STATUS OF UNIFIED ACCELERATOR LIBRARIES

Nikolay Malitsky and Richard Talman,
Laboratory of Nuclear Studies, Cornell University, Ithaca, NY 14853

Abstract

The Unified Accelerator Libraries (UAL) form an object-oriented programming toolkit for developing distributed accelerator software. At this time the UAL joins accelerator programs DA, PAC, and TEAPOT thereby implementing a set of fundamental accelerator data structures and algorithms: an accelerator lattice model, element-by-element particle tracking, differential algebra, and others. The Application Programming Interface (API), written in Perl, provides a universal homogeneous environment for invoking, customizing, and extending diverse accelerator algorithms, and integrating them with other computer software. It can be considered a model of the Accelerator Simulation Facility based on the CORBA Business Object Facility and the UAL framework.

1 BASIC CONCEPTS AND ARCHITECTURE

The Unified Accelerator Libraries toolkit is designed as a customizable and extendible environment for developing mission-critical applications. This goal is achieved by the fundamental Model/View/Controller (MVC) paradigm[1]. Following the MVC pattern, the UAL is partitioned into three categories: Accelerator Objects (accelerator model, bunch, *etc.*), Physics (analysis, optimization and correction algorithms, *etc.*), and Application Programming Interface ('input language'). Each accelerator program is considered as a separate self-contained class, that may have its own internal organization and methods. Connection with the UAL is by common data objects of the first category. At this time the UAL joins three object-oriented accelerator programs (as shown in Fig. 1): Platform for Accelerator Codes (PAC)[2], Thin Element Program for Optics and Tracking (TEAPOT)[3], and Differential Algebra (DA)[4].

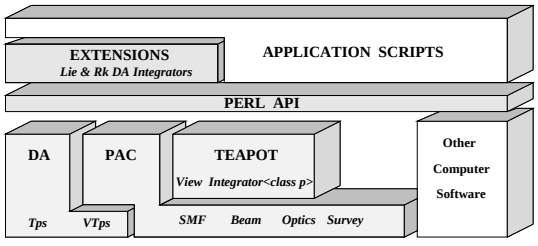

Figure 1: Unified Accelerator Libraries Environment.

The PAC is a collection of Accelerator Objects that can be shared, exchanged,or converted by other codes and processes. TEAPOT and DA are the implementations of the two different theoretical algorithms for simulating particle motion in accelerator elements. The UAL distributed architecture makes it possible to merge these diverse approaches seamlessly to provide optimal conditions for studying accelerator performance. Moreover, the present TEAPOT tracking engine is developed as a C++ template that can be instantiated either to provide the original functionality or as a DA integrator. The 'input language' is a key part of accelerator programs. According to the Standard Input Format (SIF)[5] (and the MVC paradigm) it may be divided into two major parts: the Accelerator Description and Actions. Accelerator is a complex system that includes many elements of different physical types, each having many attributes, all organized in a more or less hierarchical fashion. The second part, Actions, have traditionally consisted of an even more heterogeneous collection of commands and directives. This has resulted in the creation of diverse 'input languages' and formats, each requiring a 'proprietary', 'embedded' parser to perform the conversion from human readable ASCII files to the internal data structures. To resolve the problem we have introduced an Application Programming Interface (API) based on the standard object-oriented interpreter (e.g. PERL) and the UAL Accelerator Objects:

API = PERL + UAL Accelerator Objects

This approach makes it possible to include other 'input languages', such as MAD[5] and COSY[6], and provides a description of *all* accelerator elements and access to *all* accelerator methods. One can consider such a homogeneous shell as a foundation for establishing the Catalog of Accelerator Designs and Scenarios. Especially, it seems interesting for storing 'start-up' scripts of future Simulation Facilities.

2 APPLICATIONS

The last several years have witnessed a new phase in accelerator physics' evolution that is characterized by two tendencies: focus on new physical effects and devices (e.g. spin and helical dipole) and research on combined heterogeneous effects (beam-beam + machine nonlinearities + tune modulation + ...) on accelerator integral characteristics (luminosity, period of long-time instability, and others). The very flexible and extendible organization of the UAL environment is addressed to solving these kinds of tasks. In 1996/97 it has been applied by several groups to test its capabilities:

- **CESR**, Wilson Laboratory:
  - instantiation of uniform accelerator description that can be initialized from scripts or embedded C/C++ wrappers of control data;

0-7803-4376-X/98/$10.00

  - preliminary simulation of beam-beam performance for Möbius accelerator;
- **RHIC**, Brookhaven National Laboratory:
  - evaluation of performance with a helical magnet;
  - investigation of tune modulation;
- **Recycler**, Fermi National Laboratory:
  - evaluation and comparison of several theoretical models for combined function magnets;

