LA-UR- 04-1564

Approved for public release;
distribution is unlimited.

Title:	XML-Based Resources for Simulation
Author(s):	Robert L. Kelsey Jane M. Riese Ginger A. Young
Submitted to:	SPIE Defense & Security Symposium Orlando, FL April 12-16, 2004

XML-Based resources for simulation

Robert L. Kelsey[a], Jane M. Riese[b], and Ginger A. Young[b]

Los Alamos National Laboratory
[a]X-8 MS-F645, [b]CCN-12 MS-B295,
Los Alamos, NM 87545

ABSTRACT

As simulations and the machines they run on become larger and more complex the inputs and outputs become more unwieldy. Increased complexity makes the setup of simulation problems difficult. It also contributes to the burden of handling and analyzing large amounts of output results. Another problem is that among a class of simulation codes (such as those for physical system simulation) there is often no single standard format or resource for input data. To run the same problem on different simulations requires a different setup for each simulation code.

The eXtensible Markup Language (XML) is used to represent a general set of data resources including physical system problems, materials, and test results. These resources provide a "plug and play" approach to simulation setup. For example, a particular material for a physical system can be selected from a material database. The XML-based representation of the selected material is then converted to the native format of the simulation being run and plugged into the simulation input file. In this manner a user can quickly and more easily put together a simulation setup. In the case of output data, an XML approach to regression testing includes tests and test results with XML-based representations. This facilitates the ability to query for specific tests and make comparisons between results. Also, output results can easily be converted to other formats for publishing online or on paper.

Keywords: physical system simulation, XML-based representation

1. INTRODUCTION

The search for answers to large and difficult problems has driven an increase in the size and complexity of computer simulations and models. This increase in the software has in turn driven an increase in the size and complexity of computer hardware. Bigger problems need bigger machines able to execute bigger simulations. A decade ago grand challenge problems such as global climate modeling, multi-phase flow, new materials technology, and quantum chromodynamics were performed on computers like Thinking Machines CM-5.[1] Now, the CM-5 is no more.

High-performance computing hardware has moved into an era of distributed systems with massively parallel processors. These distributed systems have required advances in networking technology, mass storage systems, and heterogeneous computing capability.[2] Such systems have become increasingly complex to design and build. In fact, the design and performance evaluation of such systems is done with simulations and models.[3,4] These simulations for designing modern computer systems are also becoming more difficult and time consuming to build.[3]

In some cases the problems being studied have not changed, but their complexity has increased. This can be due to the need for a higher fidelity answer or from adding more detail to the input data. Sometimes it is a change in scale or dimension to the problem that increases the complexity. For a class of physical systems problems the move from two dimensions to three is not trivial. It not only increases the number of inputs, the number of outputs, and the detail and granularity, but also makes the problem harder to think about.

Further author information: (Send correspondence to Kelsey)
Kelsey: E-mail: rob@lanl.gov
Riese: E-mail: jriese@lanl.gov
Young: E-mail: gingery@lanl.gov

All of these things have led to more complex problem setup, simulation execution, and output analysis. Now as high-performance systems evolve there is opportunity for supporting users tasked with these simulation activities. Even small steps can help make these activities easier to accomplish. The following sections discuss a set of resources designed to support the simulation activities of problem setup, simulation execution, and output analysis. Equally important is the use of eXtensible Markup Language (XML) in the design and implementation of the simulation resources, thus it is a prominent part of the discussion.

