Towards Domain-Oriented Semi-Automated Model Matching for Supporting Data Exchange

H. Wang, B. Akinci, J. H. Garrett, Jr.

Department of Civil and Environmental Engineering

Carnegie Mellon University, USA

O. Akin, M. T. Turkaslan-Bulbul, I. Gursel

School of Architecture

Carnegie Mellon University, USA

Summary

The process of matching data represented in two different data models is a long-standing issue in the exchange of data between different software systems. While the traditional manual matching approach cannot meet today's demands on data exchange, research shows that a fully automated generic approach for model matching is not likely, and generic semi-automated approaches are not easy to implement. In this paper, we present an approach that focuses on matching data models in a specific domain. The approach combines a basic model matching approach and a version matching approach to deduce new matching rules to enable data transfer between two evolving data models.

1. Introduction

The exchange of data among different data models is a long-standing challenge in the application of information technology in almost all engineering-oriented domains. When one application tends to share data with others, data must be converted from a specific problem-oriented data model to another public more generic data exchange standard. This leads to a need to represent many different specific data models using the public data exchange standard without losing data. Since it is highly unlikely to change an internal data model according to a standard public model, matching of the data represented internally to the standard way of representing those data is necessary. The challenges in such data exchange become more pronounced when the specific problem-oriented data model and /or the public data model are changing frequently.

Traditionally, such model matching activities are performed by domain experts and it is knowledge-based and time-consuming work. This manual matching approach is becoming an increasingly common and difficult problem given the rapidly growing use of specific domain-oriented data models. Furthermore, since some of these specific domain-oriented data models and the data exchange standards themselves are continuously evolving, a manual model matching approach becomes an unsustainable solution to solve data exchange problems.

Several researchers have worked on developing intelligent approaches to convert one data model to another (Li et al. 1994, Madhavan et al. 2001, Milo et al. 1998, Mitra et al. 1999, Palopoli 1998). Prior experience on this area shows that a general-purpose fully automated approach for model matching is extremely difficult and impractical

(Rahm et al. 2001). It is difficult for a software application to understand the data representations for different models since the models might differ both semantically and syntactically.

A semi-automated matching approach is much more achievable, especially when performed within only one specific domain. Human intervention can improve model matching from two aspects. First, a human expert is able to set up a matching context, by applying domain constraints or configuring heuristic parameters, to speed matching. Second, a domain expert can correct some errors during the matching procedure then train that matching procedure to avoid future errors. Therefore, compared to a fully automated approach, a domain-specific semi-automated approach that utilizes prior matching knowledge and domain knowledge will undoubtedly lead to better performance and accuracy.

The goal of the research discussed in this paper is to develop approaches to improve the data model matching process by utilizing prior matching rules and domain knowledge. This specific research activity is part of a larger research project to evaluate the effectiveness of the Industry Foundation Classes (IFC) data exchange standard in supporting Building Commissioning (BC) related activities.

The IFC data exchange standard is an effort initiated by International Alliance for Interoperability (IAI) to enable interoperability between different software systems in the Architectural/Engineering/Construction (A/E/C) and Facilities Management (FM) industries (IAI 2003). Building commissioning is a systematic process of determining that interacting building systems and components perform consistently with design intent and the owner specified performance requirements (Oregon Office of Energy 1997). A continuously evolving BC data model is being created in this larger research effort and we have been exploring how well the IFC data exchange standard can represent the data items in the BC model. Since both data models, the BC and the IFC, have been changing frequently, we are testing the ability of the most up-to-date IFC release to represent the most up-to-date version of the BC model using a semi-automated test rig. A primary requirement of this test rig is a means to perform a quick and effective data model matching process between the BC and IFC models. When developing this test rig, we keep in mind that the issue of matching between two models has far more implications than the matching of BC and IFC models and hence we are working on generalizing our approach and the lessons-learned for a general model-matching domain.

2. Background

2.1 Semi-Automated model matching approaches

Model matching produces a correspondence between elements of two input models (Rahm et al. 2001). It is a common research topic in the area of data integration, e-business and data mining. Manual matching is a tedious and time-consuming job. When the scales and complexity of models become larger, manual matching suffers from being burdensome and error-prone. Several existing research projects demonstrate that a semi-automated schema matching approach is a good means of providing some support for manual data model matching (Li et al. 1994, Madhavan et al. 2001, Milo et al. 1998, Mitra et al. 1999, Palopoli 1998).

