SLAC-PUB-11918
October2006

THE ACCELERATOR MARKUP LANGUAGE AND THE UNIVERSAL ACCELERATOR PARSER *

D. Sagan, M. Forster, Cornell University, Ithaca, NY, 14850, USA
D. A. Bates, LBNL, Berkeley, California, 94720, USA
A. Wolski, University of Liverpool and the Cockcroft Institute, UK
F. Schmidt, CERN, Geneva, Switzerland
N. J. Walker, DESY, Hamburg, Germany
T. Larrieu, Y. Roblin, Jefferson Lab, Newport News, Virginia, USA
T. Pelaia, ORNL, Oak Ridge, Tennessee, USA
P. Tenenbaum, M. Woodley, SLAC, Menlo Park, California, USA
S. Reiche, UCLA, Los Angeles, California, USA

Abstract

A major obstacle to collaboration on accelerator projects has been the sharing of lattice description files between modeling codes. To address this problem, a lattice description format called Accelerator Markup Language (AML) has been created. AML is based upon the standard eXtensible Markup Language (XML) format; this provides the flexibility for AML to be easily extended to satisfy changing requirements. In conjunction with AML, a software library, called the Universal Accelerator Parser (UAP), is being developed to speed the integration of AML into any program. The UAP is structured to make it relatively straightforward (by giving appropriate specifications) to read and write lattice files in any format. This will allow programs that use the UAP code to read a variety of different file formats. Additionally, this will greatly simplify conversion of files from one format to another. Currently, besides AML, the UAP supports the MAD lattice format.

INTRODUCTION

New particle accelerator facilities are increasing in scale and complexity. This increase brings an ever greater need for global collaboration, and for accurate modeling of proposed designs to ensure technical performance and to manage costs. Various labs around the world have developed excellent accelerator modeling codes that could be used for such analysis. Using a variety of codes to analyze a single machine brings benefits because different codes tend to be optimized for different purposes. Additionally, cross-checking results between different codes is often essential for validating the results. However, different codes generally require input files in different formats, and this is a significant obstacle to collaboration. At present, there is no generally accepted lattice format that is sufficiently comprehensive to meet the needs of the community.

To address this problem, a lattice description format called Accelerator Markup Language (AML) has been created. AML is based upon the standard eXtensible Markup Language (XML) which provides the necessary flexibility for AML to be easily extended to changing requirements. Moreover, the extensibility of XML enables AML files to be used beyond lattice descriptions to include such information as the control system configuration, blueprint and other documentation, magnet history information, etc. In short, AML could be used as the basis for a complete database of an accelerator laboratory complex.

In conjunction with AML, a set of routines is being developed to simplify and speed the integration of AML into any program. These routines, collectively known as the Universal Accelerator Parser (UAP), will also provide the means to specify additional lattice file formats. This will allow programs that use the UAP code to read a variety of different file formats. Currently the UAP supports, besides AML, the MAD lattice format. The UAP will additionally be able to convert lattice files between any formats for which appropriate specifications are given.

ACCELERATOR MARKUP LANGUAGE

AML is an application of XML for describing accelerators. That is, XML defines, in a general way, the format for representing data in a file or within a program, and AML defines exactly what kind of data is actually present.

XML represents data in a hierarchical fashion. With AML, the root (top level) node in the hierarchy is the `<laboratory>` element, which contains not only the actual accelerator machine or machines but the entire laboratory complex. An example snippet is shown in Figure 1. The `<laboratory>` node contains the children (subnodes) that appear between the opening `<laboratory>` line and the ending `</laboratory>` line. In this example, there are two children; `<doc>` and `<machine>`. Nodes can contain text and/or subnodes. The `<doc>` node contains text but no children. The `<machine>` node contains the children `<beam>` and `<element>`, etc. In an actual accelerator there would be multiple `<element>` children defining all the different elements in the accelerator. The AML specifi-

* Work supported by the National Science Foundation and by the US Department of Energy under contract numbers DE-AC02-05CH11231 and DE-AC02-76SF00515.

```
<laboratory name = "Wilson Lab">
  <doc author = "NJW" date = "2005/01/22">
    This is a bit of documentation
  </doc>
  <machine name = "CESR">
    <beam>
      <energy value = "3.5e9" />
    </beam>
    <element name = "Q02W">
      <length value = "1.3" />
    </element>
    ... etc ...
  </machine>
</laboratory>
```

Figure 1: Brief example AML file.