2. XML-BASED REPRESENTATION

2.1. XML Primer

The eXtensible Markup Language (XML)[5] and its parent the Standard Generalized Markup Language (SGML)[6] are languages used to create markup languages. A document that is written in a markup language contains the words of the document as well as labels about the words. The labels are markup and function as semantic information about the words being labeled. Figure 1 shows a simple example of a reference in a bibliography. At the top is the reference and at the bottom is an XML instance of the same reference. The markup in the XML instance gives meaning to the words the markup encapsulates. For example, the < *title* > markup contains the title of the reference. This is why XML documents are sometimes called self-describing documents.

```
Charles Jones, Fundamentals of Scheduling Algorithms,
  Journal of Computer Science Simulation, March, 2000.


<journalreference>
  <author>Charles Jones</author>
  <title>Fundamentals of Scheduling Algorithms</title>
  <journal year="2000" month="March">
    Journal of Computer Science Simulation
  </journal>
</journalreference>
```

Figure 1. A simple example of a bibliography reference is shown followed by the XML instance of the same.

One of the original purposes for XML was to facilitate the interoperability of data over the Internet.[7] This was to allow different parties with different platforms and applications to communicate and share data. In these types of applications XML is used to define an intermediary format between different data formats. The defined format becomes an interchange format and language. Interchange continues to be a primary application use of XML today.

To create an XML application one creates a document type definition (DTD). A DTD is like a grammar for a language. The DTD contains the rules that describe the notation for making the pieces of content in a document and how those pieces relate to one another. This is the syntax and structure of the language application being created. Some example XML applications are the Chemical Markup Language, MathML, and VoiceXML.[8]

XML need not be limited to just creating markup language applications. XML can also be used for creating information and knowledge representations. An XML-based representation can be thought of as an extremely light-weight ontology. The constructs within the representation have associated attributes that help describe properties and characteristics of each construct. There are relationships between the constructs but the types of relationships are limited. This is why as an ontology, an XML-based representation is much less powerful. While power is sacrificed there is a gain in flexibility and a reduction in complexity. Full-scale ontologies can be difficult to create and use.

An XML-based representation is particularly well suited for applications with knowledge that can be decomposed with objects. Other considerations include the breadth of the knowledge domain, type of knowledge, amount of knowledge, and the intended use of the represented knowledge. For example, if the knowledge is more

relational in nature as opposed to object based, then a relational database may be the better form of representation. An XML-based representation can provide for a taxonomy of objects with aggregation and inheritance relationships.[9] This leads to not just object-based knowledge but structured knowledge.

If the knowledge in the domain is focused and application specific this is another indicator that an XML-based representation should be considered. This is true also, if the knowledge is to be used as opposed to just contained. Examples of use might be sharing, re-formatting, and manipulation. Trying out XML for an application need not take a lot of time and effort. It is much faster and easier 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 for a more traditional method.

2.2. Simulation Data Resources

The simulation data resources are designed to support simulation activities. These activities include problem setup, problem execution, and problem output recording and analysis. The problem setup activity is biased towards the setup of problems specifically for simulating physical systems. However it could be easily generalized to accommodate other types of problems. The goal of providing data resources is to make the simulation activities easier, more uniform, and more streamline.

The activity of problem setup includes three resources. These are a materials database, an example problems database, and a tutoring database. The materials database contains materials and their associated properties and characteristics for use in physical systems problems. The example problems database contains a set of common problems that are used for testing simulation programs. It is also used to create new physical systems problems. Users will often begin with a similar problem and modify it as necessary to create a new physical systems problem. New problems are seldom created from scratch. The tutoring database can be used with the example problems database to teach new users about how to setup and execute physical systems problems.

The activity of problem execution includes the simulation job control resource.[10] The simulation job control resource helps a user to execute one or more problems. The user can specify the problem or problems to be executed and the simulation program or programs to be used. The source of problems can be specified such as from local files or the example problem database. Parameters can be substituted from one execution to the next. The source of substitutions can be specified from equations, local files, or databases. The materials database is an example source of parameters for substitutions. The simulation job control resource is particularly useful for executing parameter studies.

The activity of problem output includes the enotebook resource[11] and the test results database resource. The enotebook resource allows users as well as simulation programs to write and read entries into and from electronic notebooks. The enotebook resource can also be used within the problem setup and problem execution activities. The test results database resource is used to post and display the results of test suite executions. This is particularly useful for regression testing and other verification and validation (V&V) activities.

