Lawrence Berkeley National Laboratory
Recent Work

**Title**
Dynamic Accelerator Modeling

**Permalink**
[https://escholarship.org/uc/item/9nf7f0m0](https://escholarship.org/uc/item/9nf7f0m0)

**Author**
Nishimura, H.

**Publication Date**
1993-05-17
Dynamic Accelerator Modeling

H. Nishimura

May 1993
DISCLAIMER

This document was prepared as an account of work sponsored by the United States Government. Neither the United States Government nor any agency thereof, nor The Regents of the University of California, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by its trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof, or The Regents of the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof or The Regents of the University of California and shall not be used for advertising or product endorsement purposes.

Lawrence Berkeley Laboratory is an equal opportunity employer.
DISCLAIMER

This document was prepared as an account of work sponsored by the United States Government. While this document is believed to contain correct information, neither the United States Government nor any agency thereof, nor the Regents of the University of California, nor any of their employees, makes any warranty, express or implied, or assumes any legal responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by its trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof, or the Regents of the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof or the Regents of the University of California.
DYNAMIC ACCELERATOR MODELING

HIROSHI NISHIMURA

ACCELERATOR and FUSION RESEARCH DIVISION
Lawrence Berkeley Laboratory
University of California
Berkeley, CA 94720

MAY 1993

This work was supported by the Director, Office of Energy Research, Office of Basic Energy Sciences, Materials Sciences Division, of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098.
Dynamic Accelerator Modeling*

H. Nishimura
Lawrence Berkeley Laboratory, University of California, Berkeley, California 94720

Abstract
Object-Oriented Programming has been used extensively to model the LBL Advanced Light Source 1.5 GeV electron storage ring. This paper is on the present status of the class library construction with emphasis on a dynamic modeling.

I. INTRODUCTION
The Advanced Light Source (ALS) at Lawrence Berkeley Laboratory represents one of the new generation of electron storage rings being developed for high brightness synchrotron radiation experimentation [1]. These low emittance storage rings require high accuracy, multi-parameter accelerator models for trajectory calculations and model-based control systems. During the lattice design phase, computer-intensive off-line modeling and simulation programs were developed to study magnet structures and tolerances [2]. The models were later made more flexible and interactive by taking advantage of Object-Oriented Programming (OOP) languages and techniques [3]. This paper describes the next logical step to integrate the modeling software with the accelerator control system in order to provide model-based control and automated analysis of the accelerator.

II. DYNAMIC MODELING AND OOP
A. Dynamic Modeling
Traditionally, a tracking or modeling code supports only one accelerator configuration and it is tightly coupled to a particular approximated Hamiltonian and its integrator. Dynamic Modeling [4] is a new modeling technique that supports multiple accelerator configurations at run time. It also isolates the simulation code framework from the detail of its numerical integrator. These tasks can be supported quite naturally using an OOP concept. The general software requirements that include correctness, robustness, extendibility, reusability and compatibility [3] are all supported by OOP. Additional specific requirements for the accelerator control system described in the reference [5] also requires OOP.

Dynamic Modeling is just one of the merits we can get from OOP. Instead of creating a virtual accelerator using all the lines of a code, we can construct a class of accelerators and create, manipulate and annihilate multiple virtual accelerators at run time. It makes the calibration of the model efficient because virtual machines behave like dynamic variables. It also makes the modeling of operations with undulators easier by keeping many configurations with different undulator settings.

B. Class Libraries
Our effort has been focused on the development of class libraries which serve as building blocks of various kinds of applications. There are three kinds of class libraries: modeling, hardware access and applications. A class for modeling and simulation is called Goemon [6] and supports Dynamic Modeling.

C. Modeling on the ALS Control System
The programs we used in the lattice design phase [2] were developed on VAX/VMS and written in VMS Pascal. The first step to create Goemon was to extract a linear modeling engine from them. Then it was rewritten in ANSI C for the use on Unix workstations and IBM PC clones that have online access to the hardware of the accelerator [7]. We used Eiffel (v2.3) [8] on Unix to construct a class library at a very high level, keeping the numerical engine in C [4]. Now it has been completely rewritten in C++ on PC clones running Microsoft Windows 3.1 or NT and is being ported to Unix. This version covers the range from the lower level numerical engine to the higher level optics calculations and fittings.

D. Design and Analysis
The Object-Oriented Approach was applied not only for programming but also for design and analysis. We used the Object Modeling Technique [9] with OMTool [10] for object design and analysis. In this notation, a class is represented as shown. (We may omit names of attribute and operation in this paper.)

The physics part (Goemon) was designed to have:
1. Simplicity
2. Distinction between Component and Machine.
3. Separation from Hardware Layer
4. Separation from Graphics
5. Separation from Machine Operation

*This work was supported by the Director, Office of Energy Research, Office of Basic Energy Sciences, Material Sciences Division, of the U.S. Department of Energy, under Contract No. DEAC03-76SF00098.
Simplicity is important in the class library construction. The choice of inheritance or aggregation was the main issue for us. Component and Machine corresponds to magnets in a warehouse and an accelerator assembled from them. But the term Machine is usually used for the hardware, therefore we will call it Accelerator. Separation is to keep the model portable. Since the low level machine access can be performed without modeling, a model layer should be independent from it. Graphics heavily depend on the development environment, therefore the model should be separated from them. Machine, operation means various kinds of parameter fitting and machine study. Since it accesses the hardware and requires graphics, it must not be a part of the model. These requirements on separation can be well described as follows: Model, hardware access and graphics should be supported independently by their own class libraries and serve as suppliers to the client classes that include machine operations and studies. This is again the matter of has-a and is-a relationship.