<beambeam>	<match>
<bend>	<monitor>
<bend_sol_quad>	<multipole>
<custom>	<octupole>
<drift>	<patch>
<collimator>	<quadrupole>
<elseparator>	<rfcavity>
<knob>	<sextupole>
<i_beam>	<solenoid>
<instrument>	<sol_quad>
<kicker>	<taylor>
<lcavity>	<wiggler>
<marker>	

Table 1: Element types in AML.

cation defines the legal building blocks of an AML file. For example, the AML rules state that a <doc> node may be a child of a <laboratory> node. These rules are stored in an XML Schema file which can be used to validate an AML file. The AML specification allows multiple <machine> children to appear within the <laboratory> node, which enables an AML file to describe an entire accelerator complex, consisting of many different beamlines.

The fundamental advantage to using XML as the basis for describing accelerators lies in the fact that any XML application can be extended to include new information. For example, suppose that at a particular laboratory it is necessary to add information about blueprint numbers to the AML file for some machine elements. The AML Schema file can easily be extended so that <element> nodes are allowed to have a <blueprint> child. Significantly, such an addition will not break, say, an existing program that computes the Twiss parameters in the lattice, since the software routines within the program for reading in the AML file do not change with the addition of the <blueprint> child. Only the Schema file changes. In short, AML can be used as the basis for a database of information on all aspects of the accelerator.

AML defines a set of machine elements as shown in Table 1. Besides the standard set of elements — bends, quadrupoles, etc. — there are a number of additional elements, such as an <i_beam> element that models support structures. Machine element attributes and other parameters are grouped into sets. For example, an element may have a <multipole> attribute, that contains information on all the magnetic multipole components in the element.

Machine elements can "inherit" attributes from another element. Additionally, orientation errors can be specified separately from the design values. Consider the following example:

```
<element name = "Q03W" inherit = "Q0">
  <quadrupole>
    <orientation>
      <tilt value = "SKEW" />
    </orientation>
  </quadrupole>
  <orientation>
    <x_offset value = "0.02" />
  </orientation>
</element>
```

Here, the Q03W quadrupole inherits the attributes of the Q0 quadrupole. The <orientation> element within the <quadrupole> element defines the design values. The <orientation> element that is a child of the <element> node specifies an error offset of 0.02 meters.

AML uses SI (Systéme International) units for element attributes and other parameters. The only exception is that the unit of energy is eV. There is a standard set of defined mathematical and physical constants (π, c, etc.) and intrinsic functions (sin cos, etc.) Named constants may be defined whose values are declared using arithmetic expressions.

To specify the sequence of elements in a lattice, AML defines the <sector> node. Sectors may contain other sectors and/or elements, and can be defined with an argument list similar to a MAD line. The elements within a <sector> may be specified as a contiguous list of elements, or they may be placed by reference to some fiducial point similar to a MAD "sequence".

```
<sector name = "this_sect">
  <sector ref = "sect1" />
  <element ref = "Q01W" />
  <element ref = "d001" />
  <element ref = "DETa" at = "3.4"
                at_origin = "BEGINNING">
    <ref_point element = "Q01W"
                origin = "MIDDLE" />
  </element>
  ...
</sector>
```

In this example the Q02 element is "superimposed" on the line with the beginning point of the element 3.4 meters from the middle of the reference Q01W element.

