THE RHIC/AGS ONLINE MODEL ENVIRONMENT: DESIGN AND OVERVIEW *

T. Satogata†, K. Brown, F. Pilat, A. Alai Tafti, S. Tepikian, J. van Zeijts
BNL, Upton, NY, USA 11973

Abstract

An integrated online modeling environment is currently under development for use by AGS and RHIC physicists and commissioners. This environment combines the modeling efforts of both groups in a CDEV[1] client-server design, providing access to expected machine optics and physics parameters based on live and design machine settings. An abstract modeling interface has been designed as a set of adapters[2] around core computational modeling engines such as MAD and UAL/Teapot^{++}[3]. This approach allows us to leverage existing survey, lattice, and magnet infrastructure, as well as easily incorporate new model engine developments. This paper describes the architecture of the RHIC/AGS modeling environment, including the application interface through CDEV and general tools for graphical interaction with the model using Tcl/Tk. Separate papers at this conference address the specifics of implementation and modeling experience for AGS and RHIC.

1 MOTIVATION AND SCOPE

Over the past five years, an infrastructure has been developed in the RHIC project that integrates delivered magnet measurements, offline long-term particle tracking, and survey of installed RHIC components. Both design and `as built' optics models of RHIC are routinely produced. Accelerator applications being developed for commissioning require a consistent optics model framework that builds upon this effort, maintaining consistency from design through construction and installation to commissioning. However, the tracking and optics programs used for design and magnet production feedback could not be easily adapted for use by RHIC controls applications.

We have developed an online modeling environment used by various RHIC correction applications (e.g. orbit, tune, chromaticity, coupling) to access design, as-built, and live optics data generated by optics model engines. By using a client-server model and an abstracted modeling interface layer, this becomes a generic modeling environment that can also be used for AGS and AGS/Booster application modeling and calculation. Multiple model engines with different interfaces and implementations are supported by a common CDEV interface for application use.

The online model design discussed here is not intended to supply a full control-system simulation of an operational accelerator control system (cf. Fermilab's Open Access Server). At this time there are no plans to implement such a simulation service for RHIC.

* Work supported under the auspices of the U.S. Department of Energy
† Email: satogata@bnl.gov

1. Read/write flat machine lattice descriptions
2. Read/set individual magnet parameters:
   - Strength (including combined function magnets)
   - Offset from design (2D)
   - Multipole corrections and errors
   - Survey and layout
3. Read/Set boundary conditions
   - Single-particle initial coords, energy, species
   - Bunch initial conditions (groups of particles)
   - Beamline initial lattice functions
4. Calculate optics to reasonable order:
   - Tunes (Q_x, Q_y, Q_S)
   - Transition energy, γ_T
   - Lattice functions (β, α, phases, η,...)
   - 6D orbit, football transfer matrices
   - First-turn and closed orbits

... Future development, specialty CMEs
   - Perform constrained model-based corrections
   - Produce Taylor Expansions, maps, DA forms
   - Perform single-particle and bunch tracking

Table 1: A summary of CME computational capabilities in the RHIC/AGS Online Model Environment.

2 COMPUTATIONAL MODEL ENGINES

A computational modeling engine (CME) is an accelerator simulation that provides an interactive interface (usually an interpreted script) to a small set of modeling capabilities. These CMEs are the core of any accelerator modeling, online or offline — they are the algorithmic guts and interfaces that transform lattice and beam definitions into beam physics output. UAL/Teapot^{++}is the CME currently in use for RHIC design and commissioning. Other commonly-used CMEs include Teapot, TRANSPORT, SYNCH, COSY, and MAD, and several locally-modified versions of MAD are also used and maintained by AGS beam physicists [4].

Online modeling requires interactive, real-time CMEs. Most popular CMEs are used by accelerator designers and modelers in a batch job mode, driven by command scripts written in highly idiomatic command languages, and parsed and interpreted line by line. Though the script interface can also be used interactively, optics output is usually only output to files, an inefficient path for application interaction with a CME. The online modeling architecture described here abstracts the common features of many CMEs into a simple network command interface that can

be used by accelerator applications on distributed controls consoles.

CMEs load accelerator lattice definitions using another idiomatic language, though there is much more commonality here as many accelerator codes use the the MAD input language, Standard Machine Format (SMF). However, SMF is cumbersome or deficient in some areas required to model a fully hierarchical design model of an accelerator, modified by `as-built' constraints. Recent collaborative efforts between several labs have made progress towards developing SXF, a Standard Exchange Format, to supersede SMF for collaborative LHC design work [5]. The RHIC/AGS modeling environment provides an abstract interface to lattice and strength table read/write, allowing adaptors to be easily written that integrate local lattice databases and definitions with various CMEs in the online model.

