ [9] in the data migration use case. COMA++ mainly exploits schema information such as element names, description, data type, and schema structure. These schema-level matching approaches did not perform well on the used schemas. The explanation for this can be found by taking a close look at the used schemas. As in the introductory example, the schemas use cryptic elements names and differ largely in their structure. Identifying corresponding elements in these schemas is very difficult even for a human.

6.4 Use of Sample Data

Table 3 shows the matching results achieved by the Equality matcher exploiting sample data. Depending on the scenario, the instance-based matcher achieves f-measure values between 0.43 and 0.74. Note that the precision in all cases is 1. This means that the instance-based matcher does not produce any false positives. The relatively low recall values in some of the scenarios originate from two facts. First, sample data is only provided for a subset of elements (cf. Section 4.2). Second, some matches are not found as Equality and SplitConcat matchers only identify mappings of the categories Move, Split and Concatenate.

6.5 Domain-specific Background Knowledge

Table 4 shows the matching results achieved by QuickMig using the Equality and the SplitConcat matcher in combination with the Ontology-based matcher. The combination of these two approaches achieved F-measure values between 0.66 and 0.79, while still retaining a precision of 1. This is a significant increase over just using the instance-based matcher. The results shows that by using background knowledge modeled in an ontology the matchers were able to find more complex matches which could not be identified based on sample data only.

6.6 Reuse

In order to evaluate the performance of the Reuse matcher, the correct mapping between the source and the target schemas for vendor data (see 6.1) as well as between the target schemas were created. These mappings were available in the mapping repository for reuse. The Reuse matcher therefore had to perform one matching task between the source schemas for vendor and customer data. Table 5 shows the results achieved by the Reuse matcher. It is important to note that, in contrast to the instance-based and ontology-based matchers, this matcher sometimes only achieves a precision of 0.80. The reason for the lower precision is that both source schemas are unknown and are matched using only schema based approaches.

Table 5: Result achieved by the reuse matcher

Scenario	Prec.	Recall	F-Meas.
R/3 4.0	0.80	0.50	0.61
SAP ERP	0.81	0.43	0.56
SAP B1	1	0.80	0.89
Average	0.87	0.58	0.69

Table 6: Result achieved by matcher combination

Scenario	Prec.	Recall	F-Meas.
R/3 4.0	0.99	0.67	0.80
SAP ERP	0.99	0.70	0.82
SAP B1	1	0.80	0.89
Average	0.99	0.72	0.84

6.7 Combination of Sample Data, Background Knowledge & Reuse

The final experiment evaluates the performance of the combination of the Equality matcher, the SplitConcat matcher, the Ontology-based matcher and the Reuse matcher.

The result of this experiment is shown in table 6. The results show that the approach of using the sample instances in order to remove wrong matching proposals of the reuse matcher is valid. Furthermore, the Equality, the SplitConcat, the Ontology-based and the Reuse matcher cover different kinds of matches. Consequently, the superset covers significantly more matches than the single matcher results.

6.8 Mapping Categories

In the experiments using real SAP schemas the *Move* mapping category was assigned to about 60% of the matches. Another 15% of the matches were of the category *Complex*, and 14% of the category *ValueMapping*. All other mapping categories occurred with a much lower frequency.

The evaluation of the quality with which the mapping categories are proposed by QuickMig was performed differently than the evaluation of the matcher results. As mapping categories are only proposed for matches identified by QuickMig, missing matches were not counted as false negatives when evaluating the quality of the proposed mapping categories. Consequently only the precision of finding the correct mapping category for the proposed matches was calculated.

For the matches proposed by QuickMig the correct mapping category was proposed with a precision of 0.98 in the R/3 4.0 scenario, with a precision of 0.99 in the SAP ERP scenarios and with a precision of 0.93 in the SAP B1 scenario. Furthermore, it is important to note that all the mapping categories introduced in sec. 3.4 occurred in the evaluation scenarios.

6.9 Evaluation Summary

Taking all the results together we think that the effort to

find and implement the mappings necessary in data migration scenarios can be reduced significantly. Using the schema reduction approach, the complexity of the target structure can be reduced significantly, simplifying the verification of the matching results. Combining sample data, background-knowledge and reuse of existing mappings, about 75% of the matches are found automatically.

Of course, as we have a semi-automatic approach, we also have to consider the effort to answer the questionnaire and maintain sample data instances in the source system. Our experience is that these tasks are relatively simple, and could, in our case, be done in less than one hour per source system.