III. CLASS LIBRARIES

Currently, we classify as follows:

Physics (Goemon)
Component Class
Accelerator Class
Hardware
Device Class
Client
Graphics Class
Operation Class
We describe the structure of 5 classes mentioned above.

A. Component Class

A beam line is a series of elements like drift spaces, magnets and monitors. The class Element serves as a base class for these elements. Drift is a class for drift spaces and a base for thick linear elements like quadrupole and bending magnets. Marker is a base class for markers and thin elements that cover multipole magnets. Wiggler is treated as a special quadrupole which will be enough for elementary linear optics calculation. When a better model for a wiggler/undulator is required, it will be a derived class of it.

The most important method of Element is pass that transfers a particle \(v=(x, p_x, y, p_y, \delta p, P_0)\) through it. Here \((x, y)\) is a transverse coordinate, \((p_x, p_y)\) is the canonical momentum, \(\delta p=\)momentum deviation, \(P_0=\)nominal momentum. The standard 4x5 matrix formalism is used as the integrator but it can be easily replaced with another formula making use of the inheritance mechanism without influencing other parts.

B. Accelerator Class

This is a class for virtual machines. BeamLine represents a beam transfer line that is a list of Component Class objects. It has an array of Element. Celement class has an Element, path and Twiss functions at each position. Ring is a circular BeamLine. It has a collection of Element objects to assemble a beam line. Ring is the class that directly supports dynamic modeling. An instance of Ring is a virtual accelerator. The figure (right) shows the relationship between Component and Accelerator classes. The ALS storage ring classes are derived from Ring. ALSSR is the ideal ring with full symmetry, ALSSR for full lattice and ALSSRW with wigglers/undulators. ALSSR has knobs to manipulate any magnets around the ring freely. Then it calculates all the linear optics, synchrotron integrals and related parameters.

C. Graphics Class

A graphical user interface library should support both windows environment on Unix workstations and PC running MS Windows. We use zApp [11] on PC to cover Windows 3.1 and NT. We will be using zApp when OSP/Motif version is released. A customized graphics class for ALS is being developed using zApp.

D. Device Class

Layered on top of the ALS control system, Device provides virtual devices. Channel corresponds to each hardware access channel (DMM database entry [7]). The figure shows that Device has one or more channels. PSMagnet is a base class for magnet power supplies. Since there are many quadrupoles, steering magnets and BPMs and are frequently used by application classes, dedicated access is provided.

Previously, Device was a descendent of Channel and served in the commissioning phase for process
controls. But this design turned out to be inadequate to handle devices that have many channels and is being rewritten as described above. This is also a matter of is-a and has-a relationship.

PSmagnet is for the magnet power supplies and contains subtasks to perform slow settings of currents. PSquads and PSsteer are for ganged operation that synchronizes subtask objects. DCCT is a class for a beam current monitor that keeps track of beam current intensity. BPMons handles 96 beam position monitors and have been used intensively during the commissioning period.

E. Operation Class

This supports various kinds of parameter fitting operations on the Accelerator object which covers tune fitting, orbit correction and undulator compensation. This class is a client of all other classes mentioned above and is specific to the ALS. The construction of this class has just started. Currently we have Smatrix and Bump.

Smatrix is for the sensitivity matrix manipulation including file access. An on-line data taking application and Goemon both use this class to have a common data format.

Bump is for local orbit bumps with 3 steering magnets. It can be associated with ALSSR or Smatrix, which makes both model-based and model-free local orbit corrections possible. As Bump is not a part of the model, it is possible to pass its objects to the real-time control layer for fast orbit corrections.

IV. FUTURE PLAN

The following items are on the list.

A. Persistency

As OOP itself does not support persistency, device objects must read and write the values of their internal parameters including the nominal current settings from/to files. There is a need for a database management system to administer these values with access to the objects. We are evaluating the Object-Oriented Database Class Library Raima Object Manager [12] for this purpose. It gives persistency to objects by using multiple inheritance.

B. Model-based Control

The effort to implement a model-based control layer on top of the existing control system has just been started for the operation with undulators. The migration of Goemon to the control system will be done by providing the server-client mechanism over the network. OOP in this area has not yet been well investigated.

C. Data Analysis

The ALS storage ring was operated for 6 weeks with RF. During that period, most of the machine studies were to measure fundamental parameters (closed orbit, tunes, chromaticities and sensitivity matrices) and they have not yet been fully analyzed. Several application programs based on the class libraries are being used, but there should be an effort to construct a class for data analysis.

V. ACKNOWLEDGMENTS

We would like to thank the ALS controls group for their continuous support, ALS operators for their patience with our programs, A. Jackson for his encouragement and Carl W. Cork for stimulating discussions and valuable suggestions.

VI. REFERENCES


[8] Eiffel 2.3 ( Interactive Software Engineering, CA).


[10] OMTool ( GE Advanced Concepts Center, PA)


[12] Raima Object Manager ( Raima Corporation, WA)