XML-Based Representation

Robert L. Kelsey
Los Alamos National Laboratory
Mail Stop F645
Los Alamos, NM 87545 U.S.A.

Abstract For focused applications with limited user and use application communities, XML can be the right choice for representation. It is easy to use, maintain, and extend and enjoys wide support in commercial and research sectors. When the knowledge and information to be represented is object-based and use of that knowledge and information is a high priority, then XML-based representation should be considered. This paper discusses some of the issues involved in using XML-based representation and presents an example application that successfully uses an XML-based representation.

Keywords: knowledge representation, XML, object-based representation

1 Representation Issues With XML

The eXtensible Markup Language (XML) [13] and its parent the Standard Generalized Markup Language (SGML) [7] are languages used to create markup languages. They can also be used for creating information and knowledge representations for specific types of applications. Applications with knowledge and information that can be decomposed with objects are particularly well suited for XML-based representations [9]. Additionally, applications that must repeatedly manipulate the information and knowledge represented can be well served with an XML-based representation.

XML contains three constructs that are used to create a document type definition (DTD) or grammar for a markup language. The three constructs are element, attribute, and entity. Attribute constructs are associated with element constructs. An entity construct is like a macro that can be expanded. Any duplicated definitions can be specified as an entity and called to save time and lengthy text repetition. In an object-based context an element is an object and an attribute is an object attribute. Attributes are the properties and characteristics of an object.

An element construct can contain other elements. In this way, a taxonomy of objects can be represented. An XML-based representation can contain aggregation and inheritance relationships, although these relationships are not as explicit as what may be found in ontological representations. Other relationships and notation which may contribute to a logical reasoning mechanism are also missing from an XML-based representation. Some of these relationships and notation could be explicitly added to an XML-based representation.

It has been argued that XML lacks the formal semantics necessary for representing complex objects and relations [4]. This is agreed, but many languages and representation schemes suffer from the same problem. XML is useful for representing declarative knowledge and information. Procedural knowledge and information is another issue. Procedural knowledge is knowledge about how to perform a task or activity [1]. This type of knowledge is somewhat difficult to define and much harder to represent. Production rules are often used to represent procedural knowledge, but rules are not that distinct from other forms of
representation for declarative knowledge [11]. Data interchange and knowledge and information sharing contribute to adding semantics to XML, especially due to e-commerce applications [4, 12].

Interchange and sharing are application uses which require a certain level of procedural knowledge or "application-level processing semantics" [4]. In the example application discussion that follows the XML-based representation is for declarative type knowledge and information. The knowledge and information necessary to achieve application use and processing is embodied in the programs, scripts, and routines that manipulate the knowledge and information in the XML-based representation.

An XML-based representation is not the best path to take for all applications. The tradeoffs depend on the application, its size, the number of users, time to deployment, type of knowledge/information, and other considerations. If the application makes heavy use of the represented knowledge and information and is focused in its use and users, then an XML-based representation may be the right choice. The following sections describe an example application that uses an XML-based representation that is showing success with its users.

2 Application Example

2.1 Introduction

An application example that not only illustrates the use of XML for representation but how XML can help tie together separate pieces of knowledge and information is an enterprise system for simulation of physical systems. At the foundation of the system is an object-based representation of physics and simulation constructs and concepts implemented in XML. Additional XML tools and software facilitate the storage, sharing, and use of the represented knowledge and information. These tools include an XML-specific database, conversion programs written in Java, and Java/XML standards and APIs for XML content manipulation.

In this application users have a number of in-house physical simulation codes at their disposal. These codes have been created and modified over the course of many years and are written in several different languages including FORTRAN, C, and C++. Each code has its own unique input file format and execution nuances. There are many expert users in the use of one or two of these codes, but there are few generalists in the use of many of these codes. The enterprise system facilitates the use of more of these codes by an individual user. It also provides the means for migrating users and problem sets to more modern codes.

2.2 How XML Is Used

An XML DTD was created that serves as the representation of physical system simulators. The representation is a taxonomy of physics and simulation objects and their associated properties and characteristics. The DTD is also the grammar of the representation language. Creating an XML instance of the DTD creates a description of a particular physical simulation problem.

