A SEMI-AUTOMATED SCHEMA MATCHING APPROACH BASED ON AUTOMATED DECTION OF VERSION DIFFERENCES

Hongjun Wang[1], Burcu Akinci[2], and James H. Garrett Jr.[3]

ABSTRACT

The process of matching data represented in different data models is a long-standing issue in enabling interoperability between different software systems. The challenges associated with model matching become even more pronounced when a source or a target model is being updated frequently. Such situations generate a need for handling the matching process efficiently. Some prior computer-aided approaches have performed a considerable portion of the schema matching effort automatically and produced satisfactory results under certain circumstances. However, determining how to reuse existing matching knowledge has rarely been studied. In this paper, we present a semi-automated approach to perform efficient schema matching in a situation, where a data exchange standard is frequently updated. This approach builds on automated version detection and utilizing existing matching knowledge in creating the new matching correspondences. A taxonomy is developed to precisely identify differences between different versions of the same data model. We apply these differences to update existing matching results between prior versions and generate new matching results between the new versions of the data models. A set of matching patterns are developed to define what the new matches should be created based on existing matches and identified categories of version differences. The approach is validated through matching a domain specific data model to the recent releases of the Industry Foundation Classes (IFC).

KEY WORDS

Interoperability, Data exchange standard, Model matching, Semi-automated approach

INTRODUCTION

The need for exchanging data between computer applications has existed for decades in the Architecture, Engineering, and Construction (AEC) industry. Data exchange among applications involves translating data, which is specific for one application, into data that can be understood by another application. It requires that the target data model represents the source data as accurately and completely as possible to minimize data loss during exchange (Fagin et al., 2003). This requirement arises in cases where various applications are involved during an AEC project and data sharing is desired between these applications.

[1] Ph.D. Candidate, Dept. of Civil and Environmental Engineering. Porter Hall 118D. Carnegie Mellon Univ. 5000 Forbes Ave. Pittsburgh, PA 15213, Phone +1 412/268-6634 hongjunw@andrew.cmu.edu

[2] Assist. Professor, Dept. of Civil and Environmental Engineering. Porter Hall 123F. Carnegie Mellon Univ. 5000 Forbes Ave. Pittsburgh, PA 15213, Phone +1 412/268-6218 bakinci@andrew.cmu.edu

[3] Professor, Dept. of Civil and Environmental Engineering. Porter Hall 123A. Carnegie Mellon Univ. 5000 Forbes Ave. Pittsburgh, PA 15213 Phone +1 412/268-5647 garrett@andrew.cmu.edu

The AEC industry is recognized as a multi-disciplinary and multi-participant industry. It involves users like owners, designers, contractors and inspectors, who deploy many different domain-specific applications. Inefficient interoperability can result in a marked impact in the real world. For example, a recent NIST report indicated that the cost of inadequate interoperability in the U.S. capital facilities industry is up to $15.8 billion per year (NIST 2004).

To enable interoperability between different software systems in the AEC industry, data must be smoothly exchanged between multiple users or applications by some public data exchange standards. Recent development of widely accepted standards includes Industry Foundation Classes (IFC) (IAI 2005a), ifcXML (IAI 2005b) and CIMsteel (Eureka 2004). For example, the IFC data exchange standard was initialized in 1995 to achieve a significant economic benefit from interoperability of software (IAI 2005a). After 10 years of development and several releases, the IFC data exchange standard is now widely accepted by a large number of major AEC software packages (e.g., CAD systems) (Steinmann 2004).

There does not exist a single data model that can cover all aspects of all data modeling requirements to serve all task specific applications. Therefore, the need still exists to match task-oriented data models to a set of public data exchange standards, such as IFC, without losing or altering relevant data. This creates a challenge in the matching of the models because task-specific data models and public data exchange standards often use distinctly different representational approaches and data structures.

Traditionally, matching of data models is manually performed. Manual matching of data models is time-consuming, error-prone and tedious work. Given the rapidly growing number and scale of data models used in today's applications, manual matching is becoming a much harder task. Moreover, this challenge gets much more pronounced when either data model changes over time. For example, in the last three years, the IFC data exchange standard had undergone two major updates, Release 2x and 2x2, but a large number of IFC-compatible commercial software only supports Release 2.0 or even 1.5 (Steinmann 2004).