To summarize, we are convinced that our approach can provide a real benefit for migration projects.

7. CONCLUSION AND FUTURE WORK

This paper presented QuickMig, a system for the semi-automatic creation of schema mappings in data migration scenarios. The system uses a 5-step migration process. In the first step, the complexity of the mapping problem is reduced by answering a questionnaire. Next, sample instance data is manually created in the source system and imported into QuickMig. In the fourth step instance, ontology and reuse based matching algorithms are used to create an initial mapping, which is reviewed and completed in the fifth step.

The approach was experimentally evaluated using real SAP schemas. In these experiments the QuickMig approach achieved an average precision of 0.99, an average recall of 0.72 and an average F-measure of 0.84. Furthermore, QuickMig not only returns matches between schema elements but also assigns mapping categories to these matches, enabling the automatic creation of parts of the mapping. In the experiments with real SAP schemas QuickMig was able to identify the correct mapping categories with an average precision of 0.97.

In the future we plan to prototypically integrate the QuickMig approach into SAP data migration tools and apply it in further scenarios.

8. REFERENCES

[1] S. Bergamaschi, S. Castano, M. Vincini, and D. Beneventano. Semantic integration of heterogeneous information sources. *Data & Knowledge Engineering*, 36(6):215–249, 2001.

[2] J. Berlin and A. Motro. Autoplex: Automated discovery of content for virtual databases. In *Proc. 9th Intl. Conf. on Cooperative Information Systems (CoopIS)*, 2001.

[3] J. Berlin and A. Motro. Database schema matching using machine learning with feature selection. In *Proc. 14th Intl. Conf. on Advanced Information Systems Engineering (CAiSE)*, 2002.

[4] A. Bilke and F. Naumann. Schema matching using duplicates. In *Proc. 21. Intl. Conf. Data Engineering (ICDE)*, 2005.

[5] A. Bonifati, E. Chang, T. Ho, L. Lakshmanan, and R. Pottinger. HePToX - marrying xml and heterogeneity in your p2p databases (software demonstration). In *Proc. 31rst Intl. Conf. Very Large Databases (VLDB)*, 2005.

[6] R. Dhamankar, Y. Lee, A. Doan, A. Halevy, and P. Domingos. imap - discovering complex semantic matches between database schemas. In *Proc. of the ACM SIGMOD Intl. Conf. on Management of Data (SIGMOD)*, 2004.

[7] H.-H. Do. *Schema Matching and Mapping-based Data Integration*. Verlag Dr. Müller (VDM), 2006. ISBN 3-86550-997-5.

[8] H.-H. Do, S. Melnik, and E. Rahm. Comparison of schema matching evaluations. In *2nd Int. Workshop on Web Databases (German Informatics Society)*, 2002.

[9] H.-H. Do and E. Rahm. COMA - a system for flexible combination of schema matching approaches. In *Proc. 28th Intl. Conf. on Very Large Data Bases (VLDB)*, 2002.

[10] A. H. Doan, P. Domingos, and A. Halevy. Reconciling schemas of disparate data sources - a machine-learning approach. In *Proc. of the ACM SIGMOD Intl. Conf. on Management of Data (SIGMOD)*, 2001.

[11] A. H. Doan, J. Madhavan, P. Domingos, and A. Halevy. Learning to map between ontologies on the semantic web. In *Proc. 11th Intl. World Wide Web Conf. (WWW)*, 2002.

[12] L. Haas, M. Hernandez, H. Ho, L. Popa, and M. Roth. Clio grows up: From research prototype to industrial tool. In *Proc. ACM SIGMOD Intl. Conf. Management of Data*, pages 805–810, 2005.

[13] W. Li and C. Clifton. Semint - a tool for identifying attribute correspondences in heterogeneous databases using neural network. *Data and Knowledge Engineering*, 33(1):49–84, 2000.

[14] J. Madhavan, P. Bernstein, and E. Rahm. Generic schema matching with cupid. In *Proc. 27th Intl. Conf. on Very Large Data Bases (VLDB)*, 2001.

[15] S. Melnik, H. Garcia-Molina, and E. Rahm. Similarity flooding a versatile graph matching algorithm. In *Proc. 18th Intl. Conf. on Data Engineering (ICDE)*, 2002.

