mapping IFC versions

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

ABSTRACT: In order to cope with the growing number of versions of IFC schema being utilized by design tools in architecture, engineering and construction domains it is necessary to generate mappings between the different versions. This paper describes a system which can interrogate two schema versions in the same domain and automatically generate a mapping specification between them. The system supports refinement by a domain expert, to complete mappings which could not be automatically determined, or to correct those which were misidentified by the system. The categorization of mappings which are identified by the system are described in the paper along with statistics for the system's use on major IFC versions.

1 INTRODUCTION

There are a growing number of versions of IFC (Industry Foundation Classes) data with the annual release of IFC models by the IAI (International Alliance for Interoperability). Currently, there are three major IFC versions (IAI 2002) which are being utilized in software products (1.5.1, 2.0, and 2.X). With each progression of IFCs we can expect more versions to be in use, and a growing number of software products supporting different versions. Given the current size of the IFC model (around 600 distinct classes and types), and the fact that it will grow over time, it is clear that defining a mapping between one version and the next is a major undertaking. However, it is imperative that this is accomplished as data will need to be moved between the different versions supported by the different design tools, and data stored in older IFC versions should be able to be migrated to the newest version.

The fact that a new IFC version builds upon the previous version reduces the complexity of the mapping problem significantly. For example, approximately 15~25% of classes are equivalent between IFC versions. It is also clear that the differences between many of the other classes are fairly minor. This leads us to believe that it will be possible to automatically examine two versions of the IFC model and automatically produce a mapping which can handle the majority of the data structures in the two versions.

This paper examines the approach taken to understand the complexity of mapping between IFC versions and the progress made to produce mappings (currently both EXPRESS-X and XSLT) for any two versions of the IFC. Initial work has categorized the potential differences that can be found between two schema. This has led to a system which can examine two schema and identify how many of each of the mapping categories exist between the two schema. The result of this analysis is fed into a generic mapping generator which is capable of producing mapping descriptions for any of the mapping language models incorporated into it (currently just EXPRESS-X and XSLT).

The paper details the analysis of mapping categories between schema versions and shows statistics of the correspondences between different versions of the IFC models. The paper also details the system created to analyze any two schema (in a variety of notations, e.g., EXPRESS or XML-DTD) for known mapping categories. The last section in the paper describes ongoing work in generating the required mapping specification in order to map data between versions.

2 REQUIREMENTS FOR VERSION MAPPING

With the release of each new IFC version the total number of ENTITYs and TYPEs increases by over one hundred (see Table 1). The total number of attributes in a schema and the relationships between classes grows significantly between each version as well. Currently, the IAI do not publish a specification of the changes between versions in a form which can be used to define a mapping.

Table 1. Number of ENTITY and TYPE declarations in different IFC versions

IFC Version	ENTITYs	TYPEs
1.5.1	186	95
2.0	290	157
2.X	370	228

It is also clear that not all design tool developers will be able to devote the development effort to update their IFC interface to the newest versions when they come out. Currently, a large number of design tools still utilize IFC 1.5.1 interfaces (see Steinmann 2002 for the interfaces utilized by different design tools).

If design tool developers need to provide interfaces for the different versions of IFCs, or a translator between the different versions, then this is clearly another major development cost for their organization.

In this project we aim to reduce the effort required to map between versions by automatically generating as much of the mapping between two IFC versions as possible. The main requirements we are working towards are as follows:

- Being able to generate a skeleton mapping between any two versions of related schema. This aims to reduce the amount of human time required to specify mappings between two schema versions.
- Finding a generic representation of schemas to ensure the mapping tool can work with schema versions described in any formalism. For example, schemas defined in EXPRESS, XML-DTD, XML Schema, SQL, etc should all be able to be manipulated by the system.
- Providing the user with a tool to augment, and complete, the automatically defined mappings. This tool must enable the user to cope with the potentially large schemas being developed in technical domains.
- Being able to generate mapping code in a range of mapping languages (e.g. EXPRESS-X, XSLT, or SQL) or even to generate straight programming language code for a mapping (e.g. C++ or Java). This will allow developers to select mapping code compatible with their development environment.
- Providing a tool which can map a data model between two different versions of a schema. For example, taking a SPF (STEP Physical File) describing a building in IFC 1.5.1 and generating the equivalent SPF for IFC 2.X.