2.3. Creating Resource Representations

Each resource has its own representation and therefore its own DTD. The constructs for creating a DTD include the element construct and the attribute construct. Elements can have associated attributes to help describe their properties and characteristics. When doing object-based representation[12] the DTD constructs a map to object-based constructs. An element maps to a class. An attribute maps to an attribute of a class. Creating an instance of a DTD is similar to instantiating a class. The difference is that a DTD often has several elements arranged in a taxonomy. Thus the instantiation of a taxonomy of classes occurs all at once. Figure 2 shows a portion of the DTD for the bibliography reference example used previously.

Most of the representations for the simulation data resources are small. They each have less than a dozen element constructs in their DTDs. The number of levels in these DTDs varies from three to five. The largest of the representations is for specifying a physical system. The DTD for this representation contains over 80 elements and over 260 associated attributes. This DTD is as much as six levels deep. The representation for physical systems includes constructs for describing simulation startup information, a computational mesh, materials information, volumetric geometry information, and simulation output information. The representation for the materials database resource is actually a portion of the physical systems representation.

```
<!ELEMENT journalreference (author+, title, journal)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT journal (#PCDATA)>
<!ATTLIST journal
   year   CDATA   #REQUIRED
   month  CDATA   #REQUIRED>
```

Figure 2. A portion of the DTD for the bibliography reference example.

3. RESOURCE USE

3.1. Problem Setup

Typically a user starts with a known problem setup. This is usually in the form of a simulation input file. A user will modify this input file and evolve it into a new problem setup. These known problem setups can come from a user's own collection or from the example problems database. When the setup comes from a user's own collection it will usually be in a native format specific to a simulation program. Conversion routines are available to translate the native format to the XML format of the physical system representation. A setup that comes from the example problem database is in the XML format of the physical system representation.

The user is free to work with the format that is most familiar. When evolving a know problem setup, common modifications include substituting materials, changes to material models, and changes to the geometry. For example, the equation of state model for a material may be switched to better express a material property or behavior. As far as geometry, the shape of materials may be changed to study the affects on the system as a whole.

The materials database is designed for making easy substitutions of materials. A material can be selected from the database via query or browsing. The selected material can be plugged into the problem specification. The format of materials in the database is an XML format. This format can be converted to other native formats as necessary. The XML-based representations facilitate the ease of converting between formats. Figure 3 shows the resources available to the process of problem setup.

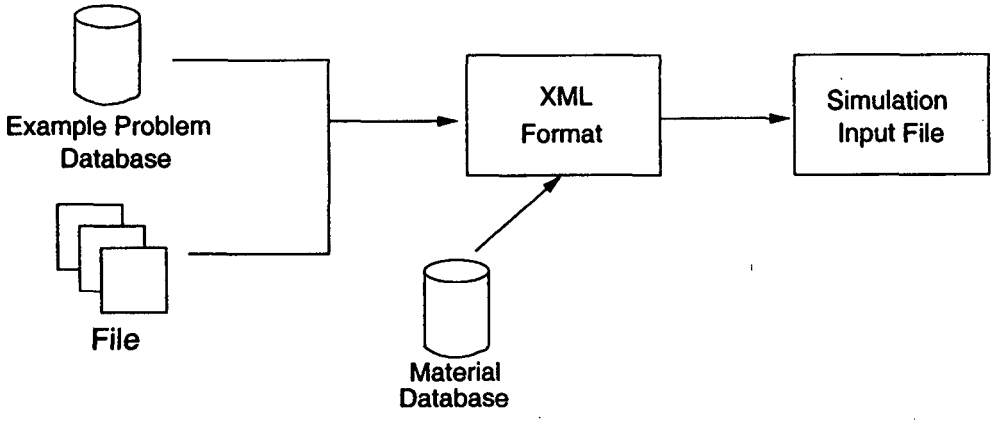

Figure 3. Shown are the resources available and the flow of the process of problem setup.