Functional requirements for CMEs are derived from application and commissioning priorities, as well as commonality of existing CME functionalities. Experience has shown that the most common requests to an online CME during commissioning are those shown in Table 1. In particular, one must be able to change magnet strengths and offsets, and calculate full 6D linear optics parameters for use in control application analysis and correction. Many CMEs are capable of these calculations with very similar interfaces, though they vary wildly in their implementations and compromises between speed and completeness.

3 SERVER ARCHITECTURE

The client-server architecture for RHIC/AGS online modeling is shown in Figure 1. Client applications interact with the model via CDEV calls, as described in Section 4. Each model server for a supported CME is compiled from several C^{++}classes. The CDEV modeling interface is provided by a CDEV Model Server class that is derived from the CDEV Generic Server[6]; derived servers may extend the interface to provide extended access to underlying CME capabilities and data structures.

3.1 Generic model data classes

The model server uses a small set of generic model data classes to provide the data interface between optics and magnet settings in the CME and the model server class. This supports the CME capabilities in Table 1, and includes arrays of lattice functions at user-specified monitor elements, as well as matrices for higher-order optics. All model data follows Teapot unit and coordinate conventions, for initial implementation convenience. However, the current strongly-typed model data class is not dynamically extensible to accommodate different CME data structures, and this data class will be reimplemented as a generic cdevData container extension in the near future.

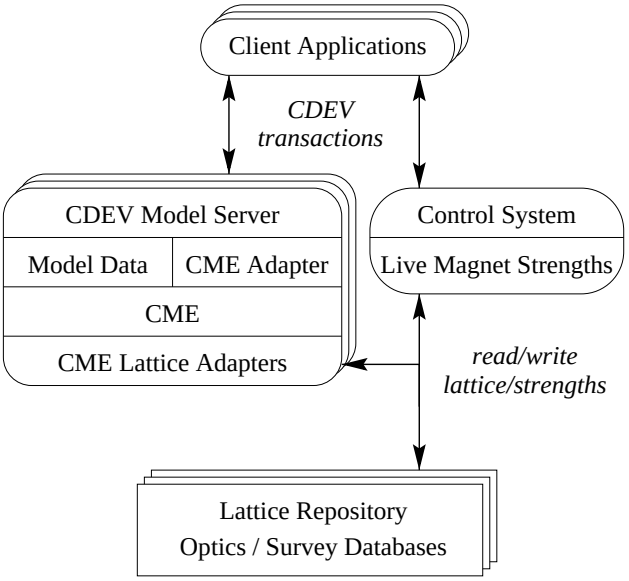

Figure 1: The RHIC/AGS online model architecture. Each running model server is a separate unix process, binding the server interface to an instance of a CME.

3.2 CME Adapters

Each CME must have an adapter class that translates generic method calls from the server to CME-specific function calls or implementations. Writing such an adapter is the major effort required to integrate a new CME into this environment. All CME adapters are derived from a parent interface class; this general interface is then used by the model server when making requests, and the model data classes are used to retrieve optics and lattice output.

CME translation and interface becomes complicated when the CME only has a parsed scripting interface that is not easily bindable (e.g. AGS/MAD, or MAD with no source libraries). In these cases the CME adapter must fork a separate process instance of the CME and interact with it via file descriptors, generating command text and parsing output files as requests are made. Other CMEs (such as UAL/Teapot^{++}) that have direct C^{++}object interfaces may be used directly in the CME via shared libraries that are loaded at run-time.

3.3 Lattice adapters

Lattice adapters provide translations from one lattice representation to another within a given environment, such as a control system or accelerator design project. They separately implement read/write of accelerator lattices and survey (slowly-changing or static layout), and magnet strength (dynamic control) information. In particular, they can also implement methods to load live accelerator magnet settings from the control system into the model, allowing interactive online comparisons between live and expected machine optics.