3 TYPES OF MAPPINGS

The initial phase of this project concentrated on examining three versions of the IFC (1.5.1, 2.0 and 2.X) to determine the main categories of mapping that would be required to specify translations between the versions. This analysis identified six main categories of mappings which could be automatically identified for both entities and types in EXPRESS. These are:

- IDENTICAL: This describes an entity, or type, from a source schema which is totally identical to an entity, or type, in the target schema. Being identical in this form requires all attribute names and types to be identical, as well as INVERSE attributes and WHERE clauses. It also requires that all SUPERTYPEs are identical as well.
- RENAMED: This describes an entity, or type, from a source schema which is totally identical to an entity, or type, in the target schema as described above except that the name of the ENTITY, or TYPE, has been changed.
- EQUIVALENT: This describes an entity, or type, from a source schema whose content is nearly identical to an entity, or type, in the target schema except that the name of the ENTITY, or TYPE, has been changed.
- MODIFIED: This describes all other entities and types in a schema which are not identical in any of the forms described above, but where a matching entity or type has been identified (e.g. because it has the same name).
- ADDED: This describes an entity, or type, which appears in the target schema but was not in the source schema.
- REMOVED: This describes an entity, or type, in the source schema which does not appear in the target schema.

The mapping categoriser built as part of this system will take any two schema and attempt to identify what the correspondences are between their entities and types. The outcome of this categorization between IFC 1.5.1 and IFC 2.0 is shown in Table 2.

Table 2. Identified relationships between IFC 1.5.1 and 2.0

Relationship	ENTITYs	TYPEs
IDENTICAL	24	43
RENAMED	0	1
EQUIVALENT	7	9
MODIFIED	146	19
ADDED	113	85
REMOVED	9	23

This is not a perfect categorization of the differences between the two schema as a hand mapping categorizes the entities and types in a slightly different manner. However, this provides an initial mapping between two schema which is relatively complete.

After this initial categorization of mappings the system re-examines all top level mapping relationships as there is a range of more detailed categories which can be recognized, either over the whole entity and type or for individual attributes within an entity. For entities and types these include:

