# DEVELOPMENT OF BEAM DYNAMICS APPLICATIONS WITHIN A CORBA FRAMEWORK AT THE SLS

M. Böge, J. Chrin, M. Muñoz, A. Streun, PSI, Villigen, Switzerland

*Abstract*

A distributed client-server model, based on the Common Object Request Broker Architecture (CORBA), has been established to interface beam dynamics applications at the Swiss Light Source (SLS) to essential software packages such as the accelerator physics package TRACY and the Common DEVice (CDEV) control library. Within this model remote clients can invoke computer intensive methods, such as beam orbit correction procedures, on a dedicated server. Access to the SLS accelerator devices is achieved through a dedicated C++ CDEV server. A status report on application development within the established CORBA framework is given.

## 1 INTRODUCTION

The SLS[1] is a 2.4 GeV electron storage ring currently under construction at the Paul Scherrer Institute (PSI), Switzerland. Electrons from an injector booster synchrotron, fed by a 100 MeV linac[2], are transferred to the storage ring at full operating energy. Scheduled for operation in August 2001, the SLS will provide synchrotron radiation of high brilliance to experimenters from a variety of disciplines. A considerable number of high-level beam dynamics Application Program Interfaces (APIs) are required for the commissioning and operation of the SLS accelerator complex and for machine physics studies. These APIs typically share a number of generic tasks including:

- access to an accelerator physics package,
- accelerator device control,
- database access and management, and
- logging of messages and alarms.

With the aid of object-oriented methodology, common functions can be identified and developed as reusable components. Furthermore, a distributed system allows optimal use of available resources, an important consideration given the CPU intensive physics algorithms employed by the accelerator modelling procedures. To this end, a distributed client-server model, based on the Common Object Request Broker Architecture (CORBA)[3], has been proposed[4]; client programs readily access shared services, either locally or across the network, through CORBA objects.

## 2 THE CORBA FRAMEWORK

In the evolution of object-oriented distributed computing systems, CORBA is a recent standard that provides a mechanism for defining interfaces between distributed components. Its most distinguished assets are platform independence, in so far as the platform hosts a CORBA Object Request Broker (ORB) implementation, and language independence, as ensured through the use of the Interface Definition Language (IDL). The latter feature is of particular interest to SLS beam dynamics API developers as it provides for the option between high-level application languages.

### *2.1 Server Hardware and System Software Components*

The chosen platform for server software components is a dual 600 MHz Intel Pentium III system ("Model Server") running Linux (RedHat 6.0). The use of Linux and the GNU project C++ compiler (egcs) avoids vendor dependency; compilation with egcs further reduces the dependency on the operating system thereby increasing the portability of applications. A mirror server is permanently available to provide redundancy.

The ORB (Object Request Broker) employed is MICO[5] a fully compliant CORBA 2.3 implementation available under GNU public license terms. Use is made of the Implementation/Interface Repository facilities, the Naming Service and the Event Service[1] of the ORB. MICO provides IDL to C++ mapping. The Tcl interface Combat[6] adds the Tcl mapping to MICO. For Java mapping the Java-based ORB JavaORB[7] is used[2].

### *2.2 Client-Server Software Components*

Fig. 1 illustrates the conceptual design of the client-server software components. The C++ based server components run exclusively on the "Model Server". The Tcl/Tk[8] and Java[9] based clients reside on the consoles. The following describes the server components:

- **CDEV Server:** Synchronous and asynchronous interaction with the EPICS-based SLS control system[10] is achieved through the use of the CDEV C++ class library (version 1.7.2)[11]. A dedicated CORBA CDEV server responds to the CDEV-type verbs "set", "get" and "monitor" which respectively download setpoints, readback device attributes and monitor selected channels. A change in value of a monitored channel invokes the CDEV callback function, wherein the

[1]The Event Service provides services for the creation and management of CORBA event channels that may be used by CORBA supplier/consumer clients to propagate events asynchronously on a push or pull basis. Event channels are created and registered with the CORBA Naming Service allowing clients to obtain object references in the usual manner. Communication is anonymous in that the supplier does not require knowledge of the receiving objects. The CORBA Event Service has been usefully employed in the monitoring of hardware devices and in the distribution of recalibrated data to client consumers.

[2]The Internet Inter-ORB Protocol (IIOP) allows for the communication between different ORBs which fulfill the CORBA 2.0 specification.