Individual algorithms, such as linguistic comparison, structure analysis, and combination of multiple matching approaches, have been verified to work well (Rahm et al. 2001). A widely used linguistic comparison approach is name-based matching. It matches elements with equal or similar names. This approach often requires a thesauri or dictionary to find equality of synonyms or acronyms. Another common matching approach is structure-level comparison. It calculates structure similarity of the contexts in which the elements occur in two models using a measure of similarity (Madhavan et al. 2001). For example, it searches for similarities of the patterns formed by the model element, its ancestor, siblings and child elements in two models. Some approaches also utilize artificial intelligence algorithms, such as machine learning, which extracts data model matching knowledge from training cases selected by domain experts (Doan et al. 2001).

2.2 Version matching approaches

Besides model matching, research based on comparing different versions of the same data model is also relevant in this research. When a new version of either the BC model or the IFC standard is released, version matching is a viable approach. Version matching is the matching between different releases of the same model, so it is actually a specialization of the matching between different models (Ge, 2002). Model evolution is a common issue since both problem-oriented and generic models are often changed to take account of new content that must be represented.

In the context of our research, both the source model, the Building Commissioning data model, and the target model, the IFC data exchange standard, are under development and are frequently changing. Compared to a data model matching approach, a version matching approach is relatively easier to implement because a new version of a data model is usually built upon its previous version. Therefore, given two frequently changing data models associated with the context of our research domain, one goal is to determine how to obtain matching knowledge from the last matching process applied between the previous two model versions.

An IFC version matching framework, created by Amor and Ge, was able to verify that complexity of matching between IFC versions can be reduced significantly because of the fact that a new IFC version is built upon its prior version (Amor and Ge 2002). Test cases show that more than 65% of the IFC entities and types in release R2.0 can be matched automatically from release 1.5.1 using basic algorithms, such as name matching and structure comparison. After knowing the difference between versions, a new set of matching rules can be deduced from the previously used set.

2.3 Domain knowledge-based approaches

It is also our intention to further support the data model matching activities using domain knowledge and constraints. The integration of domain knowledge and constraints, which are often not embedded in the data models themselves, can improve matching accuracy by aiding software in removing ambiguous matching results. Using domain knowledge also results in utilizing basic matching algorithms, for example by supplementing the dictionary used in name matching with specific terms from the domain being represented by the data models.

The LSD (Learning Source Description) system uses an improved machine-learning algorithm that combines matching results of multiple basic matching algorithms (Doan et al 2001). It allows domain constraints, which provides supplement model properties, e.g. frequency that a specific term could be used, to be incorporated with basic matching algorithms, as an additional source of knowledge. Such additional usage of domain knowledge and constraints can improve the accuracy by 7-13% in its test cases (Doan et al, 2001). Since our research focuses on models within one specific domain, we expect domain constraints can boost the matching accuracy.

3. Towards Domain Oriented Semi-Automated Model Matching

3.1 Research Goals

In this research, we are designing and implementing a test procedure to evaluate the degree to which the IFC data exchange standard supports the exchange of BC data in the post-construction and facility management phase of building commissioning. This test procedure is intended to provide a quick and effective assessment of the degree to which BC data is able to be matched to the IFC data exchange standard when BC models and IFC releases are frequently changing.

We plan to generate matching rules from existing matching knowledge and combine that knowledge with results from version matching. That is, given 1) matching rules between a version of source model, S1, and target model, T, and 2) the difference between S1 and a new version of source model, S2, we can deduce new rules for matching S2 to T (Rahm et al 2001). This approach is expected to work better than generic approaches to model matching because it can get support from domain knowledge and existing matching rules that are created manually from the first one or two versions of a data model.

3.2 Research Roadmap

This research project is a multi-year, multi-stage endeavor. For the entire project, we will create three prototypes of the BC-IFC matcher using three increasingly difficult approaches based on where and how matching rules are declared.

1. *Statically Embedded Rules*: The matching rules are embedded in the source code of matcher application because it is easy to implement such a prototype. The purpose of this stage is to discover initial matching rules manually and test the effectiveness of using these deduced matching rules. This prototype helps us explore how well new matching rules can be deduced from old ones.
2. *Manually Generated Declarative Rules*: This is a transition step between manually matching and semi-automated matching. We will extract matching rules from the source code created in the last stage and select a proper representation of these rules to separate the matching rules from the corresponding implementation codes. The matching rules will be stored in an external file that contains manually declared rules and that could be updated independently. Domain knowledge is also determined and added at this stage.
3. *Semi-Automated Generated Declarative Rules*: The purpose of this stage is to develop a matcher application to generate matching rules desired by the matcher application. Compared to manually generated rules in approach 2, the semi-automated matcher attempts to deduce matching rules automatically where possible. The difference between versions could be obtained by comparing their

semantic meanings and structures. Given prior matching rules and differences between a new version and the prior one, the matcher can deduce new matching rules.

