XML Representation of EXPRESS Models and Data

Edward J. Barkmeyer and Joshua Lubell
Manufacturing Systems Integration Division
National Institute of Standards and Technology
100 Bureau Drive, Stop 8260
Gaithersburg, MD 20899-8260 USA
+1 301 975 3508
{edbark, lubell}@nist.gov

ABSTRACT
EXPRESS is a modeling language for use in engineering data exchange standards that combines the entity-attribute-relationship and object modeling paradigms. This paper discusses some issues we encountered when attempting to represent EXPRESS models and data sets as XML (Extensible Markup Language). Our experience should be applicable to other projects concerned with providing XML-based exchanges of information modeled in relational or object-oriented languages.

Keywords
Data model, EXPRESS, mapping, representation, schema, XML.

1 BACKGROUND

EXPRESS [1] is a data modeling language that combines ideas from the entity-attribute-relationship family of modeling languages with object modeling ideas of the late 1980s. It became an international standard (ISO 10303-11) in 1994 for use in engineering data exchange.

The primary EXPRESS concept is the *entity type*, which models a domain of conceptual or real-world objects and the collection of information units that describe them. An entity type has *attributes* that model the descriptive information units, and each attribute has a *data type*, which specifies the nature and values of the information unit. Data types can be the common computational types (Boolean, integer, real, string, enumeration), or entity types, or aggregates (set, list, array) of any of these. EXPRESS also supports *defined types*, which are new data types defined by the modeler to be represented by values of any of the other data types.

As in object models, an EXPRESS entity instance is considered to have an identity distinct from its modeled attributes and properties. That is, EXPRESS does not consider any attribute value or the set of attribute values to denote the entity instance. An entity instance is considered to be an *object*, which is partly represented by the modeled attributes, and has an unmodeled unique identifier.

EXPRESS has no explicit *relationship* construct. Relationships are modeled as attributes whose data type is an entity type or an aggregate of an entity type. Some relationships are reified as entity types with role attributes whose values are the participating entities.

There are important characteristics of EXPRESS that we do not discuss in this paper. EXPRESS has an elaborate constraint language in which limitations on the values of attributes and the populations of entity types, and relationships among entity instances and values of attributes can be specified. But this does not affect XML mapping. For the sake of brevity, the EXPRESS mechanisms for defining complex inheritance relationships between entity types are not discussed, even though they affect XML mapping.

A project is underway in ISO Technical Committee 184 (Industrial Automation) to develop standards for representing EXPRESS models and data as XML [2]. Some compelling reasons for doing this are [3]:

- Unlike EXPRESS, XML is widely used and XML processing software is widely available.
- XML defines a well-tested, standard syntax for representing structured data.
- The popularity of XML may help to make established EXPRESS data models more accessible to new user communities.

Early Bindings and Late Bindings
XML representations of data modeled in EXPRESS may employ either an *early binding* or a *late binding*. In an early binding, the named components of the XML vocabulary correspond directly to data types and attributes defined in the EXPRESS model. For example, consider the following EXPRESS definition of a point on a plane with x and y axes:

```
ENTITY point;
  x, y : REAL;
END_ENTITY;
```

This EXPRESS definition specifies a point as having two attributes whose values are real numbers corresponding to the `x` and `y` coordinates of the point. An early-bound XML representation of a point might look something like

```
<point id="e1">
  <x>3.1</x>
  <y>5.7</y>
</point>
```

with `id` representing the unmodeled unique identifier.