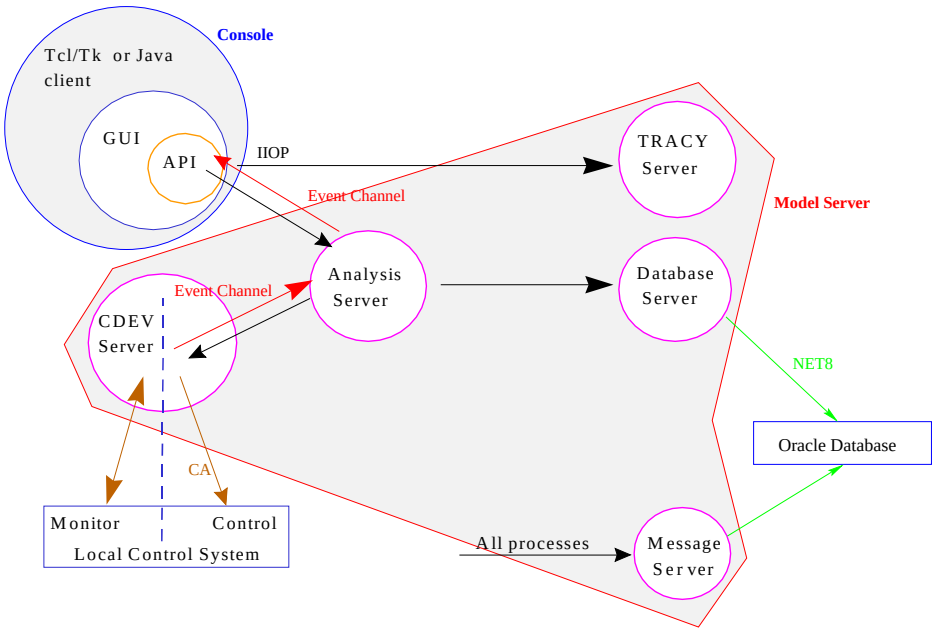


Figure 1: Conceptual design of the client-server software components.

new data value is both stored in memory and supplied to a CORBA event channel. Clients can either retrieve data from memory through the invocation of a CORBA method or be informed of new values by subscribing to events from the appropriate event channel.

- **Analysis Server:** Typically data from the real-time control system require recalibration and/or analysis before being displayed by the client consumer. Such tasks are handled by the analysis server. Recalibrated data are then distributed to clients through a CORBA event channel.
- **TRACY Server(s):** The TRACY[12] servers provide access to the TRACY based models of transfer lines, booster synchrotron and storage ring. This includes methods to retrieve linear optics parameters for given machine settings and to perform model based corrections. The capability of having access to all accelerator physics routines of the C-based TRACY library in itself provided strong motivation for the use of CORBA as it allows access to the same machine model as used in offline simulations; procedures tested in simulation can be effectively employed for the optimization of the accelerator *online*.
- **Database Server:** The CORBA database server provides access to the Oracle instances. It utilizes the OCI8 (Oracle Call Interface) API through the OTL (Oracle Template Library) library[13]. The OTL code is expanded into direct database API function calls, thus providing ultimate performance and reliability. For the time being the server gives access to "static" database tables, but it is foreseen to use it for fast "online" data storage as well.
- **Message Server:** All client-server processes are able to report messages and alarms to a dedicated CORBA message server. The server employs the UNIX syslog message logging facility incorporating a variety of priority levels. Syslog entries are further written to a named pipe. A listening server converts them to SQL insert queries for immediate entry into the Oracle database utilizing the native Oracle SQL*Plus client. Error messages are viewed either through a Tcl/Tk based browser polling the log files or by retrieving the corresponding database table.

## 3 THE APPLICATIONS

Table 1 summarizes the applications needed for the linac (commissioned in February 2000), the linac-booster transfer line and the booster synchrotron (to be commissioned in July 2000).

To illustrate the look and feel of the CORBA based applications, a screen shot of a Tcl/Tk based application is shown in Fig. 2 which displays linear optics parameters of the linac-booster transfer line optics derived from the actual magnet currents. This client invokes methods on a dedicated TRACY server implementing the optics model of the transfer line, the CDEV server to set quadrupole currents and the message server to perform the logging of messages. The actual quadrupole currents get automatically pushed to the client through a CORBA event channel to which the client subscribes. The analysis server acts as the push supplier retrieving the data from the CDEV server which monitors the devices. It is interesting to note that the Tcl/Tk client program is comparatively short in length and, therefore, quite managable. Optimal use of the Tcl/Tk package is made for building the graphical user interface component of the API, while the more complex components are routed to server processes on the "Model Server".

