GENERIC MODEL HOST SYSTEM DESIGN*

P. Chu#, J. Wu, SLAC, Menlo Park, CA 94025, U.S.A.
G. Shen, BNL, Upton, NY 11973, U.S.A.
J. Qiang, LBNL, Berkley, CA 94704, U.S.A.

Abstract

There are many simulation codes for accelerator modelling; each one has some strength but not all. A platform which can host multiple modelling tools would be ideal for various purposes. The model platform along with infrastructure support can be used not only for online applications but also for offline purposes. Collaboration is formed for the effort of providing such a platform. In order to achieve such a platform, a set of common physics data structure has to be set. Application Programming Interface (API) for physics applications should also be defined within a model data provider. A preliminary platform design and prototype is discussed.

INTRODUCTION

For modern accelerators, online model is necessary for beam control and physics studies. However, usually simple online model is not sufficient for many physics problems. For applications based on real-time model data, they have to run model inline, which can take tens of seconds for going through machine data acquisition, model computation and result update. Also, there are many other simulation codes which may provide better accelerator modelling than the online model in use. Among the existing modelling codes, each one has some strength but not all. An idea is to provide a platform to host multiple modelling tools so one can switch among the codes easily. A middle layer in between any model program and user application can provide transparent access to the model data. Additionally, the software infrastructure for such model platform should be extremely robust. User applications can be benefited from simple model access and control Application Programming Interface (API). To make such platform useful for beam control, the performance should be greatly improved for many beam dynamics codes.

The model codes are the engine for beam computation. The supporting infrastructure, which prepares model input parameters, sets up a model run and packages model data, is called *Model Engine*. The Model Engine then serves up model data via some commonly used communication protocol to its client applications.

ARCHITECTURE OVERVIEW

To integrate various model codes within a common platform, data format conversion between each individual code and the platform data structure is an essential component of the software architecture. Model run control for scheduling and managing multiple runs is another crucial component.

In Fig.1, a schematic diagram shows the data flow for the model engine and its automated data update service. Detail for each component is explained here:

- Model input data will be obtained from relational database (RDB) for static information such as device location and names, from control system for lattice configuration such as magnetic fields and accelerating cavity settings. Beam parameters at the beginning of tracking such as phase space coordinates and Twiss parameters should be read from diagnostics data provider for extant machine model or from RDB for design model.
- The model input data is then formatted in a common data structure. For each specific modelling code, there will be a data adaptor to convert the common data format to the corresponding code input format.

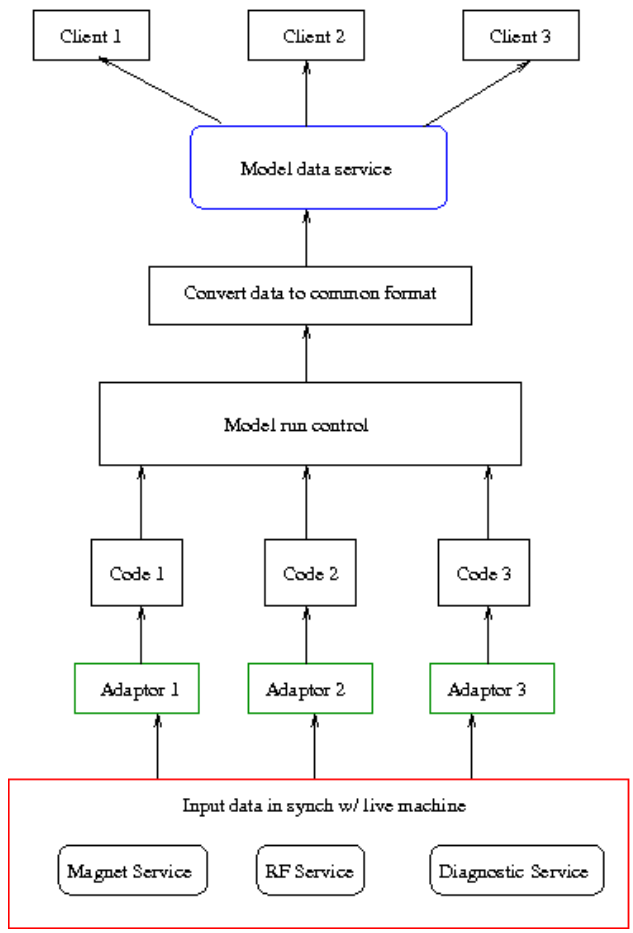

Figure 1: Data flow for model engine and service.

*Work supported in part by the DOE Contract DE-AC02-76SF00515.
#pchu@slac.stanford.edu

- A model run-control program can then schedule a model run and monitor the run status. This model run-control program should provide user-friendly interface for changing physics parameters. A couple of run-control prototypes have been tested.
- After a model run, the model data are then converted from specific model code output data format to a common data structure.
- The final structured model data are posted to a model service provider which can then serve up the data to any client.
- Physics applications can access the final data via simple API over the network.

MODEL ENGINE

Model engine is a platform capable of running some predefined model codes. Previously, some work has been done for start-to-end simulation [1, 2]. The work includes run-control program and easy user interface for physics parameter editing.

Model engine should consider the following items:

- A common model data structure. XAL [3-5] accelerator hierarchy format can provide common input data structure. The output data should be similar to the XAL model data object which is capable of describing a modelled beamline.
- Model data adaptor. Each specific code needs a conversion between the common data structure and its own format for both input and output. The complexity level for each adaptor varies from code to code.
- Model input data storage. File or RDB can be the input data storage. Initially, model seed parameters can be saved as files. Each model run scenario will have a corresponding directory for saving both input and output files.
- Model output data storage. RDB and memory will be the primary storage for better performance and management reason. Files can be an optional choice. Because the complexity of many model runs, it is preferable to store the model data in RDB. For quick deployment, file I/O should be support as well.