To date, we have completed the first level approach. A Java application has been developed to test the data model matching between a version of the BC model and three recent releases of the IFC data exchange standard: R2.0, R2x and R2x2. Based on the experience and the lessons learned from this matching approach, we are in the process of implementing the second prototype focusing on manually generating declarative rules.

4. Approach for Statically Embedded Matching Rules

4.1 Manual creation of statically embedded rules

To implement the first level of the data model matching approach, we developed matching rules by comparing the BC model and the three IFC data exchange releases manually. We compared the BC entities and their attributes to each release of the IFC data exchange standard to build three IFC class diagrams that represent the BC model. To evaluate the effectiveness of the matching process, we define three levels of matching:

- *Fully matched* indicates that the IFC release not only has a category to represent the class or attribute in the BC model, but also possesses a proper entity to represent it exactly.
- *Partially matched* means that the BC and the IFC models have different representations for the same class or attribute that could be matched effectively, but at least one constraint, for example value type or the scope of a value, is not matched. For example, the location attribute of the equipment entity in the BC model is a string type, while IFC adopts a geometric type of data to represent such an attribute.

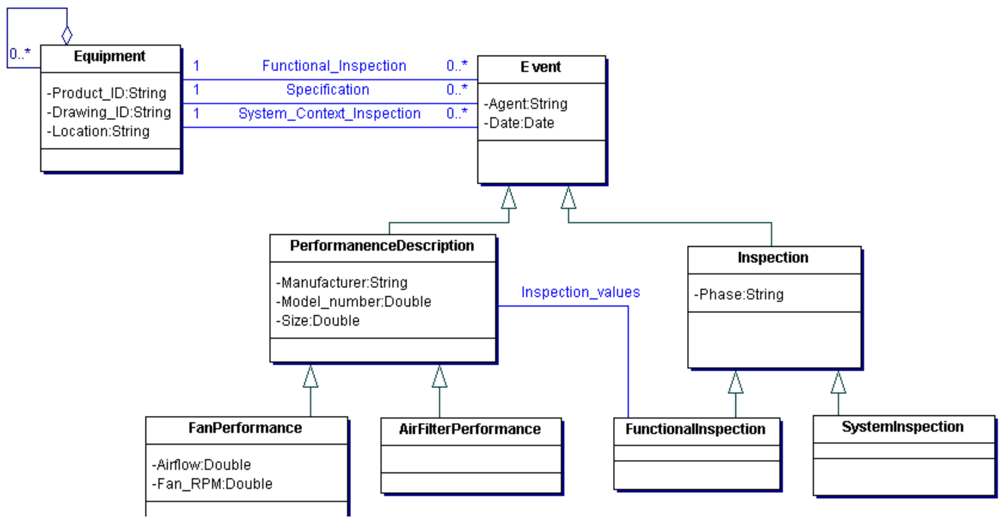

Figure 1 Partial Building Commissioning Data Model

- *Not matched* means that neither a class nor an attribute was available in the corresponding IFC release to provide a representation of the BC attribute or a class.

Among all current BC entities, relationships and attributes in the BC model (see Figure 1), about 30 percent of those data items can be fully matched to the entities in the most recent release of the IFC data exchange standard, R2x2. For example, in IFC R2x2, BC Equipment entity, which stands for HVAC equipment, is represented by IfcElement and Event entity, which stands for building commissioning activity, is expressed by IfcTask. IfcRelAssignToProcess entity is used to represent three kinds of relationships between Equipment and Event entities: Specification, System_Context_Inspection, and Functional_Inspection.