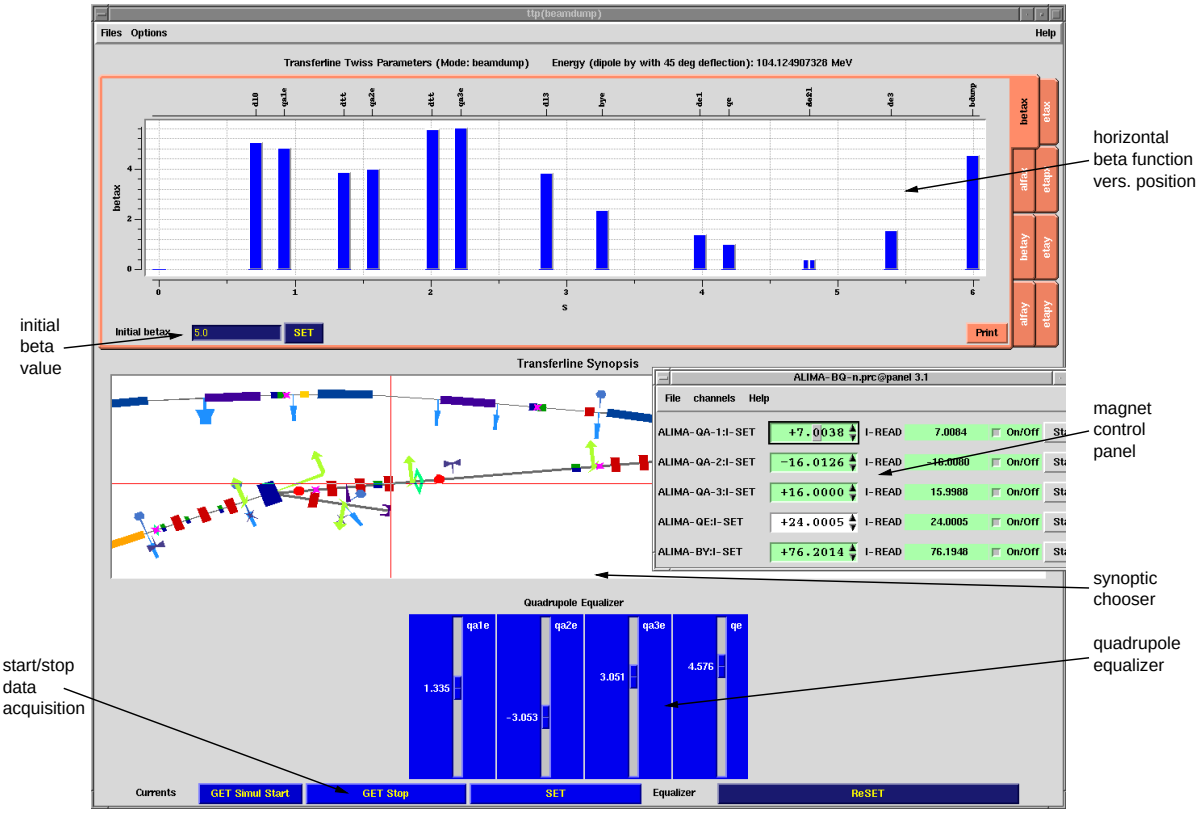


Figure 2: Linac-booster transfer line application “Transferline Twiss Parameters”.

Table 1: Application catalogue for linac and booster

| Linac | Booster |
|---|---|
| emittance measurement | injection knobs |
| energy jitter measurement | beam threader |
| monitor control | orbit correction |
| optics parameters (Fig. 2) | local orbit bumps |
| | tune measurement |
| | tune control |
| | response matrix |
| | dispersion & chromaticity |
| | ramp editor |
| | lifetime |
| | beam loss monitors |
| | lattice generator |

## 4 CONCLUSION

Beam dynamics applications have been successfully developed in a CORBA framework. It has been demonstrated that the presented complex client-server model is reliable and managable.

## REFERENCES


[1] M. Böge *et al.*, “The Swiss Light Source Accelerator Complex: An Overview”, EPAC’98, Stockholm, June 1998.

[2] M. Pedrozzi, C. Piel, “Commissioning of the SLS-Linac”, Contribution to this Conference.

[3] Object Management Group (OMG), “The Common Object Request Broker: Architecture and Specification, Revision 2.2”, February 1998.

[4] M. Böge, J. Chrin, “A CORBA Based Client-Server Model for Beam Dynamics Applications at the SLS”, ICALEPCS’99, Trieste, October 1999.

[5] A. Puder, K. Römer, “Mico is CORBA, Version 2.2.2”, Pub: dpunkt.verlag, December 1998.

[6] F. Pilhofer, “Combat, CORBA scripting with Tcl”, http: //www.informatik.uni-frankfurt.de/~fp/Tcl/tclmico/.

[7] J. Daniel, “JavaORB”, http://www.multimania.com/dogweb/details_javaorb.html.

[8] J.K. Ousterhout, “Tcl and the TK Toolkit”, Pub: Addison-Wesley, 1994.

[9] “Java”, http://java.sun.com/.

[10] S. Hunt *et al.*, “Control and Data Acquisition System of the Swiss Light Source”, ICALEPCS’99, Trieste, October 1999.

[11] J. Chen *et al.*, “CDEV: An Object-Oriented Class Library for Developing Device Control Applications”, ICALEPCS’95, Chicago, November 1995.

[12] 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.

[13] S. Kuchin, “Oracle and Odbc Template Library Programmer’s Guide”, http://home.sprynet.com/~skuchin/.