In a late binding, the named components of the XML vocabulary do not directly correspond to EXPRESS data types. Instead they correspond to EXPRESS metadata objects — entity, attribute, data types. For example, a late-bound XML representation of a point could look like

```
<entity id="e1" name="point">
  <attribute name="x">
    <real_value>3.1</real_value>
  </attribute>
  <attribute name="y">
    <real_value>5.7</real_value>
  </attribute>
</entity>
```

Although late bindings are more verbose than early bindings, a late-bound EXPRESS-to-XML mapping is better suited to XML applications involving multiple EXPRESS information models. If an early-bound strategy is used for such applications, there must be a distinct XML tag set for each EXPRESS model. That is, one needs a separate XML namespace for each EXPRESS model. This complicates implementation. A late binding, on the other hand, allows for a single tag set to be used for all EXPRESS models, since the XML vocabulary defined by the tag set corresponds to EXPRESS metadata objects rather than to objects defined in the model.

Early bindings, on the other hand, are most useful for XML applications implementing a single EXPRESS model. They are less verbose, more human-readable, and simpler to process than late bindings, and they are also better equipped to make use of XML tooling. For the purposes of this discussion, we focus mainly on early bindings since early-bound EXPRESS-to-XML mappings seem to be preferred in the EXPRESS user community.

2 ISSUES

The following are some issues that arise when attempting to reformulate EXPRESS models as XML.

Name Mapping

In EXPRESS, the name of a data type or attribute is a sequence of letters, digits and underscore ("low line") characters, beginning with a letter. This means that any EXPRESS name is a valid XML name. Fortunately, the hyphen character is not permitted in EXPRESS names, but it is permitted in XML names. By using the hyphen, we can create binding-specific XML names that are guaranteed not to conflict with names defined in the EXPRESS model.

Name Scoping

The name of an EXPRESS entity type is required to be unique across the model, but the name of an EXPRESS attribute is only required to be unique over the entity type and the types from which it inherits. Therefore, EXPRESS attributes with the same name and different data types can exist in the same model. That is, attributes of two different EXPRESS entity types can have the same name and have values with entirely different representations. If the EXPRESS attribute is mapped directly to an XML element and the document is to have a Document Type Definition (DTD), the content model of that element must accommodate all of the possible data type representations. But that content model doesn't properly constrain any instance of that element, and validation under such a DTD is not very meaningful. For example, if the entity

```
ENTITY axis_labels;
  x: LIST OF STRING;
  y: LIST OF STRING;
END_ENTITY;
```

appears in the same model as `point` above, the declaration for element `x` in the DTD would have to be:

```
<!ELEMENT x (#PCDATA | string-value*)>
```

If the XML document is to have a DTD, therefore, EXPRESS attribute names cannot be mapped directly to element tags. The ISO project has actually explored three approaches to solving this problem:

Mapping EXPRESS Attributes to XML Attributes

Under this approach, the EXPRESS attributes are mapped to XML attributes of the element corresponding to the entity type. Our XML representation would appear as:

```
<point x-id="e1" x="3.1" y="5.7"/>
```

This approach ensures that EXPRESS attribute names are unambiguously specified in XML, but it also requires that metadata XML attributes be differentiated from their EXPRESS siblings. Hence, the point's XML identification attribute is named `x-id` rather than `id`. A more serious problem with this approach is that XML attributes cannot support all the possible representations of EXPRESS attribute values. This is discussed below in the section on aggregate types.

Mapping EXPRESS Attributes to XML Elements

Under this approach, the XML element tag has the form `entity-name.attribute-name`. And our data would appear as:

```
<point id="e1">
  <point.x>3.1</point.x>
  <point.y>5.7</point.y>
</point>
```

Although XML attribute names are guaranteed to be unique, this approach results in very cumbersome element names and complicates XML processing.

Discarding DTDs Altogether
In this approach, EXPRESS attributes are mapped to elements whose tags are the EXPRESS attribute names, as in our original early-bound definition for point. This results in an XML mapping of the EXPRESS model that, in the general case, cannot usefully be specified as a DTD. The XML mapping can, however, be specified using a non-DTD XML schema language that permits context-sensitive element types, such as RELAX [4], TREX [5], or XML Schema [6]. Although this approach combines the advantages of the first two approaches, it requires the use of XML validation methods that are not as time-tested and well understood as DTDs.

