CONFIGURABLE UAL-BASED MODELING ENGINE FOR ONLINE ACCELERATOR STUDIES

N. Malitsky, T. Satogata, BNL, Upton, NY 11973, USA,
R. Talman, Cornell University, Ithaca, NY 14853, USA

Abstract

The diversity of online accelerator modeling tasks poses a considerable challenge for accelerator physicists and software developers. Compromises between performance and completeness are always required. For example, modeling of chromatic effects requires second-order calculations, while speed favors first-order matrix multiplication. This paper presents our solution: a configurable computational modeling engine based on the UAL Accelerator Propagator Framework (APF). The choices of evolution algorithms are defined in an external Accelerator Propagator Description Format (APDF) file, permitting a flexible mechanism for employing different approaches within the context of machine studies and operation.

RATIONALE

Demands on online modeling diring commissioning and routine operation of an accelerator are fairly predictable. Often a single model (possibly only the design model) services the physicist's needs. However, more flexibility is often required while planning studies and a flexible, easily configurable model is most necessary during accelerator studies, to efficiently deal with improvisation and unanticipated beam conditions.

Previous approaches to online model flexibility[1] have wrapped model engines, ranging from simple to complex, in a single modeling environment. Though this unifies client and server layers for various models, reconfiguration of modeling layout of the underlying model engines requires architecture changes or recompilation.

The UAL framework[2] within the RHIC/AGS online model already incorporates most relevant physics, and UAL is easily extensible. We have therefore designed and implemented a flexible configuration format for UAL, to allow dynamic specification of modeling propagators and to permit evaluation of tradeoffs between speed and completeness in online modeling applications for machine studies.

ACCELERATOR PROPAGATOR FRAMEWORK

The Accelerator Propagator Framework (APF) is the next logical step of the UAL evolution. APF aims to provide a consistent mechanism for building configurable accelerator modeling engines. The first phase of the UAL infrastructure was associated with development of the Standard Machine Format (SMF) which introduced a compact and generic accelerator model for implementing various accelerator structures. The organization of SMF is exhibited graphically in Fig. 1.

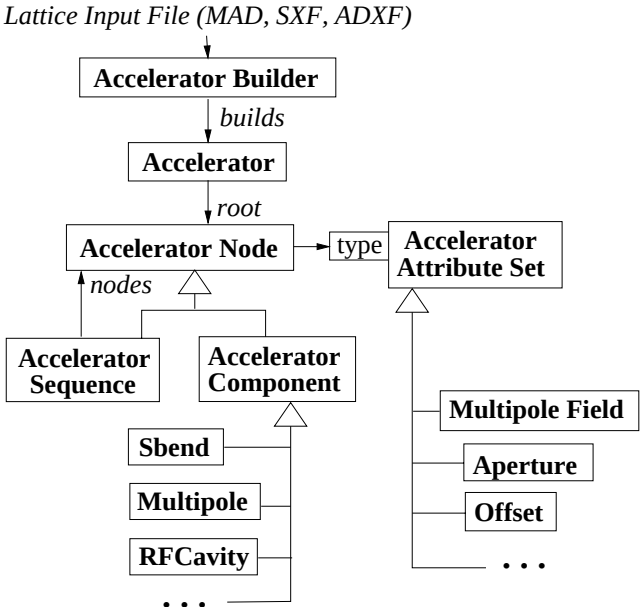

Figure 1: Standard Machine Format object model.

SMF, the result of analysis and generalization of several accelerator modeling formats, has been successfully tested by several accelerator projects. Its data structures have also been optimized from different implementation perspectives, facilitating the employment of formal methodologies for mapping SMF objects into various physical representations, such as Java classes, XML, relational databases tables, and GUI components. The second part of the UAL infrastructure, the Accelerator Propagator Framework (APF), has been based on proven SMF design patterns. Fig. 2 shows the APF model.

In APF, *Accelerator Propagator* replicates the hierarchical organization of the Accelerator structure with one important distinction. Each *Propagator Nod*e may be associated with an entire accelerator sector identified by begin and end *Accelerator Nodes.* This scheme allows us to accommodate most accelerator modeling algorithms and to bridge the gap between element-by-element and map-based approaches. The structure of the configurable Accelerator propagator is described in the Accelerator Propagator Description Format (APDF). Then one can consider the APDF file as a complement to the MAD and SXF lattice input files.

