Issues in Mapping GenCAM℠ to XML

Michael McLay
NIST
Electronics & Electrical Engineering Laboratory
email michael.mclay@nist.gov

Andrew Scholand
Georgia Institute of Technology
Engineering Information Systems Laboratory
email andrew.scholand@cad.gatech.edu

Robert Fulton
Georgia Institute of Technology
Engineering Information Systems Laboratory
email robert.fulton@me.gatech.edu

ABSTRACT

This paper outlines some design issues uncovered in our investigation of an XML mapping of the Institute for Interconnecting and Packaging Electronic Circuits GenCAM℠ 1.1 Standard. XML is an industry standard format developed by World Wide Web Consortium as an extensible replacement for HTML. Because it is flexible and concise, XML is rapidly being adopted as a standard mechanism for representing complex documents with industry specific content. GenCAM℠ is an ANSI-approved PCB/A (Printed Circuit Board/Assembly) data standard sufficiently detailed for tooling, manufacturing, assembly, inspection and testing requirements. The shift from the current ASCII format of GenCAM℠ to a specific XML markup language will not change the higher level, semantic objects represented by GenCAM℠. Only the instance file syntax will change to a newer and more widely used format. The results of our research to date support the utility of a convergence of XML and GenCAM technologies.

INTRODUCTION

Realizing that data exchange problems cost the U.S. Electronics Industry upwards of 150 million dollars annually [7], the IPC (Institute for Interconnecting and Packaging Electronic Circuits) created the GenCAM (generic computer-aided manufacturing) format. GenCAM℠ facilitates and reduces errors in the communication of PCB/A manufacturing data from the designer to the fabricator [9]. These standards (IPC-2511 through IPC-2518) integrate functional descriptions, test data, and administrative information into a single file format. This format is sufficiently detailed for tooling, manufacturing, assembly, inspection and testing requirements. A rigorous design and development process under the supervision of the IPC produced a highly efficient object model for information interchange of data related to these activities.

XML (eXtensible Markup Language) is a web-enabled data interchange language derived from the SGML (Standard Generalized Markup Language). The W3C (World Wide Web Consortium) [1] approved XML as a W3C recommendation in February 1998. XML is used to define application-specific markup languages to represent structured data.

An important feature of XML is validation. Validation in XML-defined grammars allows an application to ensure that a given instance of data is complete, correctly hierarchical, and with acceptable content values. Comparing a data instance to a DTD (Document Type Definition) enforces validity. A DTD is a computer processable description specifying which tags and attributes, and in what order, are allowed in the data instance file. A DTD can be included in line with a data file, or the instance file can point to a DTD using the URI (Universal Resource Identifier) mechanism.

An XML version of GenCAM℠ is desirable for greater integration with the next generation of IPC standards for the electronic interconnection industry, CAMX. We have been researching the possible alternative ways of creating a specific XML markup language mapped from the GenCAM℠ standard, with the aim of providing input to the IPC GenCAM℠ Committee for future versions of GenCAM℠. Our working title for this generic mapping process is GenX, an abbreviation of the term GenCAM℠ in XML. Once a final design is approved by the IPC and its member companies, the GenX DTD will be similar in some respects to the contents of the IPC-251x series of documents that currently define the GenCAM℠ standard; both describe what is legally allowable in the file. Our research to date has shown, however, that achieving perfect correspondence between the current GenCAM℠ standard and an XML version is difficult. This paper documents some of the difficulties and design issues we have encountered in mapping from the existing GenCAM℠ object model to a DTD-constrained XML model.

GENCAM℠ DATA REPRESENTATION

The current version of GenCAM℠ uses a special purpose file format derived from a proprietary format developed by a CAM software vendor. The format is essentially a mapping of a network of objects in computer memory to an ASCII representation. The memory pointers are mapped to ''qualified strings'', e.g. ''board1''.''U1'' might be a component identifier. The first portion of the string is an identifier for a namespace. A typical use for a namespace would be to group together all package, device, or component definitions used on a board or assembly into a collection. The separate namespace is needed to keep the names of packages, devices, or component identifiers from conflicting with each other when more than one board or assembly is included in a single GenCAM℠ file.