3.2. Problem Execution

When a user is executing a single problem starting the simulation can be as simple as a single command. Other simulation programs are not as simple to execute even when running a single problem. These simulation programs may require the user to interact via a graphic user interface (GUI) or other user interface. The user interfaces may also provide additional features such as batch submission of problems, multiple problem execution, and

execution monitoring. The goal of these user interfaces is to ease the task of problem execution and control as simulations and the hardware they run on becomes more complex.

The range of users varies but many of them are most comfortable with command-line interfaces. This is perhaps an artifact of the porting of legacy simulation programs from early Cray vector processors to the more modern massively parallel systems. It is also a choice of many users because command-line interfaces are quick and simple to use. Another advantage of their use is that commands can be scripted together in small programs. Although the legacy simulation programs have evolved to fit their current execution platforms, their use and user interfaces remain much the same.

The job control resource is neither a GUI nor a command-line interface. It is like a job control language represented in XML and designed specifically for simulation job control. Although it can be used to execute a single problem it is most useful for executing a series of problems. A user creates an XML instance of the job control representation to specify a series of problem executions. Each individual problem can be executed with a different simulation program. This allows simulations to be compared with one another. Parameters in each individual problem can be varied and substituted from one problem to the next. This allows parameters to be studied over the series of executions.

The job control specification functions not only as the set of execution instructions but also as a historic record of what was simulated and how it was simulated. This can be an important record for parameter studies, testing, and for reproducing problem runs at a later date. The enotebook resource can also contribute to the historic record. Using an application programming interface (API) simulation programs can write entries directly into a notebook without user intervention. Such entries include information about the physical system problem, information about the simulation, and information about the computing platform.

The most important piece of information about the physical system problem is the input file or the XML representation of that input file. While these items could be put in a notebook, in some cases this would result in large entries. It might be more useful to point to these items from a notebook entry or put portions of them in a notebook. The XML format facilitates breaking out portions of the representation. For example, if a user runs the same problem over and over again with only changing materials then just the material data from one run to the next need be recorded in a notebook.

Like a physical system input file a simulation program is too large to be an entry in a notebook. Therefore entries in a notebook contain metadata about the simulation program. The entries can also point to archived copies of the simulation program or copies in a configuration management repository. The metadata about the simulation program includes build date and version number, version numbers of libraries used in the build, and the compiler version used for the build. The libraries include off-the-shelf code libraries, open-source code libraries, in-house code libraries, and in-house data libraries. These are any code or data packages that are compiled into or linked into the simulation program.

Information about the computing platform is similar to the metadata of the simulation program. The computing platform metadata includes the operating system version; hardware information such as processor type, version, and available memory; and hardware configuration for the simulation run. The configuration specifics can include the number of processors being used and the distribution of processors and memory across the interconnection network. Any computing hardware platform information that can change from one execution to the next is important enough to record in a notebook.

All of the problem, simulation program, and computing platform information is entered by the simulation program into a notebook at the time of problem execution. The point of this type of note taking is to help with problem reproducibility. It is important to know what was done, when it was done, and how it was done. In this respect, the act of computer simulation should be no different than the act of laboratory experimentation. Note taking is vital to both. Figure 4 shows the resources available to the process of problem execution.

3.3. Problem Output

The output of these simulations is most often raw numeric data. Tools are then used to post process the data into other forms. Most of the output forms are for visual analysis. Large numbers of graphs and plots are still produced, but today's visual forms also include animations and movies. Tools for generating these visualizations