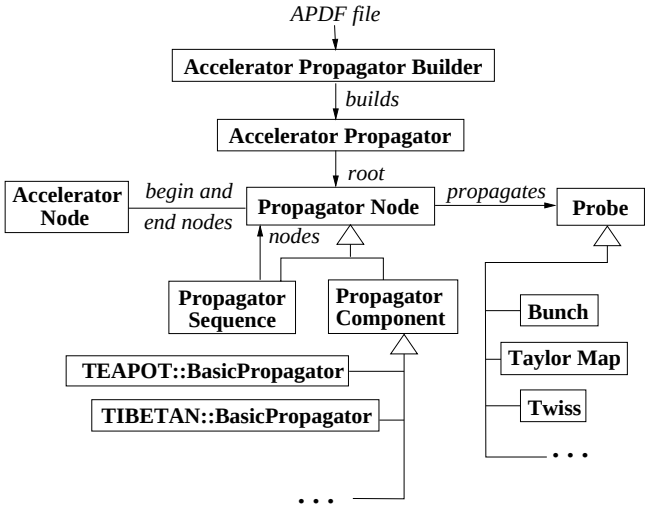

Figure 2: Accelerator Propagator Framework

APDF SPECIFICATION

The structure of the Accelerator Description Format (APDF) is determined by a single conceptual statement: *Accelerator Propagator is a sequence of links between accelerator nodes and corresponding accelerator algorithms.* This statement introduces two entities, *Accelerator Propagator* and *Link*, and their relationship. According to the formal methodologies, like the lattice description, this model should be naturally represented by an ordered list of all elements with their element-algorithm associations. However, for the sake of brevity, we favor a command-oriented SQL-like approach that allows one to define many associations in a very compact way. The following sections describe the structure and semantics of the APDF elements.

APDF Propagator

Accelerator Propagator is a sequence of element-algorithm links that can be created, inserted, and deleted in the APDF file:

```
<!ELEMENT propagator (create|insert|delete) >
<!ELEMENT create (link)*>
<!ELEMENT insert (link)*>
<!ELEMENT delete (link)*>
```

The ⟨create⟩ procedure builds a propagator instance from scratch. The ⟨insert⟩ and ⟨delete⟩ elements aim to facilitate user's extensions of some common and sharable accelerator propagator version.

APDF Link

The APDF link, the key element of the APDF specification, provides the mechanism for associating a selected family of accelerator nodes with appropriate accelerator algorithms:

```
<!ELEMENT link>
<!ATTLIST link
   algorithm CDATA #REQUIRED
   sector    CDATA #IMPLIED
   elements  CDATA #IMPLIED
   types     CDATA #IMPLIED
>
```

The following table contains a brief description of link attributes:

```
sector    a pair of begin and end accelerator
          element design names, e.g. "d1, qf1";
          (sector includes d1 but not qf1)
elements  a regular expression for selecting
          accelerator nodes with specified design
          names, e.g. "q1|q2"
types     a regular expression for selecting accel-
          erator nodes with specified element types,
          e.g. "Quadrupole|Sextupole"
algorithm a full class name of the associated prop-
          agator, e.g. "TEAPOT::MltTracker"
```

Sector, elements, and types define three different approaches for selecting families of accelerator elements. In case of overlapping, sector-based priority is first, element-based priority is second, and type-based priority is third.

APPLICATIONS

This approach addresses the spectrum of applications ranging from small special tasks to full-scale realistic beam dynamics studies encompassing heterogeneous algorithms and special effects. This will be illustrated by several models that exhibit different features of APDF description.

Longitudinal Tracker

A traditional accelerator longitudinal model is usually represented by a combination of 2D sector matrices and RF cavity nodes. In the APF framework, this structure can be described by the following (complete) APDF file:

```
<apdf>
  <propagator>
    <create>
      <link algorithm="TIBETAN::SectorTracker"
            sector= "Default" />
      <link algorithm= "TIBETAN::RFCavityTracker"
            elements="rfac1" />
      <link algorithm="TIBETAN::WCMonitor"
            elements="mend" />
    </create>
  </propagator>
</apdf>
```

In contrast with traditional programs, linear coefficients and nonlinear momentum-compaction factors of TIBETAN *SectorTracker* are not part of the new input file; they are calculated during an initialization phase from a common lattice description (such as MAD or SXF input file) shared by other modeling engines. This approach facilitates the synchronization of input parameters among various project

applications and the consistent transition between different simulation scenarios. In addition to longitudinal tracking algorithms, the above model is also able to accommodate a diagnostics device, Wall Current Monitor, which accumulates beam profiles after each turn and forwards them to an intermediate buffer. The extensibility of the APDF-based longitudinal tracker also permits the addition of longitudinal space charge and various impedance effects as associations of corresponding algorithms with arbitrary elements or markers of the accelerator lattice.

