Supporting standard data model mappings

R.W. Amor
Department of Computer Science, University of Auckland, Auckland, New Zealand

ABSTRACT: Very little work has been done on specifying a standard mapping between the overlapping semantic specifications in the standardized data models used in architecture, engineering and construction (A/E/C, e.g., IAI-IFC and ISO-STEP standards). However, several companies have developed bespoke mappings from these standards into their design tools, and back out again. With this approach it is difficult to understand how complete their mappings are, and what assumptions are made in the development of the mappings. Yet for semantic mappings, as distinct from mappings over geometric representations, this has a profound implication for the correctness of the resultant data. In this paper the development of a suite of mapping support tools is discussed to illustrate the level of support required to ensure semantically correct mappings across data models.

1 INTRODUCTION

1.1 *Problem statement*

The specification of a semantically correct mapping between any two standard data models used in the A/E/C industries is an enormous task. Data models have in the order of 500 entities and many thousands of relationships and attributes (e.g., IFC 2.x, IAI 2004). The mere task of sitting down and describing which entities are related to each other is daunting, let alone managing to encompass the full semantic coverage of the contents of each of these entities. Yet without some definition of a mapping to be implemented it is basically impossible to guarantee the correctness of any implemented translator for a standard data model.

It is clear that human experts are needed to perform this task, knowledgeable in both schemas being mapped between. Yet even for such experts the management problem of describing a mapping over such large schema forces a requirement for some computerized support. This support comes in the form of notations and environments to specify what is equivalent between two schema in a form that can then be used to generate the code to actually perform the mapping.

In the last decade there was an active research community developing approaches to mapping languages in engineering domains (Khedro et al 1996; Verhoef et al 1995; Eastman 1999: Chapter 11). Several of those efforts have been pursued in the development of the ISO mapping standard EXPRESS-X (Hardwick and Denno 2000), and in the development of mapping tables (ISO 1993).

These mapping approaches are now being utilised on a wide range of standard data models available from ISO 10303 STEP, ISO 13584 Parts libraries and catalogs, CIS/2 (Crowley and Watson, 2000) and IAI's IFCs (IAI 2002). However, every mapping between two of these standard data models will be duplicating the work of previous attempts. If it were possible to specify a mapping in an easily comprehensible manner, and there were tools that industry experts could use to agree on the correctness of the defined mapping, then a consensus on major mappings between schema could be developed and published in much the same way that standard schema are published today. This paper examines what tools would be required to reach this position.

2 A FRAMEWORK OF TOOLS

To manage the task of developing a mapping between two data models there is a requirement for a range of support functions for the specifier. These include:

- A graphical mapping notation to enable the specifier to visually comprehend the mapping being described between subsets of the data models.
- A mapping specification environment to enable navigation through, and partitioning of,

the space of mappings specified. Such a tool can also determine what has, or has not, been mapped between.

- Automated mapping support to enable a significant proportion of the mappings required between two schemas to be automatically determined.
- A mapping interpreter to allow evolving mappings to be tested on partial sets of data.
- A verifier to check the correctness of the developing mapping specification. Such a verifier would offer support from basic syntactic checking across the data models through to a more comprehensive semantic analysis of the proposed mapping.

The development of such a support environment is described in the following sections. With this environment in place it is then possible to move on to providing standard mappings between the major standard schemas which exist in our domain.

3 MAPPING NOTATIONS

In order to describe the equivalences which exist between data structures in two different schema it is necessary to have a notation for the specification. A range of notations have been developed and utilised ranging from straight specification within a standard programming language (such as C or Java), through ISO mapping tables (ISO 1993), and the evolving ISO mapping language EXPRESS-X (Hardwick and Denno 2000).

In many respects these approaches are analogous to the use of the EXPRESS language to specify the conceptual data structures for a schema for a particular domain. These approaches provide for a complete and detailed specification of how the mapping between portions of the schemas will have to be realised.

However, they do not provide a way to gain an overview of the mappings which have been developed between two schema or the completeness of any particular mapping. Where schema have several hundred classes in them this is of major concern to the specifier. In the same way that EXPRESS-G is used as a high-level notation for describing the basic structures within a schema, and to view various subsets of a schema, a graphical mapping formalism will allow a high-level overview of the mapping between schemas to be presented.