It has been demonstrated that a computer-enabled process can provide further help in this matching process. Some prior studies (e.g., Doan et al., 2000, Madhavan et al., 2001, Mitra et al., 2000, Li and Clifton, 2000) could perform a considerable part of schema matching automatically and output satisfactory results under certain circumstances. Two generic computer-aided methods that are widely used by these prior studies are: 1) a linguistic-based approach that finds matched elements using their names or descriptions (e.g., comments extracted from specifications); and 2) a constraint-based approach that considers the similarities of certain constraints, such as data types of an attribute, schema hierarchical structures and relations between elements (Rahm and Bernstein 2001).

One case that has rarely been studied is how to reuse existing matching results (Rahm and Bernstein 2001). In a manual matching process, if a domain expert knows how to match a source schema to a prior version of a data exchange standard, when a new version of the data exchange standard is issued, s/he will not match the source schema to the new release *tabula rasa*. Instead, s/he will certainly try to reuse existing matches wherever possible and only adjust the matches affected by the upgraded version. This process is an efficient and time-saving approach, greatly reducing human workload. However, due to the size and complexity of today's data schemas, even only finding changes between releases is becoming

a difficult and challenging task. It takes a large amount of time and effort to find all differences between an original version and an upgrade of a data model, especially if there is no document (i.e., a change log) that describes version differences. For example, the IFC R2x2 introduces significant changes to the IFC R2x (e.g., the numbers of declared entities and types), however, since there is no official change log published, it took us several days to identify all modifications manually.

In this paper, we developed an approach to partially address the above issue, improving the data model matching process by utilizing prior matching results created manually in a certain AEC domain (i.e., Building Commissioning domain). The proposed approach takes advantage of an automated version matching process developed and applies differences between two releases of the same data model to update existing matching results.

PROPOSED SCHEMA MATCHING APPROACH

The approach discussed in this paper is aimed at improving the current data model matching process by incorporating version differences. A three step procedure is planned in the approach. The first step is to obtain initial matching results between the source data model (S) and the target data model (T1). This step can be performed by this research, other computer-aided matching approaches, or even manual process. When a new version of the target model (T2) is introduced, the second step will precisely identify the differences between the new version of the target data model and its prior version (T1). The third and last step is to merge the identified differences into existing matching results, instead of matching the two models (i.e., S and T2) directly and *tabula rasa*.

Figure 1 illustrates the overall procedure through an example, where a task specific data model that represents some Building Commissioning tasks is involved (referred to as the BC data model) (Wang et al. 2004). The source class is the *BCEvent*, which represents an inspection activity in a commissioning process. In the IFC Release 2.0, it was matched to the class *IfcWorkTask*, which is a unit of work. In the IFC Release 2x, the class *IfcWorkTask* was renamed to *IfcTask*, so the class *BCEvent* was directly matched to *IfcTask* in the IFC Release 2x. Compared to scanning the entire IFC R2x schema to find the proper IFC classes, updating prior matching results can potentially save time and effort and achieve better accuracy.