The authors believe that the last is the best choice — that DTDs are of little value in validating data that is to conform to an object model or relational model.

Hierarchies
EXPRESS models have no intrinsic hierarchical structure. Nothing in an EXPRESS model identifies an entity instance as part of, or belonging to, a higher-level entity. One might assume that an entity-valued EXPRESS attribute would model a subsidiary relationship, except that that same syntax (entity with attribute) must also be used to model many-to-many relationships and reified relationships. And since EXPRESS does not distinguish among entity types that model independent objects, entity types that model relationships, and entity types that model simple information units that comprise multiple data elements (such as an address), there are no clues in the EXPRESS model to suggest which entity types can be mapped to hierarchical constructs in XML. An EXPRESS model is a network of largely independent entity types connected by relationships somehow modeled using attributes whose values are entity instances.

Accordingly, an EXPRESS data set is mapped into a sequence of XML elements representing EXPRESS entity instances. Each entity element has an XML ID attribute, providing the unmodeled unique identifier for the entity instance (described above). Assuming EXPRESS attributes are mapped to XML elements and not XML attributes, the element representing the entity contains elements representing its EXPRESS attributes. In addition, every entity-type also maps to a *reference element* — an XML element type with an XML attribute of type IDREF and empty content. An occurrence of this element represents a reference to the entity instance with the given ID value. For example, an instance of a triangle EXPRESS entity type modeled as

```
ENTITY triangle;
  p1, p2, p3 : point;
END_ENTITY;
```

in a data set containing point instances with unique identifiers `e1`, `e2`, and `e3` might look like

```
<triangle id="triangle1">
  <p1><triangle-ref ref="e1"/></p1>
  <p2><triangle-ref ref="e2"/></p2>
  <p3><triangle-ref ref="e3"/></p3>
</triangle>
```

or, with EXPRESS attributes mapped to XML attributes, might look like

```
<triangle x-id="triangle1" p1="e1"
          p2="e2" p3="e3"/>
```

In short, the XML document is a collection of “flat” elements representing independent entity instances linked together by IDREFs, because no hierarchical structure can be deduced from the EXPRESS model.

To make better use of XML's inherent hierarchical structure, the project is developing representation methods allowing instances to be contained within other instances [7]. This entails some formal annotation of EXPRESS models to cue the mapping to useful hierarchical data structures in XML. Some uses of these annotations might belong to the model itself, while others relate to particular uses of the model. In any case, this means that we must enhance the EXPRESS model with XML hierarchical concepts. And we must also map EXPRESS attributes to XML elements (rather than XML attributes) in order to provide a basis for the hierarchical structures.

Aggregate Types
An EXPRESS attribute whose data type is an aggregate type should be represented in XML as a sequence of *`base`* elements where *`base`* is the XML element type corresponding to the base type of the aggregate. For example, consider a polygon modeled as a sequence of three or more vertex points. The EXPRESS model would be

```
ENTITY polygon;
  vertices : LIST [3:?] OF point;
END_ENTITY;
```

and an XML instance might look like

```
<polygon id="poly1">
  <vertices>
    <point-ref ref="e1"/>
    <point-ref ref="e2"/>
    <point-ref ref="e3"/> ...
  </vertices>
</polygon>
```

This creates a need for XML element types that correspond to all the possible data types that can appear in an aggregate

type, which are in fact all the possible data types. That includes simple data types as well as all new data types declared in the model. Because EXPRESS requires that all these type names must be different from all entity type names and must be unique across the model, these can be mapped to identical XML element names. But because the base types of aggregate types could also be aggregate types, it is necessary to create an XML element type for each aggregate type that occurs in the model. The names of these types could be derived from the EXPRESS syntactic designators for aggregate types, taking measures to make sure the names do not clash with user-defined names. For example, a list of strings could be marked up as:

```
<list-value type="string">
  <string-value>foo</string-value>
  <string-value>bar</string-value> ...
</list-value>
```