A range of graphical formalisms have been developed at the University of Auckland to represent mappings to different classes of users. Figure 1 shows a programmer level formalism for specifying mappings between UML styled class diagrams in two schema.

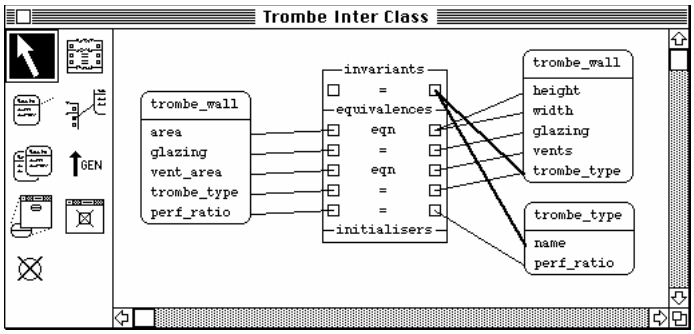

Figure 1. VML-G: a graphical mapping formalism.

The VML-G language (Amor 1997) shown in Figure 1 uses a wiring approach to denote a mapping between attributes, or classes, in a schema and an icon representing that particular mapping. The mapping icon provides three areas in order to separate general mappings between attributes and classes from the specification of invariants, which direct when the mapping is applicable, and initialisers, which describe starting values for particular attributes of a newly created object. As can be seen in Figure 1 the specification of the actual mapping is hidden from view and presented as a classification to either a straight equivalence (=), an equation (eqn), a functional equivalence (func), or a procedurally described equivalence (proc).

By examining such a graphical mapping specification it is very easy to verify that all attributes are being handled in the mapping, and by examining the invariants across several mappings it is possible to verify that all possible conditions are being modelled. It also allows a high-level specification of the equivalences between portions of a schema without concentrating on the detail of how to achieve the mapping.

The author contends that any textual mapping notation needs to be supported by a graphical formalism which allows for a high-level overview of the mappings which are being specified.

4 MAPPING SPECIFICATION ENVIRONMENT

If a simple textual notation is used to describe a mapping then it can be developed in any textual editor. However, if a graphical formalism is going to be utilised to specify a mapping then it needs to be supported by a more comprehensive specification environment. Such a specification environment must allow for both graphical and textual notations to be viewed and the consistency between these views to be maintained under edits to either view.

In Figure 1 the specification environment for VML-G allows for classes from the related schema to be viewed within a window, for a mapping icon to be placed in the window, and for wiring from attributes and classes to be drawn to the mapping icon. In this environment each window represents a particu-

lar mapping, and by navigating the various windows a specifier can examine the full set of mappings developed. The specifier can also switch to a textual view to see the full mapping specification and any edits made to the textual view are propagated back into the graphical view. From a programmer level support perspective this is very useful, but it does not tie to real data to help checking.

In Figure 2 a business level specification environment is shown (Li et al 2002). Within this environment the schemas being mapped between are visualized as business forms and a wiring approach is used to specify the mappings between various fields in the forms. In this environment the specifier can view not just the data schema in a format close to its business use, but also exemplar data within each of the fields. Tied to this is the ability to run each of the partial mappings specified and hence to view the result of the application of the specified mappings in the other business form.

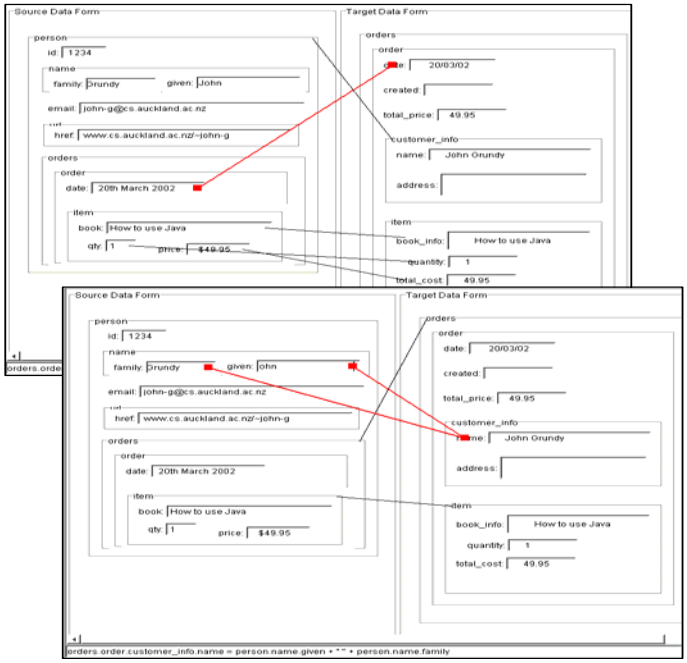

Figure 2. A business level mapping specification environment.