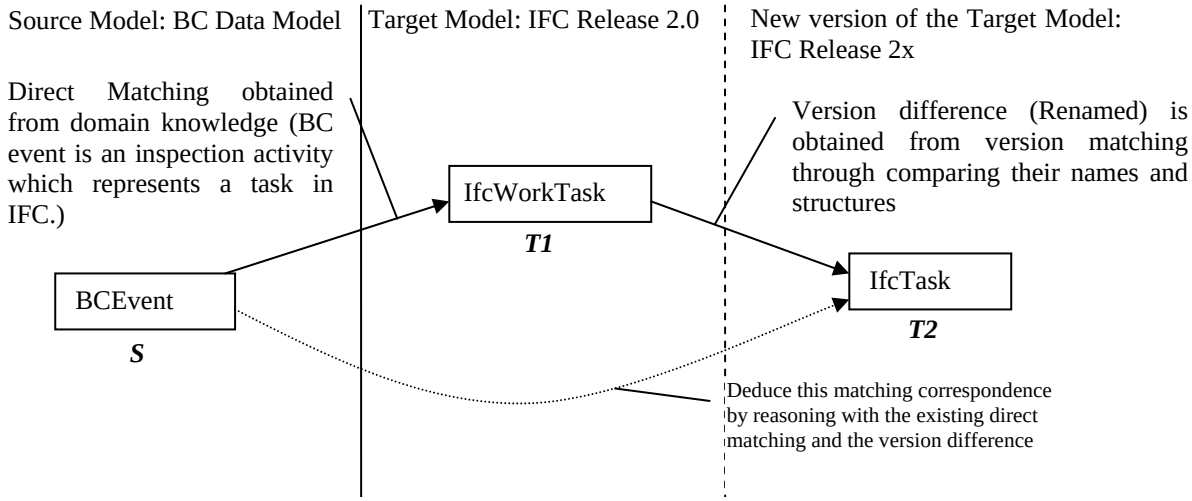

Figure 1 Example of Deducing New Matching Result

First, we developed an approach that detects differences between different versions of the same data schema. Since a new version of a schema usually builds upon its prior version and shares common content, it is reasonable to assume that differences between releases will not be significant and a computer-aided approach can potentially achieve higher accuracy in version matching problem than comparing two schemas *tabula rasa*. A recent research project performed version matching on different subsequent versions of the IFC data exchange standard and demonstrated the possibility of identifying version differences in an automated way (Amor and Ge 2002). However, this prior approach has several limitations. For example, it can only indicate whether or not there is a difference between two classes, without precisely pinpointing the location of the difference and what type of change has occurred. Without such a precise description of version differences, it is hard to build on version matching for semi-automated model matching purposes. In addition, it obtains version differences by only comparing texts of two elements without considering what these texts are standing for, so that it cannot achieve good accuracy when there are significant differences between versions (e.g., IFC R2x2 greatly changed the IFC R2x in terms of the numbers of objects and their contents). We built our approach upon this prior study and addressed its limitations stated above to improve accuracy and generality in versions matching so that it can be utilized for model-matching purposes. We created a taxonomy of version differences to exactly identify which part of an element (i.e., class or attribute) is modified, and then developed a semi-automated version matching approach that applies both linguistic-based and constraint-based schema matching algorithms to detect version differences based on the classifications.

After the version matching step, we integrated the version differences with existing matching results that map the source schema to the previous version of the target schema so as to deduce new matching results. We created a list of patterns, each of which will combine a particular kind of existing matching result and a particular type of version difference to generate a new matching result.

We validated the approach discussed in this paper using real world test cases and compared it to other schema matching approaches and the manual process. Test cases demonstrated that in a situation where a schema is being updated frequently, the approach presented in this paper can perform the matching efficiently and achieve accuracy that is similar to the results obtained when manual matching is performed.

DETECTION OF VERSION DIFFERENCES

A major problem, which hampers reusing existing matching results, is to detect similarities and differences between schemas and determining which part can be reused (Rahm and Bernstein 2001). However, in the version matching case, this problem is relatively less challenging since a new version is likely to build on the previous and utilize most of the same concepts. We developed a classification for identification of version differences between two data models. This classification is developed by considering models that are constructed in an object-oriented manner (e.g., the IFC). The following classification can be utilized to identify similarities and differences at either the class or the attribute levels of two subsequent schemas.

1) *Identical*: An element in the new release is the same as an existing element in the prior release. This means that they have exactly the same names, attributes and inheritances.

2) *Modified*: An existing element is changed in the new release. Under this category, a number of sub-categories are defined to describe changes in name, attributes, constraints or inheritance. These subcategories constitute special cases of modified category and indicate where a change has occurred. A change will be classified as one of these special types if it only contains single associated type of change, or as *Modified* category if more than one modification happens.

3) Added: A new element is inserted in a new release. In addition, the *Added* category has one special case, *Merged*, which means multiple old elements are replaced by a single element in the new release.

4) *Removed*: An existing element is deleted in a new release. Similar to the *Merged* case in the *Added* category, the *Removed* category contains the *Decomposition* special case, which means that an element is split into (i.e., replaced by) several elements in the new release. For example, the IFC R2x uses a single class *IfcPump* to represent pump device, while the IFC R2x2 uses two classes instead to implement the same function; *IfcPumpType* that manages common properties for a specific type of pump and *IfcFlowMovingDevice* that represents an occurrence of a piece of equipment including pump.

We developed a semi-automated version matching approach (referred as VMA) to detect the above-listed version differences and created a prototype to validate VMA. A framework was designed in VMA to incorporate multiple matching algorithms (e.g., linguistic-based or constraint-based algorithms) that are targeting to detect changes between two releases of any Object-Oriented schema regardless of the language in which it is written (e.g., EXPRESS (ISO 1994)).