4.2 System architecture of matcherframework

We then developed a prototype data model matching application that embedded the matching rules learned from the manual matching activity described in Section 4.1. Figure 2 presents a system component diagram of our first level BC-IFC data model matcher prototype. It contains the following major components:

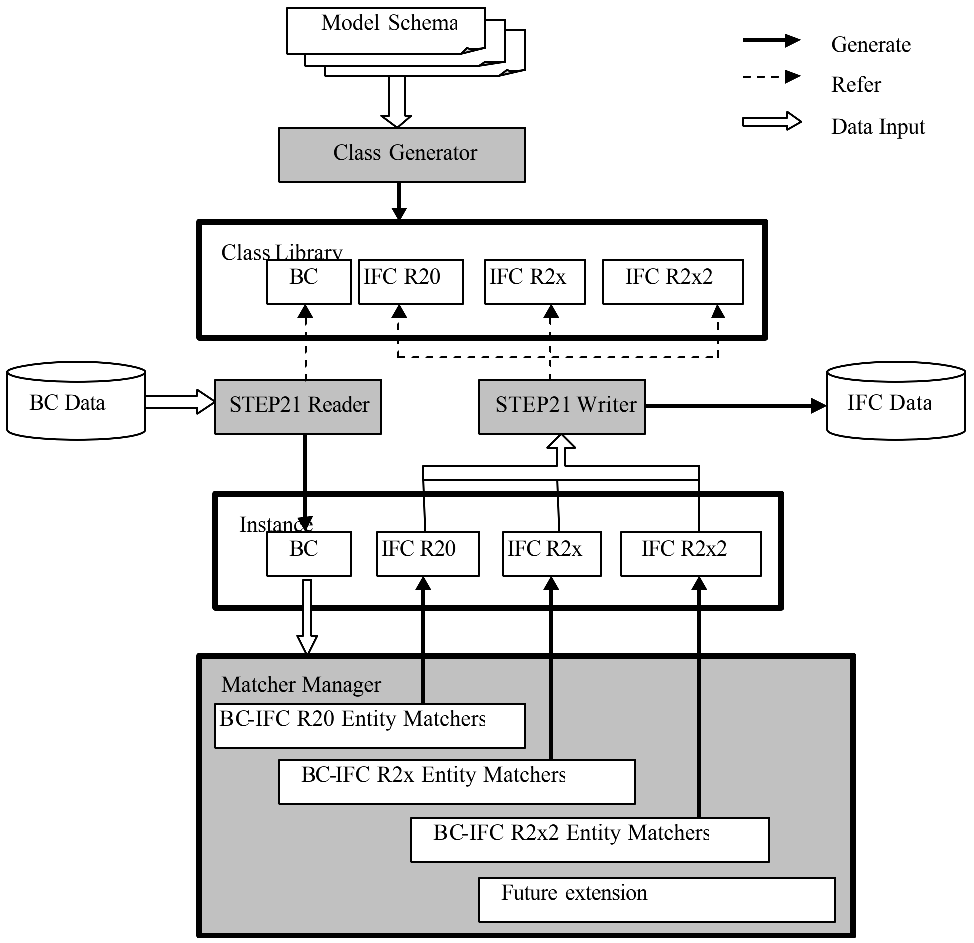

Figure 2 System Components of Matching BC to IFC

- *Class Generator*: This component is a utility to generate Java class library files from a model definition file which is saved in EXPRESS (ISO 1994) language. The class generator parses an EXPRESS schema file and creates Java class definition for each declared entity and type in the model. It gives us capability of updating class library definition automatically when the BC data model or the IFC data exchange standard changes. As a result, this ensures the class library definition consists of the most up-to-date models and speeds our development by reducing code generating time significantly.
- *Entity Matcher*: This component applies the matching rules to convert instances of one data model to instances of another data model. A data model matcher processes the entities existing in one model and generates corresponding entities in the other model. One matcher only serves one type of BC entities and generates result entities of one specific version of the IFC data exchange standard. That is, for each BC entity, we develop three entity matchers to generate IFC R20, R2x and R2x2 entities respectively. The input parameter of entity matcher is a BC object and the output is a set of IFC objects that represent the corresponding BC object.
- *Matcher Manager*: This component manages and coordinates the matcher application. It specifies a public interface through which external applications can operate entity matchers. It provides for entity matchers by defining infrastructure where the matcher works, establishing workflow of matching procedure and the internal interfaces through which one component can invoke services of other components.
- *STEP21 Reader/Writer*: This component is a STEP21 (ISO 2002) format data file parser for Java. The reader parses a STEP21 data file and generates instances for each declared entity, while the writer can output the class instances to a data file in STEP21 format.

4.3 Description of Current Matching Workflow

To perform data model matching, the first step is to generate a Java class library. Both the BC model and the IFC data exchange standard are defined in the EXPRESS language files. The class generator parses these files to create a Java class definition for each declared BC or IFC entity and type. This is a one-time task, only required when a new version of a model is released.