3 CORBA INTEGRATION

An interesting and challenging accelerator task is steering theoretical and experimental activities in a common direction toward the development of an intelligent model-based control system. In the SSC laboratory, the Accelerator Simulation Facility (ASF) project was initiated by G.Bourianoff[7]. Since that time accelerator scientists and software developers in FNAL and TJNAF[8] have achieved significant progress, but the portability and interoperability of present simulation facilities are still active problems. To address them we have combined two components, Common Object Request Broker Architecture (CORBA)[9] and Unified Accelerator Libraries (UAL):

ASF = CORBA + UAL

CORBA is an industrial standard for object-oriented distributed systems, introduced and developed by a consortium, the Object Management Group (OMG), that includes over 700 companies. Accelerator software based on the CORBA implementations automatically becomes a part of the modern technological process and inherits many powerful CORBA benefits:

- standard distributed object infrastructure;
- uniform object-based interface and high-level language bindings;
- natural integration with Java mobile environment for developing the 3-tier Object Web systems;
- component portability and interoperability;
- local/remote transparency;
- natural integration with existing systems;
- 15 famous unified Services[10];
- Common Business Objects and a Business Object Facility[11];

The OMG Business Object Facility (BOF) is a logical product of the consistent CORBA evolution from interoperability to collaboration. The BOF is designed as a foundation of industry-specific frameworks and applications. Based on this specification, OMG members are currently developing domain frameworks for Finance, Manufacturing, Telecom, Electronic Commerce, and others. The accelerator community should be ready to initiate similar efforts. Figures 2 and 3 demonstrate parallelism between an OMG Business Object Component and an Accelerator Simulation Facility. The central concept of the BOF is a cooperative Business Object Component. According to the MVC pattern, Business Object Components consist of three kinds of objects (as shown in Fig. 2)[12]: Business Objects, Business Process Objects, and Presentation Objects.

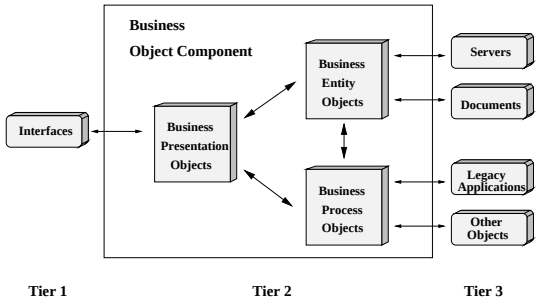

Figure 2: OMG Business Objects.

Components exist at several levels of abstraction: Enterprise Model, Domain Model, Business Object, and Atomic Concepts. They may be implemented by subtyping from existing components of the same level or by combining lower level components. To specify a Business Object Model (BOA) and to facilitate the application development process, the BOF provides also the Component Definition Language (CDL). CDL extends the CORBA Interface Definition Language (IDL) to represent the higher level concepts found in the BOA. In the BOF terminology, the ASF is an Accelerator Business Component at the Domain level (Fig. 3) that includes three cooperative objects: Accelerator, Control, and Interface.

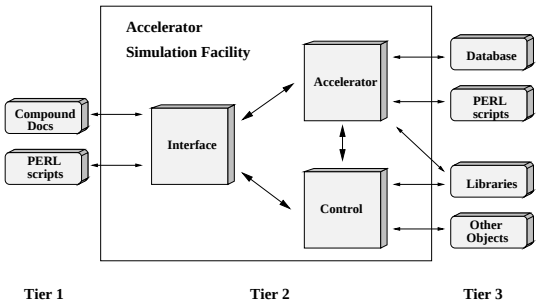

Figure 3: Accelerator Business Objects (Domain Model).

An Accelerator Business Object can be implemented from PAC Accelerator Objects, and TEAPOT tracking engine or other accelerator simulation codes. The next step in the UAL development will be the integration of TEAPOT matching and correction algorithms to become a part of the Control Business Object. The ASF will provide two different Interfaces: Perl scripts and Graphical User Interface based on the CORBA Compound Documents (containers of ORBlets). Communication between different components is by PAC Accelerator Objects. All components (and their subcomponents) may reside in one or more computers. The CORBA local/remote transparent mechanism makes these details invisible to users and significantly sim-

plifies the transformation of conventional codes into distributed facilities(Fig. 4).