The VMA was tested by recent releases of the IFC data exchange standard (i.e., IFC R1.5, R2.0, R2x and R2x2) because there are a large number of changes happened between

these releases. The tests showed that VMA could generate matches comparable to manual results in terms of accuracy. In addition, these comparable results are generated in a few seconds, compared to weeks of effort by a manual matching approach. Users only need to review the matching results to confirm and based on the accuracy testing, it is expected that only a few corrections will be required from the users. Therefore, VMA saves significant time when matching two versions of the same model.

Table 1 shows the numbers of classes identified for each type of change in the test case that compares the IFC R2x to the IFC R2x2. Since there is no official change log document for the IFC R2x2, manual matching had to be performed during this research, which took about two weeks to complete the identification of version differences. By contrast, VMA only used less than 10 seconds to achieve similar results. In terms of accuracy, VMA's accuracy is close to 97% at the class level and 99% at the attribute level, which are also much higher than those of prior study (Amor & Ge 2002) whose accuracy is less than 40% at the class level in this case. In addition, VMA can even find a few pairs of related entities that are hard to discover by the manual matching approach because the change is deeply hidden in the schema. For example, the VMA identifies that *IfcConstraintUsage* in the IFC R2x relates to *IfcRelAssociatesConstraint* in the IFC R2x2. Although these two classes have different names and different parent classes, their attributes not only have similar names, but also refer to similar types.

Table 1 Matching Results for classes of IFC R2x – IFC R2x2

	Identical	Renamed	Modified	Related	Added in IFC R2x2	Removed from IFC R2x
VMA	136	0	157	8	322	69
Manual matching performed during the research representing ground truth	133	0	161	12	317	64

UPDATING EXISTING MATCHING RESULTS

Based on the detected version matching results, we could develop a list of patterns that can be used to update existing matching results with associated version differences. The update patterns are organized in three groups for: 1) 1:1 matching case refers to a situation where an existing matching result involves one source element and one target element; 2) 1:N case refers to a situation where an existing matching result involves one source element and multiple target elements; and 3) N:M (including N:1) matching case refers to a situation where an existing matching result involves multiple source elements and target elements. Our research so far has mainly focused on the 1:1 and 1:N cases, but we are also making some progress on handling the N:M case.

The 1:1 matching is the most fundamental case, in which a source object (i.e., class or attribute) is represented by a single object in the target schema. Based on the experiences of matching real world data schemas (e.g., theBC data model and the IFC), there are 17

upgrade patterns for this 1:1 matching case at the class level and another 14 patterns at the attribute level. These patterns are based on the version difference classifications and cover all the possible changes for a target element (i.e., class or attribute) described in the previous section. For example, for the example illustrated in Figure 1, a *Renamed* pattern is defined. .

The 1:N matching is a more complicated case, where multiple associated target classes are involved. The upgrade pattern for this case is a composition of individual patterns defined for 1:1 case. Each target class will be updated individually and connections between the existing target classes will be rebuilt if possible.

N:1 and N:M matching are hard problems in the schema matching field (Rahm and Bernstein 2001). In this approach, the N:1 matching uses the same update pattern defined for 1:1 matching, with the N source classes treated as a whole. If there is a case of N:M matching, it is converted into N times 1:M matching, each of which could be processed by an individual pattern discussed above.

We have tested the capability of our approach in identifying schema matching between a model developed for a building commissioning domain (BC data model) and various versions of IFCs. In the test case of matching the BC data model to the releases of the IFC data exchange standard, the initial matching between BC data model and IFC R2.0 was performed manually, which took about two days to match all 19 BC classes to the 11 corresponding classes in IFC R2.0, contains 290 classes totally. Then, VMA detected differences between IFC R2.0 and IFC R2x, identifying 3 fully identical classes, 1 renamed class, 221 modified classes, 65 deleted classes and 145 added classes. Finally, the updated match results were achieved by applying the upgrade patterns. Although most of matches involved 1:N cases, 10 of the 11 involved IFC R2.0 classes were successfully updated. Only one class was not updated correctly, but the matching was close to the expected result. At the attribute level, although manual work was required to find matched target attributes for a few source attributes, the overall accuracy of the automatically generated results was close to 96% in this case.