MAPPING GENCAM℠ TO XML

The GenCAM℠ and XML file formats are not significantly different. Some punctuation changes will be required. XML has some readability enhancements because it uses named parameters. The mechanism used to point between objects in the file is problematic. The next section will provide some background and then the discussion of pointer mapping will be presented.

Terminology

The GenCAM℠ specification uses terminology to describe objects in the GenCAM℠ file format that are specific to the printed circuit board and printed circuit assembly industry. As a point of reference some comparisons to more widely recognized object terminology are provided here. First, a mapping of terminology to C++ and Java implementation constructs will be provided. The second mapping will be between GenCAM℠ and XML.

In C++ and Java a class defines a new type of object and the class name is the name of the object type. In XML the object definition is called an ELEMENT. In GenCAM℠ the equivalent of a class name or an element is the keyword used in a GenCAM℠ keyword statement. An example from the $ADMINISTRATION section of GenCAM℠ is the keyword statement definition is shown below:

```
    BOARD: <shorthand_board_ref>,
       <line_item>, [<quantity>];

<shorthand_board_ref> ::= string
<line_item>  ::= string  <quantity>  ::= p_integer
```

Example 1 GenCAM 1.1 BOARD keyword definition

An instance of this definition would look like:

```
BOARD: "PS134XX", "24", 50;
```

Example 2 GenCAM 1.1 BOARD keyword usage

The <shorthand_board_ref>, <line_item>, [<quantity>] portion of the statement are the attributes of the keyword. The attributes are the equivalent of members in Java and C++ or an ATTLIST in XML.

In GenCAM℠ the types of attributes are limited to ''qualified string'', string, pinteger (positive integer), number, pnumber (positive number), and FIXED_FIELD. The '[' and ']' brackets around an attribute indicates that it is optional.

In Java and C++ the string, pinteger, number, and pnumber map to fundamental types of the language. The FIXED_FIELD could be mapped to an enumerated type, or a list of strings. (The implementation decision would probably depend on the likelihood of the list being extended.) The ''qualified string'' is the equivalent of a pointer to a different object. In XML the types of ATTLIST members are limited to string, and enumerated list, and tokenized types. In GenCAM℠ we used only 3 types of tokenized attributes. The ID attribute is a unique identifier for the element instance, and so may occur only once per element. An IDREF attribute value must reference an ID. Validating parsers will check to ensure that all IDREF values correspond to valid IDs in the document. A NMTOKEN attribute is used to signify a name token. A name token has the same content rules as ID or IDREF attributes (only alphanumeric characters and '.','-','_', and ':') but without the parser-enforced regulations of uniqueness (for IDs) or valid referencing (for IDREFs). The GenCAM℠ to XML mapping of pinteger, number, and pnumber will require post-XML processing of the string elements to convert the string to the fundamental type of the underlying implementation language.

FIXED_FIELDS should be mapped into enumerated lists. Enumerated lists are preferable to the string representation mentioned above (where 'string' is a CDATA attribute) because of the relative ease of keeping the GenX DTD current with evolving technology. As long as changes to the DTD are backwards compatible (as additions to enumerated lists are), the DTD can be maintained at a centrally managed URL and be kept current independent of CAD/CAM/CAE software release cycles.

There are several possible mappings for the ''qualified string'' type. They could be mapped to IDREF attributes (which requires the NAME attributes of the referenced objects to become type ID). Another possibility is to map the two parts of the qualified string to two separate ATTLIST CDATA entries. In this case only GenCAM-specific XML applications would be able to find the referenced objects, since the two attributes must be combined to provide location information. Finally, the qualified string could become a single ATTLIST CDATA entry. Unfortunately, depending on the concatenation method used, it may be difficult to reconstitute the all of the original information contained in the CAD file. The mapping options will be discussed in detail later in this paper.

The result of these mappings is an XML element which looks like the following:

```
   <BOARD REF="PS134XX" LINE_ITEM="24"
 QUANTITY="50">
```

Example 3 GenX <BOARD> element usage

Note in Example 3 that the term ''shorthand_board_ref'' was changed to ''REF''. This shortening of attribute names where possible is part of the proposed convention used when mapping from the current notation to XML. The rules used when making mapping decisions are discussed in more detail below.