Model Codes

In the following, we will use LINAC Coherent Light Source (LCLS) as an example to elaborate the necessity of having this model server concept. LCLS is an electron LINAC machine; therefore, the discussion will mostly concentrate on LINAC related issues. Additionally, some circular machine subjects will be mentioned as well.

During the LCLS commissioning and also current operation, XAL has contributed majorly in setting up the machine model and acted as the platform for high-level-applications; yet due to some new features of the LCLS machine, there is need to improve the model itself as well as improving its speed and bringing it online.

With such a model engine, the beam dynamics can be simulated with real machine parameters and compared to real measurement data. For now, there is no single code which can serve this purpose. The popular approach is to invoke IMPACT for injector simulation, Elegant for LINAC/accelerator simulation, and GENESIS for Free Electron Laser (FEL) simulation. In the following, we give some details of these codes.

- XAL– envelope tracking optics code.
- IMPACT-T [6]—3D multi-particle tracking code capable of simulating electrons emitting from cathode, and contains full space-charge effect, 1-D Coherent Synchrotron Radiation (CSR) effect, and wake fields in the accelerating cavities. It is a parallel code and can also run on single processor.
- Elegant [7] – 3D multi-particle tracking code containing 1-D CSR effect, simplified mode for space charge effect, wake fields in the accelerator cavities, and some scattering process like electrons going through the foil. . It is a parallel code and can also run on single processor. The Elegant code can be used also for an electron storage ring, but we will not explore the details of its function for ring here
- GENESIS [8] —3D FEL time-dependent code devoted to simulate beam dynamics and FEL radiation. It is a parallel code and also can run on single processor.
- Tracy [9] - 3D tracking code including the realistic fringe field. It can be used for an electronic storage ring, injection straight section, and a complex facility such as RICH.

Among the above mentioned codes, three of them (IMPACT-T, Elegant, and GENESIS) are serving the start-to-end (S-2-E) simulation for LCLS, which is a self-amplified spontaneous emission (SASE) FEL.

Due to the fact that the above mentioned three codes are independently developed, data flow and user interface are the bottle neck for speeding up the simulation time. The idea of generic model host system is an extension of the S-2-E set up. The new platform will not only provide S-2-E functionality but also quasi-online capability.

Furthermore, some code improvement work is under consideration. In particular, the FEL itself will be manipulated. For example, in self-seeding FEL scheme, the SASE FEL generated from the first undulator will be purified spectrally; hence, X-ray optics code has to be integrated with FEL code. These codes once ready, will be integrated into the model engine picture.

Computer Cluster

Optionally, many multi-particle beam dynamics simulation codes can be run on computer clusters for good performance. As emphasized above, speeding up the model calculation is important and therefore, parallel computation can help.

Yet another reason is that such a model engine should serve multiple clients for multiple tasks. Data management and definition of read-only and writable data has to be carefully handled in such multi-task scenario.

In principle, the prototype of such a model engine can be built upon single PC concept; yet, we prefer to mention this computer cluster here to emphasize its potential importance as explored above. Whenever we want to speed up the process, parallel capability is always important.

MODEL SERVICE

On top of the Model engine, there should be a reliable and seamless mechanism for serving up model data to any subscribed client applications.

Main concern for the model service technology selection is the communication protocol. Because the model data can be complicated structure, simple communication protocol cannot satisfy such need. There are several possible candidates for the model service. Evaluation and prototyping with these protocols is underway. The communication protocol choice also depends on the performance, reliability and easy-to-use. There could be many clients accessing the same model data concurrently, therefore, the performance and robustness should be a great concern. On the client side, we intend to support Java, Matlab and Python-based programs.

Model Data

Typically, beam model data should be updated periodically. For example,

- Twiss Parameters
- R Matrices
- RMS phase space coordinates

CONCLUSION

Model server prototype study is underway [10]. Present work is focused on existing tools integration such as using one model code with simple run-control program. Model output data will be passed to a service program using a new EPICS Channel Access based program. A simple client program such as a beta function display will then consume the model data. The prototype work is expected to finish within a couple of months.

REFERENCES

[1] M. Borland *et al*, "Start-to-End Simulation of Self-Amplified Spontaneous Emission Free-Electron Lasers from the Gun through the Undulator", NIM A 483 (2002) 268-272.

[2] C. Larrieu, private communication.

[3] http://sourceforge.net/projects/xaldev/.

[4] https://wiki.ornl.gov/sites/xaldocs/default.aspx.

[5] J. Galambos, *et al*, "XAL Application Programming Structure," p. 79, Proceedings of 2005 Particle Accelerator Conference.

[6] J. Qiang, S. Lidia, R. D. Ryne, and C. Limborg-Deprey, "A Three-Dimensional Quasi-Static Model for High Brightnees Beam Dynamics simulation," Phys. Rev. ST Accel. Beams, vol 9, 044204 (2006).

[7] M. Borland, "elegant: A Flexible SDDS-Compliant Code for Accelerator Simulation," APS LS-287, 2000.

[8] S. Reiche, ``GENESIS 1.3: a fully 3D time-dependent FEL simulation code", Nucl. Instrum. Methods Phys. Res., Sect. A **429**, 243 (1999).

[9] J. Bengtsson, "TRACY-2 User's Manual", SLS Internal Document, February 1997; M. Böge, "Update on TRACY-2 Documentation", SLS Internal Note, SLS-TME-TA-1999-0002, June 1999.

[10] G. Shen *et al*, "Design of Accelerator Online Simulator Server Using Structured Data", these proceedings.