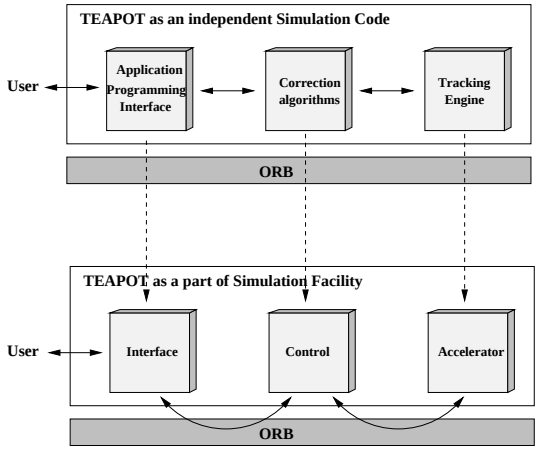

Figure 4: Simulation Code vs. Simulation Facility.

To exercise the connections between different systems, we have implemented a tiny tracking demo available over the Internet that provides the simplest remote control to the DA library (Fig. 5 and 6). Despite its simplicity, this example demonstrates two interoperability features of the CORBA architecture: high-level language bindings (Java and C++), ORB vendor independence (VisiBroker and HP ORB plus).

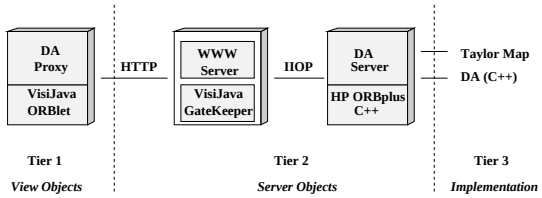

Figure 5: The 3-tier Object Web tracking demo.

4 ACKNOWLEDGEMENTS

One of the authors (N.Malitsky) would like to thank the members of the Nile project, Mike Athanas, Dan Riley, Mark Rondinaro, and Greg Sharp, for useful discussions and valuable suggestions.

5 REFERENCES

[1] G.E.Krasner and S.T.Pope. A cookbook for using the model-view-controller user interface paradigm in Smalltalk-80, Journal of Object-Oriented Programming, 1(3):26-49, August/September 1988.

[2] N.Malitsky, A.Reshetov, G.Bourianoff. PAC++:Object-Oriented Platform for Accelerator Codes, SSCL-675, June 1994.

[3] L.Schachinger and R.Talman. Teapot: A Thin-Element Accelerator Program for Optics and Tracking, Particle Accelerators, **22**, 35(1987).

[4] N.Malitsky and A.Reshetov, to be published.

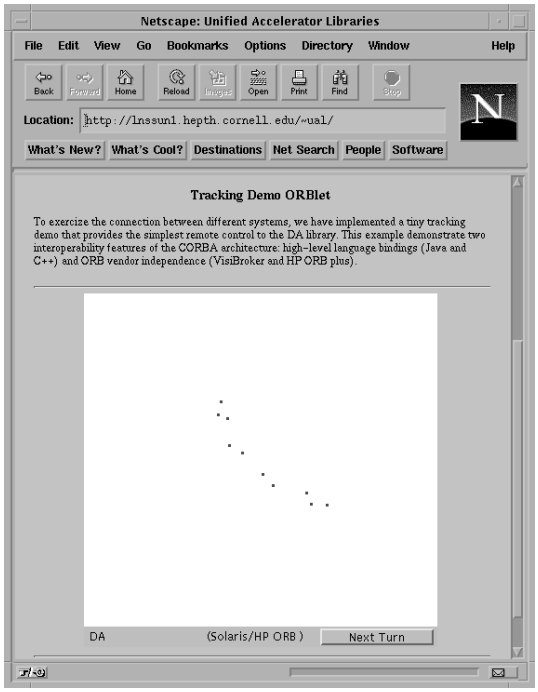

Figure 6: Tracking demo ORBlet.

[5] D.C.Carey and F.C.Iselin. Standard Input Language for Particle Beam and Accelerator Computer Programs, Snowmass, Colorado, 1984.

[6] M.Berz. COSY INFINITY, Refernce Manual, Technical Report 28881, LBL, 1990.

[7] G.Bourianoff, A.Reshetov, N.Malitsky. Object-Oriented Approach for the Design of the Simulation Facility of the SSC, SSCL-677, July 1994.

[8] C.Watson, J.Chen, D.Wu, W.Akers. Introduction to CDEV, Version 1.5, TJNAF, December 1996.

[9] CORBA 2.0/IIOP, OMG Technical Document formal/97-02-25.

[10] CORBAservices: Common Object Services Specification, 1996.

[11] BODTF-RFP 1 Submission, Business Object Facility, 1997.

[12] R.Orfali, D.Harkey, J.Edwards. Instant CORBA, Wiley Computer Publishing, 1997.