DESIGN GUIDELINES

To facilitate our translation work, we attempted to codify our design philosophies into a few simple principles. This facilitated communication and collaboration with other researchers interested in the project, and maintained consistency in the mapping process.

Rule 1: Don't make up new keywords

We attempt to use element and attribute names that exactly match the keywords in the GenCAM℠ specification. This is not possible, however, since the GenCAM℠ specification is loaded with polymorphic references: the semantics of a keyword can change substantially depending on the number and type of parameters following it. XML, in spite of being a way of indicating an object hierarchy, does not allow for polymorphism, and requires each element with different behavior to be named differently. This leads to our second rule.

Rule 2: Keep consistency with similar GenCAM℠ objects

When we could not achieve one to one mapping between all the GenCAM℠ keywords and the GenX element and attribute names, we attempted to consistently apply the same design principles in creating new names or elements. For example, the GenCAM℠ specification uses the same keywords to identify standard graphic primitives (such as a line, a circular arc, and an ellipical arc)

in both a special section reserved for grouping primitives (the $PRIMITIVES section) and elsewhere in the file. The specification places a restriction on the definitions that occur within the $PRIMITIVES section that limits the number of attributes they can have. (Primitive shapes defined in the $PRIMITIVES section must be generic in nature. Thus, they are not allowed to have attributes that refine their description, such as geometric placement, or coloring and rendering details.) The same keyword is hence overloaded- when occurring within a $PRIMITIVES section it has only a subset of the attributes. To create a second element that could be used within the $PRIMITIVES, we appended the suffix 'DEF' to the keyword. Almost all of the modified keywords end in either 'DEF' (for a definition element) or 'REF' (for a element referencing a definition elsewhere).

Rule 3: Attributes are preferable to elements

The GenCAM℠ specification contains many examples of what we call 'simple objects': a GenCAM℠ keyword with a single value. A good example is the HISTORY keyword which appears in the $HEADER section. This keyword only ever occurs within the HEADER section, and it always has a single value in any given document.

In mapping GenCAM℠ to XML, a choice is available to map any given keyword to either an element name or, if certain conditions apply, an attribute. Some DTD design guidelines recommend element formulations over attribute formulations owing to the extensibility and reusability of elements. However, if descriptive names are used, a non-empty element formulation has the disadvantage of requiring the repetition of the tag name after the element content to close the element. Since the GenCAM℠ in XML instance file will be the central means of transmitting information between the various stages of product realization, it is imperative that it be kept as small as possible. Attribute formulations offer approximately half the size requirements of element formulations.

The selection of an attribute mapping instead of an element mapping must make good design sense. The attribute value should be sensibly related to the parent element. It should have only a single value and should only occur once within the parent element.

Rule 4: All parsing should be done by the XML parser

A central goal of the mapping from GenCAM℠ to XML was to ensure that each meaningful atomic data value is mapped to a single element or attribute within the DTD. This means that applications built on top of the XML format will not have to further subdivide any information provided by the XML parsing engine.

Although this advantage can be referred to in shorthand notation as ''no further parsing required'', in reality several further processing steps remain for the GenCAM℠ application to carry out. XML only recognizes strings as types for value fields. It doesn't know how to parse floating point numbers, integers, positive integers, or return integers for values selected from a enumerated (fixed field) data type, so these type conversion operations need to be performed. XML also only has one namespace for element references, so if a multiple namespace object design is desired, applications will have to manage the name references internally.

Rule 5: Plan for conformance checking software

We attempted throughout the XML mapping process to place as many of the GenCAM℠ validity constraints as possible within the structure of the DTD itself. Thus, for example, as described in Rule 2 above, we created a new element when a subset of attributes was explicitly called for by the GenCAM℠ specification. However, Document Type Descriptions do not enforce strong typing of data, and some constraints require conditional information. For example, the DATE element in the DTD is restricted to have a MONTH value in the range of 1 to 12, and a DAY value in the range of 1 to 31, but ensuring that February has no more than 28 or 29 days, as appropriate, requires a software program.