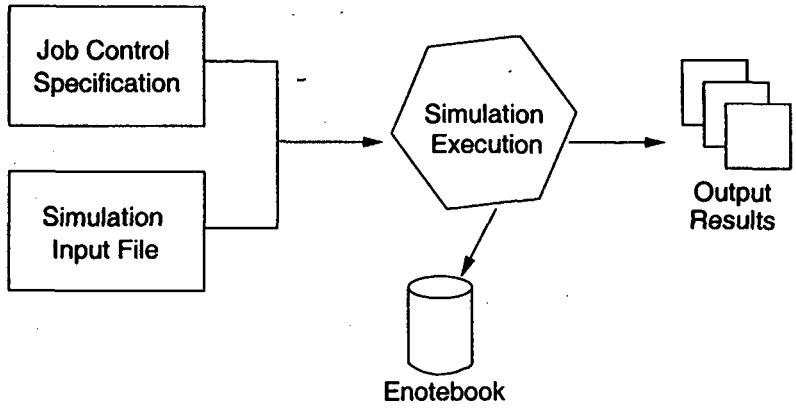

Figure 4. Shown are the resources available and the flow of the process for problem execution.

are a mixture of commercial and in-house tools. As the simulations have evolved so has the output data. The amount of data has increased tremendously. In some cases the amount of data is too much to handle. This fact has lead to more sophisticated tools for output analysis, especially for visual analysis.

The output resources are designed to help consolidate, organize, disseminate, and archive simulation output data. These functions can also help facilitate analysis of data. The enotebook resource is again included but this time as an output resource. As a simulation produces output the results can be written to a notebook by the simulation program. In this manner the results of a simulation are directly associated with the problem input information and the execution information. A chronology of information from start to finish is available.

The test results database functions as an archive of problem executions for testing. Like the enotebook resource, a simulation program can automatically write results to the database. The types of results that are written to the database depend on the simulation program, the problem being executed, and the user running the tests. The XML-based representation for test results data allows for flexibility. The simulation programs can write as little or as much data as necessary. Most of this data typically consists of numeric values. The database facilitates simple analysis of the data. Values within a test and across many tests can be compared and contrasted via database queries.

All of the resources in all of the simulation activities can be used to generate output reports. This is not just output reports in the sense of simulation results but output reports that furnish a complete sense of what was attempted and accomplished. Generation of reports becomes a parsing and format interchange function of the content in the XML-based representations. Software is available to extract, parse, and format XML content. The output reports range in content and format from brief email messages to web-based reports, to formal papers in portable document format (PDF). The reports can contain portions of content from any and all of the resources. The reports can be automatically be created. This is one of the prime advantages of using XML. Figure 5 shows the resources available to the process of problem output.

4. RESULTS AND CONCLUSIONS

The simulation resources help the user with problem creation, simulation execution, and output recording and analysis. This eases the job and focus of the user. The user can re-focus attention to the problem or question at hand and the answers being generated rather than the job of making things work due to added complexity. The resources fit into a notion of "plug and play." A user can mix and match whatever portions of the resources are needed (this is the "plug" part). Then the user can run the problem and concentrate on the answers (this is the "play" part) and what they mean.

XML plays an important part in not only making the resources available but in making the resources work together all or in part. By using XML to create light-weight ontologies for representation a container and a language is created. The container holds the application specific knowledge being represented. The language is used to describe and manipulate the knowledge in the container. These representations are focused, small, and

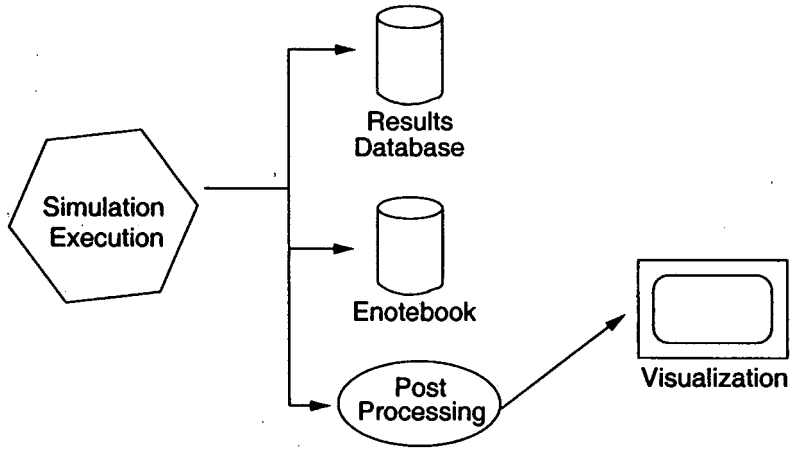

Figure 5. Shown are the resources available and the flow of the process of problem output.

light weight. They only have what is needed to get a single, specific application done. All these approaches lead to a principle of doing only "what is good enough" to get the job done. This contributes to streamlining and flexibility without loss of accuracy.