5 AUTOMATED MAPPING CREATION

While the tools highlighted in Figures 1 and 2 clearly provide for greater comprehension and checking of the mappings which are being described it is also clear that detailing the mappings between schema which comprise several hundred classes is going to take a long time.

In order to ease this workload it is useful to consider approaches which will allow for the automated specification of the mapping (or a portion of the mapping) between two schema. This is an area of ongoing research with many approaches being considered (see Rahm and Bernstein 2001 for a survey of approaches).

This sort of tool is also useful to handle mapping between versions of particular product models. For example, the IAI have produced six versions of the IFC in the last seven years and the CIS/2 LPM is expected to be updated every year. The mapping between consecutive versions of a particular schema tend to be fairly minor which make for an easier problem when considering automated mapping creation. This problem is also closely related to that of schema evolution in object-oriented databases (Banerjee et al 1987, Lerner and Habermann 1990, Eastman 1992, Deux 1990, Zicari 1992, and Atkinson et al 2000).

A previous student developed a hybrid mapper utilizing structure and name comparison to automate the creation of mappings for IFC versions (Amor and Ge 2002). This demonstrated that approximately 80% of an IFC schema could be automatically mapped to the next version.

An examination of points where this hybrid mapper failed illustrated that different approaches to identifying mappings performed well in different settings. To explore how this might be utilized in automated mapping creation there has been a project (Bossung 2003) to develop an infrastructure to allow multiple matchers to vote on their proffered mapping for particular parts of an inter-schema mapping. Figure 3 shows a screen snapshot of this tool where three matching tools (a Levenshtein matcher, a partial name matcher, and a type matcher) bid for their mapping for a partial structure match. With this tool the user can examine the highest ranked mappings for any portion of the schema and select between the mappings being offered. Further work on this tool is looking at re-matching mappings based on selections (changes) made by the user of the tool.

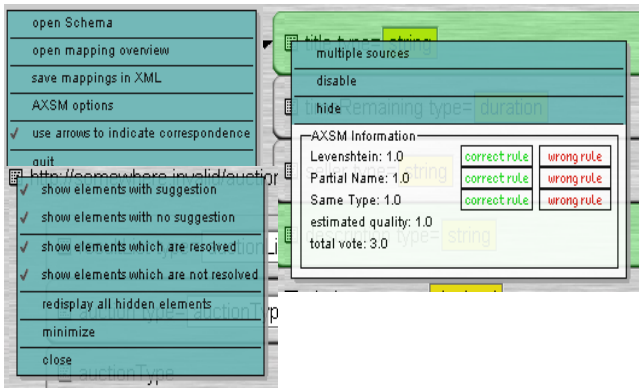

Figure 3. Voting in an automated mapping tool.

6 MAPPING INTERPRETER AND CODE GENERATOR

In order that the specified mappings can be enacted it is necessary to generate code for each mapping. Within a tool which supports visualization of mapped exemplar data this has to take the form of an interpreter for individual snippets of the mapping. Every mapping tool must be able to generate the full mapping specification in some target language.

With a high-level mapping specification language, as demonstrated in Figures 1 and 2, the mapping code generator allows for the creation of code in a number of target languages from bespoke languages such as VML (Amor 1997) and EXPRESS-X (Hardwick and Denno 2000) through to generic mapping languages such as XSLT and even straight into programming languages such as C and Java.

7 MAPPING VERIFIER

As detailed in the introduction, developing a high-level mapping provides the specifier with a way to ensure that the semantics of the data in two schemas is going to match. However, due to the size of the schemas being developed this is still a difficult process. The provision of a graphical formalism helps in checking, as do support environments which map exemplar data based on the developing mapping. But, to ensure a correct mapping has been developed requires a comprehensive testing regime based around non-trivial exemplars.

While the IAI and ISO do have a certification process and testing suites the approach is certainly not as rigorous as would be expected in a field such as software testing.