We refer to any software program that enforces constraints additional to those imposed by the DTD a ''conformance test module''. We believe that at least one implementation of a conformance test module should be made open source and attached to the standards document which specifies the XML version of GenCAM℠. This serves the double purpose of both concisely describing exactly what additional constraints are to be enforced before a file can be considered ''conformant'' to the standard, and serves as a code base interested parties can extend for further GenCAM℠ processing. A textual and/or algorithmic description of the conformance test module should also be included.

NAMESPACE ISSUES

In GenCAM℠, keyword names must be unique within a SECTION. In an XML file there is only one namespace for all IDs. This will require GenX to create a mapping between the namespace management features of GenCAM℠, such as ''qualified string'', and the flat namespace in XML.

The mapping between GenCAM℠ and XML namespaces is a design issue, and as such there are many potential conceptual approaches that may be taken. Each alternative has a different instance file footprint impact and a different level of robustness against accidental data contamination. The following design alternative represents a relatively close match to the existing GenCAM℠ specification. We present it here merely as an illustrative example, and with the caveat that this particular style has not been endorsed GenCAM℠ committee.

Consider the following GenCAM 1.1 example:

```
GROUP: "XXX.YYY";
  HOLE "hole1", 0.1;
...
THING: "whichhole", "XXX.YYY"."hole1";
```

Example 4 GenCAM℠ 1.1 GROUP and referencing syntax

Example 4 illustrates how GenCAM℠ implements a layered local/global namespace concept. A GROUP keyword, in this example with value ''XXX.YYY'', creates a namespace, which contains a HOLE object with the local name of ''hole1''. A reference to the HOLE object from outside of the GROUP ''XXX.YYY'' is outside the local namespace, and so must use a global syntax combining the quoted group name and the quoted local name with a period.

Groups do more than just create namespaces for objects that may belong on different boards but have the same short name, however. Groups can represent real segmentation of design relationships. For instance, associating all the PATH names in a ROUTES section GROUP to be included in a BOARD.

We now show the same example in one possible mapping to XML:

```
<GROUP NAME="XXX.YYY">
  <HOLE NAME="XXX.YYY:hole1"
   DIAMETER="0.1" />
...
<THING NAME="whichhole"
 HOLE="XXX.YYY:hole1" />
```

Example 5 GenX <GROUP> and referencing syntax

The use of the delimiter character ':' is used to separate the group portion of the identifier from the name portion. This facilitates translating an internal representation into a single CDATA attribute without losing information concerning what portion of the attribute is a GenCAM℠ artifact (due to the imposed structure of GROUPS) and what portion is CAD-generated data. It does impose the single constraint on the GROUP identification string to not contain the character ':'.

One nice feature of the XML mappings immediately apparent in the example above is the visual clues provided about the thing being referenced at the point of reference. For instance, the string ''XXX.YYY:hole1'' is clearly labeled as being a reference to a HOLE. Although the DTD is not shown in this example, the NAME attribute of the HOLE is of type ID, and the HOLE attribute of the THING element is of type IDREF. Notice that unlike the GenCAM℠ example in Example 4, the HOLE NAME is a fully expanded string. This is necessary under the flat namespace model of XML. The disadvantage of this approach is the requirement for more characters in the instance file to expand the strings in the GROUP statements. Preliminary calculations show a subsequent increase in file size on the order of 3 to 5 percent.

A more serious disadvantage is the potential for different human and computer interpretation of the data. A person reading an instance file would observe that the GROUP statements appear to ''own'' or in some other way manage the items occurring within their tags. The computer-interpretable references to objects, however, are handled strictly with the expanded strings. This discrepancy between the apparent function of the groups (managing namespaces) and the actual function (convenient logical groupings of related objects with no enforced checking) could lead to the introduction of errors. The advantage of this notation, however, is that it leverages the error checking features of XML, thereby reducing the low-level processing burden on the software built on top of the XML parser. The direct use of ID and IDREF attributes allows generic validating XML parsers to identify some instances in which an XML based file is incorrectly defined. (A validating parser will determine if a given IDREF attribute points to an ID attribute somewhere in the file, but it cannot verify that the type of object referenced is semantically the type desired.) To validate a GenCAM℠ XML file, a GenCAM℠ Document Type Definition (DTD) is needed. The DTD can either be supplied with the GenCAM℠ file, or the GenCAM℠ document can reference the DTD at a Uniform Resource Locator (URL), such as http://www.gencam.org/dtd/. A validating XML parser can be used to find certain syntactic level errors in the document, such as references to element names that aren't defined.