[16] T. Milo and S. Zohar. Using schema matching to simplify heterogeneous data translation. In *Proc. 24th Intl. Conf. on Very Large Data Bases (VLDB)*, 1998.

[17] F. Naumann, C. Ho, X. Tian, L. Haas, and N. Megiddo. Attribute classification using feature analysis (poster). In *Proc. 18th Intl. Conf. on Data Engineering (ICDE)*, 2002.

[18] L. Palopoli, Terracina, and D. Ursino. The system dike - towards the semi-automatic synthesis of cooperative information systems and data warehouses. In *ADBIS-DASFAA*, 2000.

[19] E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. *VLDB Journal: Very Large Data Bases*, 10(4):334–350, 2001.

[20] W3C. Owl web ontology language. online, 2004. http://www.w3.org/TR/owl-guide/.

[21] L. Xu and D. Embley. Discovering direct and indirect matches for schema elements. In *Proc. 8. Intl. Conf. Database Systems for Advanced Applications (DASFAA)*, 2003.

Table 7: The mapping categories supported by QuickMig

Category	Description	Mapping code generation	Category determination
CreateInstance	A new entry in the target system is created for every entry in the source instance or whenever the source element is not empty.	automatic	automatic
KeyMapping	Source schema element and target schema element are the identifiers of an instance. A Mapping of this identifier is required.	automatic	automatic
InternalID	The target schema element contains the internal ID of an instance. Dependent on the applied strategy, the internal ID must be drawn from a number range or the content of the source schema element is used as the internal ID of the target instance (e.g. `KUNNR` mapping in running example).	semi-automatic (strategy needs to be chosen)	automatic
LookUp	The target schema element contains an internal identifier of another object. In this case multiple fields of the source schema are used in order to find the correct identifier.	automatic	automatic
Move	Move source schema element content to target schema element (e.g. `ANRED`).	automatic	automatic
ValueMapping	The target schema element contains a code according to a specific code list. Therefore a value mapping is needed. If the source element text is found in the the code list associated to the target schema element, the corresponding code is used; if not a translation table is necessary (e.g. `ANRED` is a value mapping using a custom translation table, whereas `LAND1` uses a standard code list).	semi-automatic (translation table to be maintained)	automatic
Code2Text	The source schema element contains a code, the target schema element a text. A translation table is necessary to translate the code values into text. This translation table needs to be maintained manually (or imported from the source system code list).	semi-automatic (translation table to be maintained)	manual
DefaultValue	The target schema element is defaulted with a special value whenever the source schema element is not empty.	automatic	automatic
Split	The source schema element content is split into multiple target schema elements based on a specific split algorithm	automatic	automatic
Concatenate	Multiple source schema elements are concatenated into one target schema element.	automatic	automatic
Complex	A complex mapping expression is required.	manual	manual

Listing 1: Excerpt of the used ontology - showing annotations

```
Class(Name partial owl:Thing restriction(hasFormOfAddressCode someValuesFrom(FormOfAddressCode)))

Class(StandardCodeList partial owl:Thing restriction(isManagedBy cardinality(1))
      restriction(a:hasName cardinality(1)))

Individual(StandardCodeList_FormOfAddress type(StandardCodeList)
   value(hasName "FormOfAddressCodes"^^string)
   value(isManagedBy "SAP"^^string))

Class(FormOfAddressCode partial)
AnnotationProperty(usedStandardCodeList)
ObjectProperty(hasFormOfAddressCode)
ObjectProperty(usedStandardCodeList)
DatatypeProperty(hasName)
DatatypeProperty(isManagedBy)
```

Table 8: Composition of mapping categories in the reuse matcher. The column headings represent the mapping categories between T_1 and T_2 and the row headings the mapping categories between S_1 and T_1

	CreateInstance	KeyMapping	InternalID	LookUp	Move	ValueMapping
CreateInstance	CreateInstance	-	-	-	-	-
KeyMapping	-	KeyMapping	-	-	-	-
InternalID	-	-	InternalID	-	-	-
LookUp	-	LookUp	-	-	-	-
Move	CreateInstance	KeyMapping	InternalID	-	Move	ValueMapping
ValueMapping	-	-	-	ValueMapping	ValueMapping	-
Code2Text	-	-	-	-	Code2Text	-
Default	-	-	-	-	Default	Default
Split	-	-	-	-	Split	-
Concatenate	-	-	-	-	Concatenate	-
Complex	Complex	Complex	Complex	-	Complex	Complex