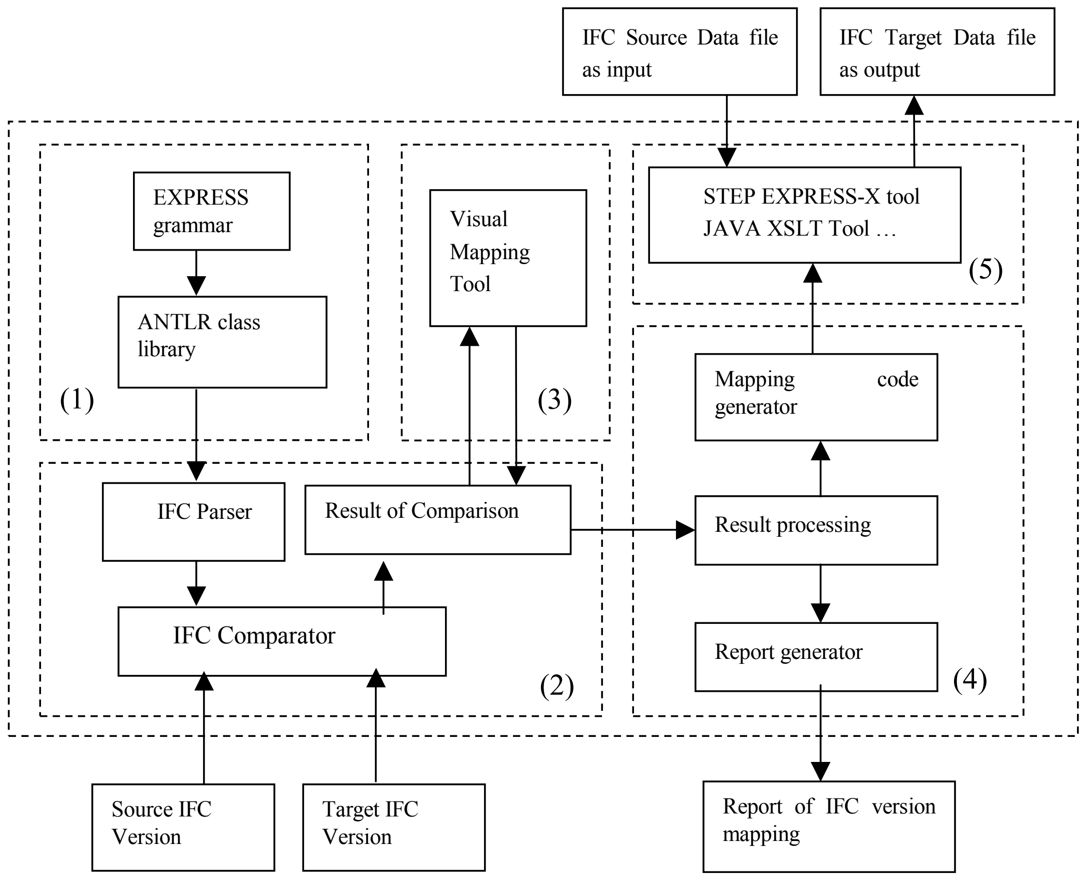


Figure 1. Structure of the IFC version mapping system

– An entity, or type, in the source schema being split into two (or more) entities, or types, in the target schema.
– Two (or more) entities, or types, in the source schema being merged into one entity, or type, in the target schema.
– An entity in the source schema maps to a type in the target schema.
– A type in the source schema maps to an entity in the target schema.

Attributes are classified in a similar manner to entities and types (i.e. identical, renamed, modified, added, removed) along with the following classifications:

– A change in the optional status of an attribute. Either from optional to non-optional or vice versa.
– An attribute in an entity being moved to one of its SUPERTYPE entities, or vice-versa.

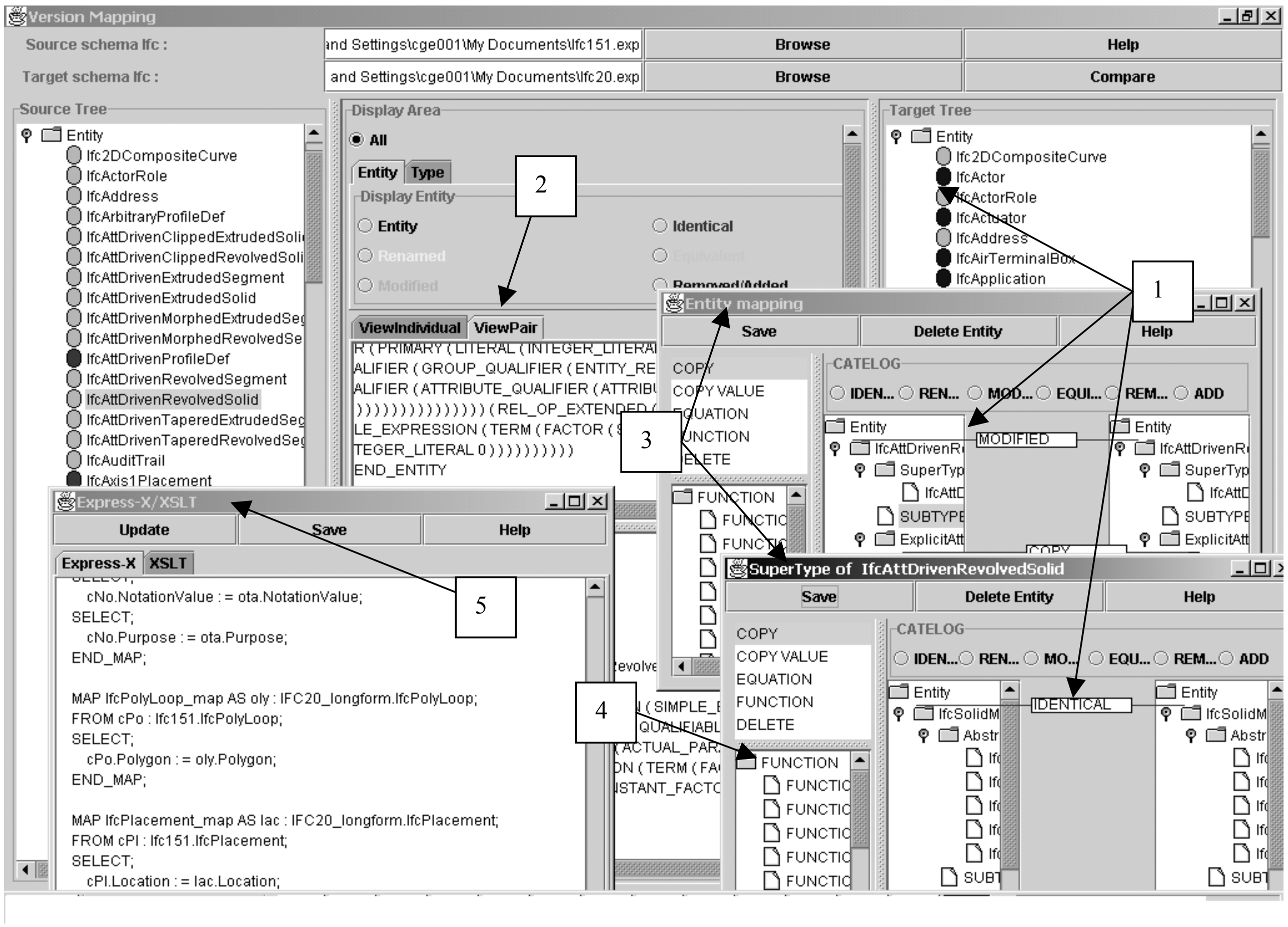
This classification is applied to all types of attributes in an EXPRESS schema, including those which are of a derived and inverse type. The classification is also applied to WHERE rules within an entity.