The need for ID verification as a debugging or error labeling process in generic tools is not the only issue. A generic XML parser would not, of course, detect GenCAM specific errors in the design file so these errors would be silently propagated. Testing a file for conformance is the purpose of validation tools and these tools could be run as a filter on any computer system, not just on a CAD or CAM system. Rather, a flat namespace is important because it means that generic XML applications can follow the cross-reference links to gather information that is distributed across multiple objects.

FUTURE WORK

The XML mappings of GenCAM we have experimented with to date have used the same objects proposed by the GenCAM committee for modeling geometric entities—lines, various types of arcs, polygons and the like. A recent, broadly supported proposed XML standard for geometry representation, SVG (Scalable Vector Graphics) [3], could be used as the base geometry model instead.

Because SVG is an extremely compact notation, and these geometric primitives make up the vast majority of data sets, the file size of an XML GenCAM℠ document could decrease to less than half the current GenCAM℠ file size. We believe this benefit merits further exploration.

We are also watching the development of the XML Schema working drafts [10], [11] with interest. The XML Schema Language offers the ability to match the GenCAM℠ data model more closely than is possible with a DTD, increasing the amount of conformance checking that can be done by the parsing engine. The resultant simplification of the subsequent layers of software built on top of the parser is desirable, but must be weighed against potential drawbacks such as the degree of native XML Schema support among the various XML editors, parsers, and utilities.

We would also like to see the GenX design support, but not require, splitting GenCAM℠ into network addressable objects. That is, the GenCAM℠ in XML design file should reference devices, patterns, and packages from standard network libraries whenever possible. At the time of this writing, the fundamental XML infrastructure to support this design objective was not finalized. As the state of the XML Fragment Interchange Standard [6], the XLink (XML Linking Language) [3], and the XPointer (XML Pointer Language) [5] continues to evolve, this objective will be achievable and will merit further attention.

CONCLUSION

We have discussed some of the key issues in mapping the GenCAM℠ object model to XML. These included difficulties with polymorphic references, conversion from multi-tiered namespaces to a flat, global namespace, and issues in enforcing data typing. We believe satisfactory solutions to these issues are available through consistent use of element naming rules, delimited ID and IDREF attribute pairs, and post-XML parsing conformance verification software. In addition, two dimensional graphics and referencing work currently in progress within the XML community appear to offer substantial benefits to an XML representation of PCB/A data.

BIBLIOGRAPHY

[1] Extensible Markup Language (XML) http://www.w3c.org/XML

[2] Extensible Markup Language (XML) 1.0 Specification http://www.w3.org/TR/1998/REC-xml-19980210

[3] Scalable Vector Graphics (SVG) Specification, W3C Working Draft 11-February-1999 http://www.w3.org/TR/WD-SVG/

[4] XML Linking Language (XLink) http://www.w3.org/TR/WD-xlink

[5] XML Pointer Language (XPointer) http://www.w3.org/TR/WD-xptr

[6] XML Fragment Interchange Requirements, Version 1.0 http://www.w3.org/TR/NOTE-XML-FRAG-REQ

[7] Parkinson, Harry, and Minchella, John. (1996) IPC Proposal submitted in reponse to ARPA BAA 96-16, Electronic Systems Manufacturing (ESM) and Design Support for Mixed-Technology Integration.

[8] Scholand, A., Peak, R., and Fulton, R., (1999) ''Enabling Distributed Data Processing for Internet Analysis with GenX'', Paper DETC99/CIE-9076 in the Internet-aided Design, Manufacturing and Commerce track of the 19th Computers in Engineering Conference, Las Vegas, NV.

[9] GenCAM Specification and Related Documents http://www.gencam.org/docs/

[10] XML Schema Part 1: Structures http://www.w3.org/TR/xmlschema-1/

[11] XML Schema Part 2: Datatypes http://www.w3.org/TR/xmlschema-2/