The representation at the highest level includes objects for a computational mesh, simulation startup information, a material set, a geometric body set, and simulation output information. These highest-level objects are further defined with sub-levels of objects as much as four levels deep. Figure 1 shows an abstract view of a portion of the material set object in the representation. The attributes associated with these objects are not shown.

A user may create a problem description (in XML) from scratch using a graphic user interface for this purpose. The interface is called XEENA [6] and is an XML editor from IBM Alphaworks. However, it is more typical for a user to convert an input file (for a particular simulation code) to the XML-based representation and then to another simulation code input file format.

An XML database provides for a repository
of known and common problem descriptions. These are example problems which can be used for testing simulations for verification and validation purposes and for training new users. The XML database is also used as a database of materials and their associated properties. In this use, rather than a complete problem description, only instances of the material portion of the representation are stored in the database.

The material database contains a set of standard materials and their properties. The database can be queried by material name and by material description. Materials can be accessed from the database and put directly into an XML problem description. This ability is especially useful for creating problem descriptions from scratch or substituting materials in an existing problem description.

Another part of the enterprise system is a language [8] for specifying a suite or series of problem runs. This language is also implemented in XML. A user can specify an example problem and make substitutions (such as materials) in that problem to create a series of problems. This is useful for parameter testing and making comparisons between simulation codes. The example problem (in XML form) as well as the problem portions being substituted can originate in files and the database. Some types of substitutions can be specified in the language explicitly or with functions or equations. Each individual run specifies a target simulation code and as many substitutions as desired. This allows the creation of a problem description from distributed and disparate sources.

Once substitutions for a given run are completed, the XML problem description is converted to the target simulation code input format and the problem is submitted for execution. The language also allows the user to specify what output parameters are to be compared between runs. The purpose of the language is not only job (execution) control, but to facilitate problem creation from disparate sources and to specify basic output analysis.

Figure 2 shows the enterprise system and its components. Along the left are sources for physical system problems or pieces to build those problems. These sources may be in XML format or in an application code specific format. The database sources may be queried and put directly in the XML problem description. The interchange format is XML and the target format is application code specific. The XEENA interface can be used to create problem descriptions from scratch and from these...
The problem execution portion of the enterprise system takes the XML problem description and an XML run specification and executes the problem. A simulation driver exists for each target application code. The driver contains application specific details needed to submit and execute the problem for the particular application code. Execution results are generated and a basic analysis is performed on the results. The documentation can be tailored for brief descriptions, formal reports, and slide presentations. The XML databases are an open-source version of dbXML [5]. The conversion routines make use of the Xerces Java parser [3] for XML and the Saxon XSLT processor [10]. The documentation tools make use of Xalan Java [2].

Whenever possible, open source tools and code were used to help build the enterprise system application. There is an increasing number of XML tools available, both from open source and commercial vendors. This availability contributes to the rapid creation of prototypes. Open source use helps avoid re-invention and can help ensure that the most recent advances in XML are being used.

This work has shown that XML is useful for quickly creating application-specific and application-oriented representations. The representations are relatively easy to create, maintain, and extend. They are easy to use and easy to build applications around. Part of the ease of use is due to the availability and increasing availability of standards, APIs, and tools for XML. However, XML-based representations are not without problems. The biggest problem is that the representations lack enough semantic content for some applications. If the terminology and/or vocabulary expressed in an XML-based representation is not known and understood by the users and use applications, then the representation will likely be worthless. More semantic content is necessary for representations that will be shared far and wide among users and use applications without complete understanding of the knowl-

Figure 2: The components of the enterprise system for physical system simulation.

3 Results And Conclusions

Users regularly use the conversion routines for input file format conversion. The routines are especially used for migrating old and legacy application code input files to newer application code input files. There is some interest in newer application codes adopting the XML problem description as their input file format.
edge and information being represented. These are some of the tradeoffs which must be considered.

When an application is focused and intends to use represented knowledge and information, then an XML-based representation should be considered and attempted. It is faster to create an XML-based representation than a full-scale ontology. If an XML-based representation does not work for the chosen application, then it can be cast aside and more traditional methods can be applied. A test of XML does not need a lot of time and effort. It is important to remember that in applications, using what works or what is good enough can be the best path to take.

References