A full description of all of these categorizations can be found in Ge (in prep.).

4 VERSION MAPPING SYSTEM FRAMEWORK

A system to generate and support mappings between versions of a schema requires several fairly complex components. Figure 1 shows the structure of the system developed in this project. The five main subsystems in this implementation are:

1 A parser generator to support the syntax of the schema specification.
2 A comparator to examine the parsed representation of two schemas and identify similarities as described in section 3.
3 A graphical user interface to allow a mapping expert to modify and add mapping specifications on top of those identified by the comparator.
4 A mapping code generator which understands the mapping specification generated in 2) and augmented by the user in 3).
5 The data mapping subsystem which is generated from the mapping code in subsystem 4).



```
R ( PRIMARY ( LITERAL ( INTEGER_LITERAL
ALIFIER ( GROUP_QUALIFIER ( ENTITY_RE
ALIFIER ( ATTRIBUTE_QUALIFIER ( ATTRIBU
)))))))))))))))) ( REL_OP_EXTENDED (
LE_EXPRESSION ( TERM ( FACTOR ( S
TEGER_LITERAL 0 ) ) ) ) ) ) ) ) ) ) )
END_ENTITY
```

```
    cNo.NotationValue : = ota.NotationValue;
  SELECT;
    cNo.Purpose : = ota.Purpose;
  END_MAP;

  MAP IfcPolyLoop_map AS oly : IFC20_longform.IfcPolyLoop;
  FROM cPo : Ifc151.IfcPolyLoop;
  SELECT;
    cPo.Polygon : = oly.Polygon;
  END_MAP;

  MAP IfcPlacement_map AS lac : IFC20_longform.IfcPlacement;
  FROM cPl : Ifc151.IfcPlacement;
  SELECT;
    cPl.Location : = lac.Location;
```

Figure 2. User interface for managing mappings

4.1 *The parser generator*

A schema is described by a particular notation, usually textual in nature. In order to generate mappings between two versions of a schema it is necessary to generate a representation of the schemas which can be traversed and compared to each other. In this project the ANTLR tool (ANTLR 2002), which is a generic parser generator, was used to generate a parser for the EXPRESS language and for the XML-DTD specification. Parsers for other notations could be added at a future date, but these two sufficed for the proof of concept for this system.

4.2 *The schema comparator*

This subsystem loads in the two versions of a schema and builds an object-based representation of each schema in a format which is neutral from the syntax of the original schema. In this way the comparator is independent of the original schema specification language. The categories described in section 3 are used to define correspondences between the two schema. Each mapping between entities, or types, in a schema is represented as a separate mapping object holding a range of information on the relationship between the entities, or types. The comparator also creates separate mapping objects to capture the relationship between individual attributes within a mapped entity. Initial tests of the comparator on major IFC versions shows that approximately 65% of the mappings required between any two versions can be generated automatically.

4.3 *The graphical user interface*

The user interface (described in section 5) allows an expert to view the generated mappings and to redefine or fix the mappings which were generated. The two schema versions are color coded by their mapping type and both schemas can be navigated to identify where they have been mapped to.