The author suggests that the testing of a round-trip mapping should be considered as the main form of verification for mapping specifications. While implemented translators for geometric models (e.g., DXF, IGES, etc) are known, and assumed, to have errors this is not such an issue as human interpretation is used to determine the semantics of a translated geometric model. However, for an object-based model such errors are far more serious. A round trip mapping which does not preserve individual objects and their original parameters has changed the specification of the building to almost any tool which uses this data.

8 CONCLUSIONS

The development of mappings between two schema is a large and very important process when developing translators for the various standard schemas being used in the construction industries. To ensure correct specifications requires not just an expert in the various schema being manipulated, but also a range of support tools to help the specifier through the process. A range of these support tools have been developed and described briefly in this paper.

However, to take the industry to the next stage where they can have confidence in the translators and mappings which exist requires that further rigor is injected into the mapping testing process. It is also recommended that a range of certified mappings be developed between the main "standard" schemas being developed for the industry as well as between the various versions of the schema which have been produced.

REFERENCES

Amor, R.W. & Ge, C.W. 2002. Mapping IFC Versions, *Proceedings of the EC-PPM Conference on eWork and eBusiness in AEC, Portoroz, Slovenia, 9-11 September*, 373-377.

Amor, R. 1997. A Generalised Framework for the Design and Construction of Integrated Design Systems, *PhD thesis, Department of Computer Science, University of Auckland, Auckland, New Zealand*, 350 pp.

Atkinson, M.P., Dmitriev, M., Hamilton, C. & Printezis, T., 2000. Scalable and Recoverable Implementation of Object Evolution for the PJama$_1$ Platform. *Persistent Object Systems, 9th International Workshop, POS-9, Lillehammer, Norway, 6-8 September*, 292-314.

Banerjee, J., Kim, W., Kim, H., & Korth, H. 1987. Semantics and Implementation of Schema Evolution in Object-Oriented Databases, *Proceedings of the 1987 ACM SIGMOD international conference on Management of data. San Francisco, USA*, 311-322.

Bossung, S. 2003. Semi-automatic discovery of mapping rules to match XML Schemas, *Department of Computer Science, The University of Auckland, New Zealand*, 71 pp.

Crowley, A., & Watson, A. 2000. CIMsteel Integration Standards, *Release Two, 5 Volumes, Steel Construction Institute and Leeds University, UK*.

Deux, O. 1990. The story of O2. *IEEE Transactions on Knowledge and Data Engineering (TKDE), 2(1), March*, 91-108.

Eastman, C.M. 1999. Building Product Models, *CRC Press, Orlando, FL, USA*.

Eastman, C.M. 1992. A data model analysis of modularity and extensibility in building databases, *Building and Environment, 27(2),* 135-148.

Hardwick, M. & Denno, P. 2000. The EXPRESS-X Language Reference Manual, *ISO TC184/SC4/WG11 N117, 2000-06-28*.

IAI. 2002. International Alliance for Interoperability, web site last accessed 18/6/2004, http://www.iai-international.org/.

ISO 1993. Guidelines for the documentation of mapping tables, *ISO TC184/SC4/WG4 M105, 1993-09-10*.

Khedro, T., Eastman, C., Junge, R., & Liebich, T. 1996. Translation Methods for Integrated Building Engineering, *ASCE Conference on Computing, Anaheim, CA, June*.

Lerner, B.S. & Habermann, A.N. 1990. Beyond schema evolution to database reorganization, *Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA), Ottawa, Canada, October*, 67-76.

Li, Y., Grundy, J.C., Amor, R. & Hosking, J.G. 2002. A data mapping specification environment using a concrete business form-based metaphor, *In Proceedings of the 2002 International Conference on Human-Centric Computing, IEEE CS Press,* 158-167.

Rahm E. & Bernstein, P.A. 2001. A survey of approaches to automatic schema matching, *The International Journal on Very Large Data Bases (VLDB), 10(4),* 334-350.

Verhoef, M., Liebich, T. & Amor, R. 1995. A Multi-Paradigm Mapping Method Survey, *CIB W78 - TG10 Workshop on Modeling of Buildings through their Life-cycle, Stanford University, California, USA, 21-23 August*, 233-247.

Zicari, R. 1992. A Framework for schema updates in an object-oriented database systems. *Morgan Kaufmann Series In Data Management Systems*, 146-182