The XML-based representations can be changed and extended as the application domain grows or more representations can be added. Adding representations keeps the knowledge being represented more specific and light weight. It is still modular and can be loosely coupled to serve applications as necessary. In this manner the representations are like small building blocks that can be fit together for a custom purpose.

Using the types of resources described in this paper and in the manner described have lead to other benefits for simulation technology. By providing resources to users a single source for resources has resulted. This is more efficient than each user having their own copy of specific data such as material data. The single source is valid and approved data. It has been checked for accuracy and contains references describing its pedigree. Users are free to make changes to the data for their own purpose, but then the data is no longer approved and should not be propagated. This process is not only efficient but safe. Another benefit has been the evolution of standard practices. This has helped increase the quality of problem solving. It has also contributed to a higher quality of simulation testing.

REFERENCES

1. D. W. Forslund, C. A. Slocomb, and I. A. Agins, "Windows on computing - new initiatives at los alamos," *Los Alamos Science* (22), 1994.
2. D. Cooke and S. Hamilton, "New directions at NASA ames research center," *IEEE Computer* **33**, pp. 63–71, January 2000.
3. K. Skadron, M. Martonosi, D. I. August, M. D. Hill, D. J. Lilja, and V. S. Pai, "Challenges in computer architecture evaluation," *IEEE Computer* **36**, pp. 30–35, August 2003.
4. S. S. Mukherjee, S. V. Adve, T. Austin, J. Emer, and P. S. Magnusson, "Performance simulation tools," *IEEE Computer* **35**, pp. 38–39, February 2002.
5. W.W.W. Consortium, *Extensible Markup Language (XML) 1.0 (Second Edition), W3C Recommendation*, October 2000. http://www.w3.org/TR/REC-xml.
6. International Organization for Standardization, Geneva, *ISO 8879:1986 Information processing - Text and office systems - Standard Generalized Markup Language (SGML)*, October 1986.
7. S. J. Vaughan-Nichols, "XML raises concerns as it gains prominence," *IEEE Computer* **36**, pp. 14–16, May 2003.
8. M.-A. Grado-Caffaro and M. Grado-Caffaro, "The challenges that xml faces," *IEEE Computer* **34**, pp. 15–18, October 2001.

9. R. L. Kelsey, "XML-based representation," in *Proceedings of the International Conference on Artificial Intelligence 2001*, 2001 International MultiConference, July 2001.
10. R. L. Kelsey, K. R. Bisset, and R. B. Webster, "Automated parametric execution and documentation for large-scale simulations," in *Enabling Technology for Simulation Science V*, **4367**, pp. 202–208, Society of Photo-Optical Instrumentation Engineers, Society of Photo-Optical Instrumentation Engineers, (Bellingham, WA), 2001.
11. R. L. Kelsey, "An electronic notebook for physical system simulation," in *Enabling Technologies for Simulation Science VII*, **5091**, pp. 357–363, Society of Photo-Optical Instrumentation Engineers, Society of Photo-Optical Instrumentation Engineers, (Bellingham, WA), 2003.
12. R. L. Kelsey, R. T. Hartley, and R. B. Webster, "An object-based methodology for knowledge representation in SGML," in *Proceedings of the Ninth IEEE International Conference on Tools With Artificial Intelligence*, pp. 304–311, IEEE Computer Society, (Los Alamitos, CA), 1997.