4.4 *The mapping code generator*

The mapping code generator takes the set of mapping relationship descriptions and generates a target mapping specification. As the mapping relationship is schema language independent the generator can be adapted to produce mapping code for a range of target languages. Currently, the EXPRESS-X (2002) and XSLT (2002) languages are supported as they are the most commonly used mapping specification languages in this domain. This subsystem can also produce summary statistics about the mapping between the schema versions in the form of a report to the user.

4.5 *The data mapping subsystems*

The generated mappings are linked into existing mapping support engines. Target mapping systems in this project were the Java (2002) XSLT tool and the EXPRESSO (2002) EXPRESS-X tool. This enables the generated mappings to be tested with data models in the most common formats utilized by design tools in the area (e.g. SPF or XML documents).

5 VERSION MAPPING INTERFACE

The user interface provides a visual mapping environment for an expert in the domain to navigate the generated mappings and to complete the specification, or fix any errors created by the automated mapping generation process. Figure 2 provides a snapshot of the major components within the user interface. These include:

- The main navigation system (large window at the back of all others). This allows the user to view all entities, types, and functions in source and target schemas (see label 2 for an example of the full source view of an entity). The user can also choose to see matching entities, or types, if they choose ‘View Pair’ in this window.
- Individual entities and types are color coded to indicate the mapping categorization they hold (e.g. IDENTICAL, RENAMED, etc). The entities and types displayed can be restricted to just those categorized to a particular type (e.g. ADDED), to reduce the number of entities or types in view at any one time. The mapping between individual entities, or types, can be opened in a separate window where a wiring approach is utilized to show the type of correspondence between components in the entity or type. Label A shows examples of this representation of the mappings.
- An ENTITY's SUPERTYPE can be visualized to determine the mappings utilized in the higher level entity description (see label 3).
- A library of functions can be built up for common mapping problems. These can be called upon when a new mapping is put together, to complement the more normal specification of equivalences and equations between attributes, types and entities (see label 4).
- The mapping specification is automatically regenerated whenever a change is made to the stored mappings. The mapping specification can be inspected at any time, and hand edited, if required, prior to its use (see label 5).

6 CONCLUSIONS AND FUTURE WORK

This paper describes an approach to defining mappings between versions of a schema based upon recognizing a classification of the relationships between entities and types in the relative schemas. To demonstrate this approach a generic mapping system was developed capable of handling schema specifications in a range of formalisms and of generating a mapping in a range of target mapping languages. Recognizing that it is not possible to automatically determine all mappings between versions the system supports user specification and modification of the generated mappings.

The developed system has been tested with versions of the IFC schema. Initial indications are that about 65% of the required mappings between two versions of IFC schema can be generated by the system. The final 35% of mappings and checking of the automatically generated mappings is still a human task, though supported by a GUI which provides visual feedback on what is required to be mapped.

The authors believe that a greater variety of mapping types can be recognized by the system and aim to determine further generic categories of mappings for automated recognition.

The authors are also interested in further extensions of the GUI to support the traversal and specification of mappings by a user. Defining a visual notation to quickly specify a mapping between entities, types, and attributes is a challenging task for schemas of the size being released by the IAI.

REFERENCES

ANTLR. 2002. *The ANTLR Translator generator*, web site last accessed 5/5/2002, http://www.antlr.org/.

EXPRESS-X. 2002. *EXPRESS-X Reference Manual*, ISO 10303, WG11, N088, web site last accessed 5/5/2002, http://www.steptools.com/library/express-x/n088.pdf.

EXPRESSO. 2002. *Express Engine*, web site last accessed 5/5/2002, http://sourceforge.net/projects/exp-engine/.

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

IAI. 2002. *International Alliance for Interoperability*, web site last accessed 5/5/2002, http://www.iai-international.org/.

Java. 2002. *Java Technology and XML*, web site last accessed 5/5/2002, http://java.sun.com/xml/.

Steinmann, R. 2002. *International Overview of IFC-Implementation Activities*, web site last accessed 5/5/2002, http://www.iai.fhm.edu/ImplementationOverview.htm.

XSLT. 2002. *The Extensible Stylesheet Language (XSL)*, web site last accessed 5/5/2002, http://www.w3.org/Style/XSL/.