Figure 2 illustrates an example of updating matches for the *CentrifugalFanContext* class in the BC data model. The upper part lists the manually created prior matching results (i.e., required IFC R2.0 classes) and the lower part shows the automatically updated matching results. In this case, three out of six IFC R2.0 classes (i.e., *IfcActor*, *IfcOrganization* and *IfcPropertySet*) are identical between the two versions and the other three classes (i.e., *IfcWorkTask*, *IfcRelActsUpon* and *IfcRelAssignsProperties*) are modified in the IFC R2x, in terms of names, attributes and/or inheritances.

Compared to the manual approach, our approach is more efficient and can achieve comparable accuracy. Compared to other computer-aided approaches that compare two schemas *tabula rasa*, our schema matching approach has following features:

1) It builds on a version matching process, making it possible to use a computer-aided approach to identify version differences quickly and accurately.

2) It treats existing matching results as repositories of human knowledge and assume they haven already been proven correct and could be reused if the new target schema is similar to the prior target schema (i.e., two versions of the same schema). Therefore, the approach discussed in this paper can reuse existing human knowledge to process some hard problems

(e.g., 1:N matching), because a problem like 1:N matching might also be existing in the matching between prior versions and have already been solved by other computer-aided approaches or manual work.

3) Additionally, although the version matching is already a simpler problem, the overall accuracy could possibly be further improved because not all of the detected version differences are related to the existing matching results. For example, IFC R2x brings more than 400 changes (e.g., modified, added or deleted) to IFC R2.0, however, in the above test case, only less than 40 changes actually relate to the initial 11 classes represented in IFC R2.0 and are used by the upgrade patterns to update the existing matching results. Hence, even if a version difference is not detected correctly, or not disclosed at all, it may not impact the upgrade process that follows because the update patterns probably do not require this specific difference.