It is the handling of aggregates that makes the use of XML attributes for EXPRESS attributes a problem. One cannot put this complex content in an XML attribute. Aggregates of some simple types can be represented by XML attributes (as NMTOKENS or IDREFS), but many other aggregates must be represented by marked up text. And the members of such an aggregate must appear as instances of the XML element type corresponding to the base data type, as above. But when the EXPRESS attribute is mapped to an XML attribute, there is no XML element to hold these data elements as content. A solution proposed by the project was to give XML ID attributes to the elements that correspond to EXPRESS aggregate types, as if they were entity types. For the EXPRESS attribute whose value is an aggregate, the corresponding XML attribute has type IDREF. Unfortunately, this adds a level of indirection and causes the XML document to be difficult for humans to read and for software to process.

For example, consider an instance of the `axis_labels` EXPRESS entity appearing earlier, where the x attribute has the value ("foo", "bar") and the y attribute has the value ("baz"). Mapping EXPRESS attributes to XML attributes produces XML like:

```
<axis_labels x-id="a1" x="l1" y="l2"/>
<list-value x-id="l1" type="string">
  <string-value>foo</string-value>
  <string-value>bar</string-value>
</list-value>
<list-value x-id="l2" type="string">
  <string-value>baz</string-value>
</list-value>
```

We conclude that the aggregate value requires a hierarchical structure that cannot appear as an XML attribute value. And for consistency of representation of EXPRESS attributes, that form should be used in all cases.

3 CONCLUSION

The above discussion demonstrates some of the problems of what we see as a common desire: to provide XML-based exchanges of information that has been carefully modeled in relational or object-oriented languages. The underlying problem is that there is a significant mismatch between the important concepts in these modeling languages and the important concepts in XML. Relational ideas include "flattened" table rows and keys. Object modeling concepts include inheritance, polymorphism, and pointers. XML, however, emphasizes hierarchical structures and annotation. Specifying the mapping to XML using a non-DTD schema language and datatyping vocabulary bridges the gap somewhat, but not completely. Recognizing this, the project plans to develop a standard mapping from EXPRESS models to XML Schema and to investigate configuring the mapping to take advantage of XML's hierarchical structure.

ACKNOWLEDGEMENTS

The authors wish to acknowledge the members of the ISO TC184/SC4 "XML representation of EXPRESS schemas and data" committee, whose collective efforts led to many of the ideas presented in this paper.

REFERENCES

1. ISO 10303-11:1994. *Industrial automation systems and integration - Product data representation and exchange - Part 11: Description methods: The EXPRESS language reference manual*.

2. ISO TC184/SC4/WG11. *ISO/PDTS 10303-28:Product data representation and exchange: Implementation methods: XML representation of EXPRESS schemas and data*. Revision N140. 2000-10-16. On-line at http://www.nist.gov/sc4/wg_qc/wg11/n140/.

3. W. Eliot Kimber. *XML Representation Methods for EXPRESS-Driven Data*. National Institute of Standards and Technology. GCR 99-781. November 1999. On-line at http://www.nist.gov/sc4/wg_qc/wg11/n095/nistgcr99-781.pdf.

4. ISO/IEC DTR 22250-1. *Document Description and Processing Languages—Regular Language Description for XML (RELAX)—Part 1: RELAX Core*. 2000 October. On-line at http://www.xml.gr.jp/relax/.

5. James Clark. *TREX - Tree Regular Expressions for XML*. 2001-02-13. On-line at http://thaiopensource.com/trex/.

6. World Wide Web Consortium. *XML Schema Part 1: Structures*. W3C Candidate Recommendation. 24 October 2000. On-line at http://www.w3.org/TR/xmlschema-1/.

7. ISO TC184/SC4/WG11. *XML representation for data sharing (CEB Binding - Draft 3.0)*. Revision N1362000-09-29. On-line at http://www.nist.gov/sc4/wg_qc/wg11/n136/.