The second step is to input the source data model. The STEP21 reader reads a data file where BC data is represented in STEP21 format. For each declared BC entity, the reader looks up correct Java class definition generated in the last step, then constructs an instance of that class in Java and initializes this BC instance with the values in the data file.

The third step has the matcher manager iterating over all BC instances. Given the target IFC version and which BC entity is being matched, the manager selects an appropriate entity matcher for each BC instance. The entity matcher uses embedded matching rules to create desired IFC instances.

The fourth and last step is to save the IFC results in a data file by the STEP21 writer. SInce the output is in STEP21 format, any applications that support the STEP21 standard can parse this file.

4.4 Experience and Lessons Learned from the Manual Matching

The current BC model is not a complicated and a large-scale domain-specific model, however, it was still a very time-consuming job to find manually all matching rules between the BC model and the IFC exchange standards. It is difficult to locate one IFC attribute that exactly matches a required value type, the scope of a value and semantic meaning, especially if one recognizes that the IFC may have multiple ways to represent the same concept in the BC model.

The concept of implementing matching rules in the source code is straightforward because it statically declares matching rules within the matcher classes. However, this approach has major limitations when one has to change the existing code to reflect the matching between a new BC model and/or a new IFC release. We usually obtain new matching rules by merging changes between two releases, such as name changing, into the existing matching rules. For example, the BCEvent entity of the BC model is matched to the IfcWorkTask entity under IFC R2.0. Given the fact that IfcWorkTask is renamed to IfcTask in IFC R2x, we can match the BCEvent entity to IfcTask under IFC R2x. When a new BC attribute is inserted, we had to manually re-scan each IFC schema to locate proper counterparts and then re-write, re-compile and re-deploy each individual matcher class in the first level matcher prototype. Additionally, although the IFC data exchange standard has a relatively stable core platform, some of its external domains such as the HVAC domain, change significantly between versions. This situation makes our test procedure utilizing statically embedded rules less effective and encourages us to develop an automated or semi-automated approach to speed up the matching procedure.

Meanwhile, the manually declared matching rules for three IFC releases verify that both the BC model and the IFC model could apply a version matching approach to their new version to find the difference between the versions. The fact that a new version is oftentimes built upon its prior version makes an automated or semi-automated version matching approach possible. Among the IFC entities and attributes related to building commissioning, over 60 percent of classes and data entities between the last two versions of IFC were overlapped.

5. Proposed Semi-Automated Approach

Figure 3 illustrates an initial architecture of our proposed semi-automated matching approach developed based on what we learned from the first level prototype described in Section 4. It is our assumption that the two data models being matched are in the same application domain, for example in the A/E/C domain. The approach is intended to work at the schema level and will be supported by dictionary and thesauri based matching approach and existing matching rules to find matches between the data models.

First, the identified data model, either the source model or the target one, will be parsed into an internal model representation, which records necessary properties of each element of the model, such as its superclass, attribute list, value type and value constraints. Then, a version matcher will compare this internal representation with the representation of the model's prior version, to find the differences, if there is any, between these two versions. Linguistic matching and structure analysis are two possible version matching algorithms that we are intending to use to compare different versions. If a version change log exists, it will be an important source to aid matching.

After version matching, new matching rules for the source and target data models can be generated via a variety of approaches. For example, one is to combine the version difference information obtained in the last step with existing matching rules for old versions to deduce new matching rules. Another approach is to continuously use linguistic matching, structure analysis and/or artificial intelligence algorithms to build matching rules. These approaches could be applied either jointly or separately and both approaches will use domain knowledge to improve their accuracy. One example of domain knowledge is additional thesauri and dictionaries to support better name matching. The final stage of result checking will still be the responsibility of the domain expert, who will be able to correct matching errors and to pick the proper matching result from a list of possible matches. Finally, the newly generated matching rules will be recorded as existing knowledge to be used the next time this version must be matched to another model.