Along with defining the syntax for describing an accel-

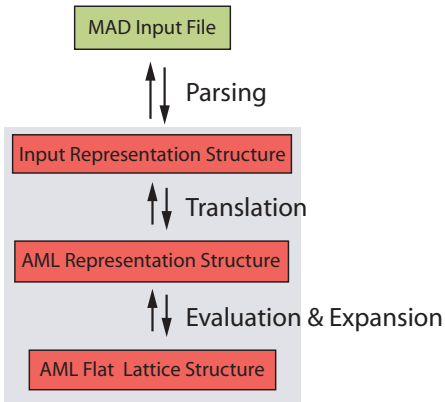

Figure 2: UAP stages.

erator, the AML standard defines enough of the physics so that the meaning of the various quantities that appear in an AML file are well defined. For example, AML defines what the sextupole <k2> strength actually means. It is important to note that the AML standard does not try to go beyond this and, in particular, AML says nothing about how physics calculations are to be made. It is up to an individual programmer to decide, for example, how to track a particle through the lattice.

UNIVERSAL ACCELERATOR PARSER

The Universal Accelerator Parser is a library for reading and translating between AML and different accelerator lattice formats. The UAP is written in C++ and compiles on most Unix, Linux, and Windows platforms. A Java port is maintained for platform independence. Software developers can easily integrate the library into existing code by using the provided hooks. The library provides the following services:

- An extensible framework for reading and translating between different lattice formats including the AML and MAD formats.
- Run-time modification and evaluation of parameters and beamline definitions.
- Expression evaluation and beamline expansion.

The analysis of a MAD format lattice file is illustrated in Figure 2. Starting from a MAD input file, the UAP software will create an "Input Representation Structure" (IRS) in computer memory. The IRS has a one-to-one correspondence with the information in the input file (including comments). From the IRS, the UAP provides software to translate from MAD syntax to AML syntax creating an "AML Representation Structure" (ARS) in memory. MAD lines become AML sectors, etc. From the ARS, the UAP provides software to evaluate all arithmetical expressions and to expand all the sectors to create the "AML Flat Lattice Structure" (AFLS). The AFLS provides an appropriate structure for the execution of physics analysis routines (such as those to perform tracking). Additionally, the UAP library has software to reverse the process to translate from an ARS to an IRS and from an IRS to generate a MAD lattice file.

The analysis of an AML file is similar to that of a MAD file, except that the IRS and ARS are identical structures – there is thus no translation stage. To convert from one file format to another (for example, from MAD to AML) an ARS is first constructed from the input file and then the ARS is used to create a file with the appropriate format.

To add a new file format to the UAP library, only the appropriate parsing and translation rules and their reverse counterparts must be created. The expression evaluation and sector expansion software is independent of the input file format. Once this software is created (which may be done semi-automatically using specialist tools), the conversion between this file format and existing formats known to the UAP does not require any additional coding.

The UAP has been designed to strike a balance between ease of use and extensibility. The library promotes rapid development of applications through high-level Application Programming Interfaces, as well as providing hooks to extend the UAP for custom programs.

CONCLUSION

The Accelerator Markup Language together with the Universal Accelerator Parser have been developed to facilitate the sharing of lattice descriptions among simulation programs. AML, through its use of XML, can be easily extended to provide the basis for a database of information on all aspects of an accelerator. The UAP, initially developed to handle AML and MAD formats, can easily be extended to handle additional formats. The use of a common library among accelerator codes will greatly simplify and speed the exchange of information.

A collaboration between a number of laboratories and institutions around the world has lead to the development of AML and UAP. The potential of AML/UAP to provide a common library for exchanging lattice information comes at the heels of large-scale collaborative projects, including the International Linear Collider and the Large Hadron Collider at CERN. These projects will increasingly rely on remote methods of collaboration, particularly remote computing. The AML/UAP stands at the forefront to handle the challenges of collaborating across formats, codes, and boundaries. All of the source code, AML documentation, and examples are freely available on the World Wide Web. The project homepage is at

`http://www.lns.cornell.edu/~dcs/aml/.`

ACKNOWLEDGMENT

Thanks must go to Richard Talman and Nikolay Malitsky for useful discussions.