1. Names and Strengths:
   - SiteWideNames
   - ModelStrength: in physics units
2. Model Output:
   - LatticeFunctions
   - Orbit
   - muX, muY: tunes for circular lattices
   - chromX, chromY: chromaticities
   - gammatransition
3. Response Matrices:
   - SteeringMatrix: for beamline steering
   - ClosedOrbitMatrix: for closed orbit correction
   - OscillationMatrix: for coherent oscillation correction
   - MatchingMatrix: for betatron matching
   - TuneMatrix: for tune correction
   - GammaTransitionMatrix: for γ_t correction

Table 2: Model keywords for requests to the CME server

4 CDEV CLIENT INTERFACE

The client side of the CME server consists of several CDEV `device' classes, and specific attributes for each class. The attributes for a model device are summarized in Table 2. The message interface allows access to the model by using `get' and `set' verbs, or notification on-change by using the `monitorOn' verb, where callbacks will be triggered when the underlying data changes. This notification allows multiple clients to stay synchronized when magnet parameters or injection parameters are changed within the model.

For retrieving response matrices we have followed the naming conventions used in the `BeamOptics' code [7]; in Table 2 we list the requests for several types of these matrices. Each request takes appropriate lists of magnet names and position pickups, which are then sent as tagged entries in cdevData interface to the model. Optionally, the outgoing context data specifies an interest in a subset of the default return data; for instance, this allows a lattice function call to only return β_x and α_x instead of the full set.

5 APPLICATION EXAMPLES

Currently several applications at RHIC use the interface described here to access the UAL/Teapot^{++}CME. They include the RHIC orbit correction application, the RHIC injection application[8], the ATR emittance measurement application, the RHIC Ramp Editor[9], and several Tcl/Tk scripts that are used for lattice function visualization, and what-if scenarios. As an example we show in fig. 2 the interface to the RHIC injection application, displaying a CME-derived orbit.

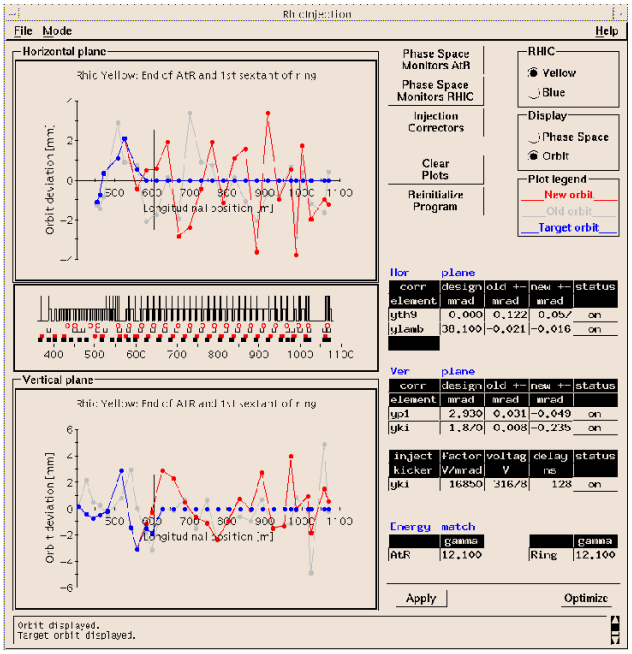

Figure 2: The RHIC injection application.

6 ACKNOWLEDGEMENTS

Our thanks to Nikolay Malitsky and Dick Talman for their development of the UAL/Teapot^{++}modeling environment that inspired this work. Future developments of this environment may extend to restructuring the client-server model around CORBA, providing a more flexible and dynamic binding between modeling components and adapters.

7 REFERENCES

[1] J. Chen, et al., "CDEV: An Object-Oriented Class Library for Developing Device Control Applications", Proceedings of ICALEPCS 1995.

[2] E. Gamma, et al., *Design Patterns.* (Addison-Wesley, Reading, Massachusetts, 1995)

[3] N. Malitsky and R. Talman, "The UAL Infrastructure for Accelerator Optimization and Correction", these proceedings.

[4] K. Brown, et al., "The RHIC/AGS Online Model Environments: Experiences and Design for AGS Modeling", these proceedings.

[5] F. Pilat, et al., "The application of the SXF lattice description and the UAL software environment to the analysis of the LHC", these proceedings.

[6] W. Akers, "An Object-Oriented Framework for Client/Server Applications", Proceedings of ICALEPCS 1997.

[7] B. Autin, "BeamOptics: A Program For Analytical Beam Optics", CERN publication 98-06.

[8] W. Fischer, J.W. Glenn, W.W. Mackay, V. Ptitsin, T.G. Robinson, N. Tsoupas, "The RHIC Injection System", these proceedings.

[9] J. Kewisch, J. van Zeijts, S. Peggs, T. Satogata, "Ramp Management in RHIC", these proceedings.