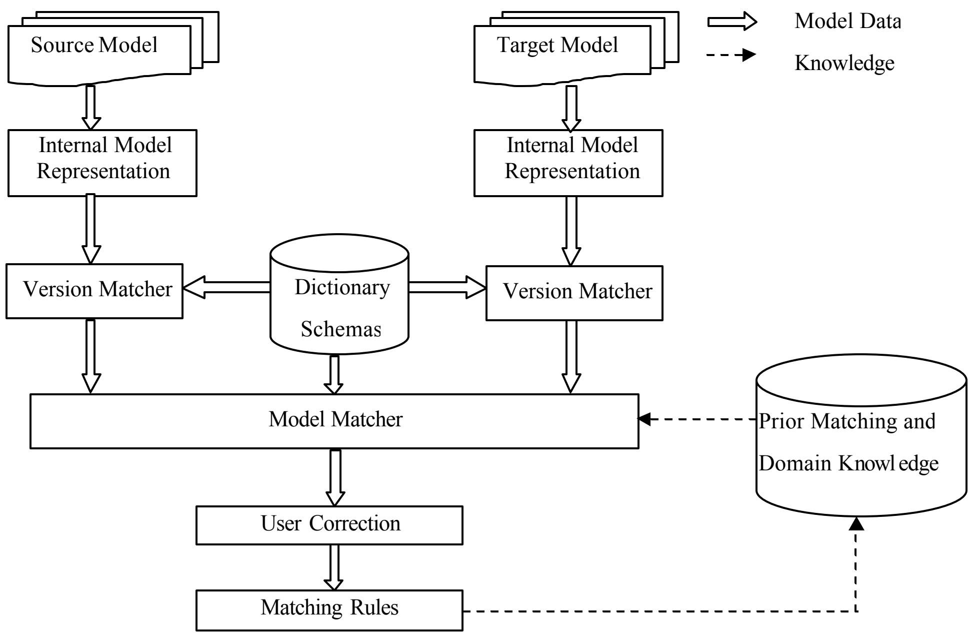

Figure 3 System architecture of domain matcher

6. Conclusion

In this paper, we discussed our ongoing research to perform fast and effective specific domain model matching. Experience and lessens learned from our manual model matching clearly demonstrate the need for an approach to conduct domain-oriented model matching effectively, especially when models are changing frequently. This approach combines basic model matching and version matching approaches. Compared to a generic model matching approach, our approach applies domain knowledge and constraints to improve matching accuracy. We also present our proposed research roadmap to implement a semi-automated domain-oriented data model matching approach.

Acknowledgements

The material in this paper is based upon work supported by the National Institute for Standards and Technology under Grant No.60NANB2D0158. We thank Kent Reed and George Kelly from NIST for their professional insights in this research.

Reference

Akin, O., Turkaslan-Bulbul, T., Brown, S., Kim, E., Akinci, B., Garrett, J. (2003), *Comparison of ASHRAE Guidelines with Building Commissioning Practice*, National Conference on Building Commissioning, California

Amor, R.W. and Ge, C.W. (2002), *Matching IFC Versions*, Proceedings of the EC-PPM Conference on eWork and eBusiness in AEC, Portoroz, Slovenia

Doan, A., Domingos, P., and Levy, A. (2001), *Reconciling Schemas of Disparate Data Sources: A Machine-Learning Approach*, Proc ACM SIGMOD Conf

Ge, C.W. (2002), *A Semi-Automated Schema Version Mapping Framework* , MSc thesis, University of Auckland, Auckland, New Zealand,

ISO (1994), *ISO 10303 Part 11 Description methods: The EXPRESS language reference manual*, http://www.iso.org

ISO (2002) *ISO 10303 Part 21 Implementation methods: Clear text encoding of the exchange structure*, http://www.iso.org

IAI (2003) *Industry Foundation Classes*, http://www.iai-international.org

Li, W, and Clifton C (1994), *Semantic integration in heterogeneous database using neural network*. In: Proc 20th Int. Conf On Very Large Data Bases, pp. 1-12

Madhavan, J., Bernstein, P.A., and Rahm, E. (2001), *Generic Schema Matching with Cupid*. In: Proc 27th Int. Conf. On Very Large Data Bases, pp 49-58

Milo, T, and Zohar, S. (1998), *Using Schema matching to simplify heterogeneous data translation*. In: Proc 24th Int. Conf On Very Large Data Bases, pp. 122-133

Mitra, P., Wiederhold, G. and Jannink, J. (1999), *Semi-automatic integration of knowledge sources*. In: Proc of Fusion'99, Sunnyvale, USA

Palopoli, L., Sacca D., Ursino D. (1998), *Semi-automatic, semantic discovery of properties from database schemas*. In: Proc Int. Database Engineering and Applications Symp. (IDEAS), IEEE Computer, pp244-253

Rahm, E. and Bernstein P. A. (2001). *A survey of approaches to automatic schema matching*. VLDB Journal, 10:334--350.