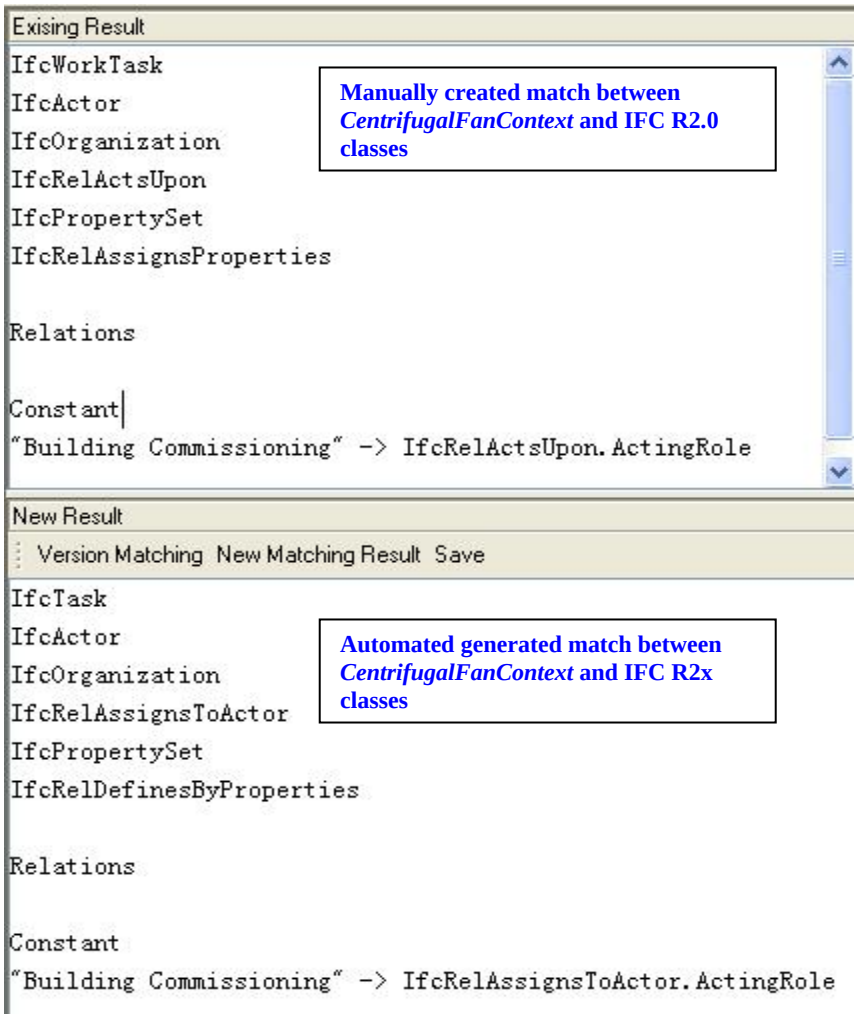

Figure 2 Example of existing matching result and updated results

Therefore, it is reasonable to expect that the schema matching approach discussed in this paper can achieve a higher accuracy than other computer-aided approaches that compare schemas *tabula rasa*, in the cases where either the source schema or the target schema is changed frequently. In addition, the approach does not apply any knowledge specific for a domain (e.g., the building commissioning), so that the generality of this approach is also improved to process other object-oriented data schemas as long as the initial matching between two corresponding schema exists.

CONCLUSION

Performing matching of two data models efficiently is challenging and yet critical in enabling interoperability between different software systems. In this paper, we presented a semi-automated approach that addresses this challenge in a specific domain, where the source or the target data model is being upgraded frequently. We tried to apply existing matching results to help the schema matching process, which is rarely studied in previous research studies.

Current research results have already demonstrated that reusing existing matching results will significantly reduce human workload on model matching and achieve comparable accuracy. In the test case presented in this paper, the approach can automatically detect version differences, much faster than the manual process with an accuracy of more than 95%. When updating existing matching results, the developed approach can complete the process within a few seconds and the accuracy is more than 90% at the class level and 96% at the attribute level. Compared to other computer-aided approaches that match two schemas *tabula rasa*, the approach discussed in this paper produces better outputs and handles some hard matching problems (e.g., 1:N and N:1 matching) more efficiently.

ACKNOWLEDGMENTS

This research is based upon work supported by the National Institute for Standards and Technology under Grants No.60NANB2D0158 and No.70NANB3D1114. We are also grateful for the support from Professor Robert W. Amor.

REFERENCES

Amor, A.W., Ge, C.W. (2002) *Mapping IFC Versions*. In: Proc of the EC-PPM Conference on eWork and eBusiness in AEC, Portoroz, Slovenia, 9-11 September, pp.373-377

Doan AH., Domingos P., Halevy A. (2001) *Reconciling schemas of disparate data sources: a machine-learning approach*. In: Proc ACM SIGMOD Conf. pp.509-520

Eureka CIMsteel Project (2004). *CIMsteel Integration Standards*. Last accessed Nov 2004. http://www.cae.civil.leeds.ac.uk/past/cimsteel/cimsteel.htm

Fagin, R., Kolaitis P.G. and Popa, L. (2003) *Data Exchange: Getting to the Core*. Proceedings of the 22nd ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, pp90-101, San Diego, California

International Alliance for Interoperability (2003) *Industry Foundation Classes*, Last accessed Nov 2004. http://www.iai-international.org

International Alliance for Interoperability (2003) *ifcXML Project*. Last accessed Nov 2004. http://www.iai-na.org/

International Standard Organization (1994) ISO 10303-11:1994: Industrial automation systems and integration -- Product data representation and exchange -- Part 11: Description methods: The EXPRESS language reference manual.

Li, W., Clifton, C. (2000) *SemInt: A tool for identifying attribute correspondences in heterogeneous databases using neural network*. Data Knowledge Engineering 33(1):49-84

Madhavan, J., Bernstein, P.A. and Rahm, E. (2001) *Generic Schema Matching with Cupid*. In: Proc the 27th VLDB Conference, Roma, Italy, 2001

Mitra P., Wiederhold G. and Kersten M. (2000) *A graph-oriented model for articulation of ontology interdependencies*. In: Proc Extending Database Technologies, Lecture Notes in Computer Science, vol. 1777. Springer, Berlin Heidelberg New York, 2000, pp. 86-100

National Institute of Standards and Technology (2004) *Cost Analysis of Inadequate Interoperability in the U.S. Capital Facilities Industry*. Last accessed Nov 2004. http://www.bfrl.nist.gov/oae/publications/gcrs/04867.pdf

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

Steinmann, R. (2004) *International Overview of IFC-Implementation Activities*. Last accessed Nov 2004. http://www.iai.fhm.edu/ImplementationOverview.htm.

Wang, H., Akinci, B. Garrett, J. H. et al. (2004) *Towards Domain-Oriented Semi-Automated Model Matching for Supporting Data Exchange*. In Proc: The 10th International Conference on Computing in Civil and Building Engineering, Weimar, Germany.