Conventional Element-by-Element Tracker

In traditional accelerator codes, tracking algorithms have been uniquely associated with particular element types. This functionality can, of course, be replicated by APDF. The following file represents the original TEAPOT element-by-element tracking engine:[1]

```
...
<create>
  <link algorithm="TEAPOT::DriftTracker"
        types= "Default" />
  <link algorithm="TEAPOT::DriftTracker"
        types="Marker|Drift|[VH]monitor|Monitor" />
  <link algorithm="TEAPOT::DipoleTracker"
        types="SBend" />
  <link algorithm="TEAPOT::MltTracker"
        types="Quadrupole|Sectupole|Multipole|
               [VH]kicker|Kicker" />
  <link algorithm="TEAPOT::RfCavityTracker"
        types="RfCavity" />
</create>
...
```

As well as preserving this traditional scheme, the APDF approach provides a consistent mechanism for combining algorithms from different UAL modules and for adding application-specific extensions. The following APDF file describes an updated model for collecting turn-by-turn BPM data in the Model Independent Analysis (MIA) application:

```
...
<create>
  <link algorithm="TEAPOT::DriftTracker"
        types= "Default" />
  <link algorithm="TEAPOT::DriftTracker"
        types="Marker|Drift|[VH]monitor" />
  <link algorithm="TEAPOT::DipoleTracker"
        types="SBend" />
  <link algorithm="TEAPOT::MltTracker"
        types="Quadrupole|Sextupole|Multipole|
               [VH]kicker|Kicker" />
  <link algorithm="TIBETAN::RfCavityTracker"
        types="RfCavity" />
  <link algorithm="MIA::BPM"
               types="Monitor" />
</create>
...
```

[1]For brevity, beginning and ending "apdf" and "propagate" tags are suppressed from the remaining examples.

In this example, MIA::BPM is a temporary application-oriented class that collects data and writes them into a global container analyzed by the MIA post-processing tool.

Fast TEAPOT

Chromatic effects are a typical accelerator feature modeled by many conventional element-by-element and differential algebra-based accelerator codes. However, the power of all-purpose accelerator programs significantly diminishes their computation speed, tending to make them unacceptable for online applications. The APDF fine-grained selection mechanism allows us to build "custom-made" models from a combination of sector linear matrices and TEAPOT symplectic integrators for chromatic elements such as the main quadrupole and sextupole elements—the so-called Fast TEAPOT:

```
...
<create>
  <link algorithm="TEAPOT::MatrixTracker"
        sector= "Default" />
  <link algorithm="TEAPOT::MltTracker"
        types="Quadrupole|Sextupole" />
</create>
...
```

The same approach can be applied to other applications for studying localized dominant effects (for example, interaction regions) or combining traditional algorithms with application-oriented, efficient approximations.

ONLINE MODELING

The RHIC/AGS model server provides CDEV access and namespace support for multiple concurrent online models. With the APDF specification in UAL, these online model instances are uniquely specified by the SMF lattice and the APDF algorithm specification. Work is in progress to extend the CDEV access to permit dynamic model instantiation with user-specified APDF files customized for specific beam studies, such as nonlinear dynamics and roll of interaction region triplets. Standard operations applications can then retrieve optics and lattice information from these customized models for study planning and feedback.

Though flexibility is paramount in an offline modeling environment where hypotheses are constantly being tested and reinterpreted, model stability is also important for machine operations and reproducibility. There is therefore also a canonical operations model that is the default source of all application modeling data for RHIC, and the APDF for this model is under strict configuration control.

REFERENCES

[1] T.Satogata, et al. The RHIC Online Model Environment: Design and Overview, PAC 99.

[2] See http://www.ual.bnl.gov.