C++ Library for Accelerator Control and On-Line Modeling

S.Kuznetsov

Kurchatov Institute, 123182 Moscow, Russia

Abstract

A new object-oriented package has been developed at the SIBERIA SR source complex. This paper presents a formal approach to the interface between the control of a storage ring or transport line and beam modeling. A class library for on-line modeling is portable and uses the standard MAD input file interface for machine description. The object-oriented conception enables the realization of suitable parent-child class trees for control in "beam" terms. Simulation is based on the first order matrix formalism.

1. INTRODUCTION

SIBERIA-2 is a dedicated light source with an electron energy of 2.5 GeV that was developed by the Budker Institute of Nuclear Physics (Novosibirsk) for the Kurchatov Institute (Moscow) and commissioned in the beginning of 1995 in Moscow.

Using on-line modeling software complexes is very popular at modern accelerator facilities.[1-4] This type of software is very important during the commissioning of the machine. One characteristic of synchrotron radiation facilities is the continuous modifications carried out for new insertion devices. The software support for work in beam terms is necessary. We have tried to design an object-oriented library that combines control methods for specific equipment (power supplies, pulse generators, etc.) with control methods in beam terms: coordinates, angles, betatron tunes, chromaticity, energy.

Beam Objects	coordinate, angle, betatron tunes, chromaticity, energy...	X, X' Z, Z' ν_x ν_z ξ_x ξ_z
Element Objects	bending magnets, correctors, quadrupole lenses, sextupole lenses, beam position monitors.	B1 , B2 KX1, KX4, F1, D2, F2
Control channel Objects	ADC, DAC	IF1, ID2, ISx, ISz, IKX1, IKX4

Figure 1. Objects of an accelerator control system.

2. DESCRIPTION

To control the magnet system of a beam transport line or storage ring we use the classes: TMSControlChannel - magnet elements control, TPSControlChannel - power supplies control, TBMChannel - virtual beam-monitor channels control. These classes are derived from the abstract class TChannel. The hierarchical class structure is presented below:

```
TChannel
    TMSControlChannel
        TMSGroup
            TLineGroup
                TRingGroup
```

TMultGroup
TPSControlChannel
TBMChannel
THandle
TMatrixHandle
TControlHandle
TSimpleHandle
TMSHandle
TQXZHandle
TSXZHandle
TCorHandle
T2CorHandle
TX2CorHandle
TZ2CorHandle
T3CorHandle
TX3CorHandle
TZ3CorHandle
T4CorHandle
TX4CorHandle
TZ4CorHandle

To realize control procedures in beam terms, on-line information about the magnetic structure of an accelerator is necessary, namely the transport matrix, input/output Twiss parameters for every element of the structure, etc. The class TMSControlChannel represents all necessary requirements for such on-line modeling.

TMSControlChannel class

```
class TMSControlChannel : public TChannel
{
……….
        PYMatrix            XMatr, ZMatr;
        PYVector            Xin, Zin, Xout, Zout, Xoffset, Zoffset;
        PTwVector           TwXin, TwZin, TwXout, TwZout;
……….
        virtual void        FillMatrix(PTEnergy);
        virtual void        SetTwissIn( PTwVector, PTwVector);
        virtual void        TwissInOut();
        virtual void        SetXZIn( PYVector, PYVector);
        virtual void        SetXZoffset(PYVector, PYVector);
        virtual void        XZInOut();
……….
        virtual void        NewAdd(PTLineGroup);
        virtual void        Copy(PTMSControlChannel NewChan);
……….
};
```

The class TMSGroup is a set of magnetic elements (e.g., objects of the TMSControlChannel class) and on the other hand it is a magnetic element itself, for which all the necessary modeling procedures are realized. The first group of methods provides the standard ‘List’ implementation for TMSControlChannel objects. Another method supports calculations for orbit and Twiss parameters.

TMSGroup class

```
class TMSGroup : public TMSControlChannel
{
protected:
        TMchanNode          MChannelList;
.....
       virtual  void                    Add( PTMSControlChannel);
        virtual void                    AddAt( PTMSControlChannel,int);
        virtual void                    Free( PTMSControlChannel);
        virtual void                    FreeAt( int);
        virtual PTMSControlChannel      At( int);
```

```
        virtual int                     IndexOf( PTMSControlChannel);
        virtual PTMSControlChannel      FirstThat( TMCondFunc, void *);
        virtual PTMSControlChannel      FindWithName(char *);
        virtual void                    ForEach( TMActionFunc, void *);
        virtual void                    ForEachIter( TMActionIter, void*);
        virtual void                    Copy(PTMSControlChannel*);
……….
        virtual void                    FillMatrix(PTEnergy);
        virtual void                    TwissInOut();
        virtual void                    XZInOut();
……….
};
```

The realization of the method which calculates standard Twiss parameters for a group is:

```
void TwissLine(PTMSControlChannel AMchannel, PTMSControlChannel BMchannel, void* PS)
{
   if(PS)
        BMchannel->SetTwissIn(PTMSGroup(PS)->TwXin, PTMSGroup(PS)->TwZin);
   else
        BMchannel->SetTwissIn(AMchannel->TwXout, AMchannel->TwZout);
    BMchannel->TwissInOut();
}
void TMSGroup::TwissInOut()
{
        ForEachIter(TwissLine,this);
        TMSControlChannel::TwissInOut();
        TwXout->mu = MChannelList->Mchannel->TwXout->mu;
        TwZout->mu = MChannelList->Mchannel->TwZout->mu;
}
```

This is the complete code for the method . One can see the readability of the code. In this manner we can build a beam transport line or a storage ring structure and get information about the machine functions and update it after any variation in the settings of magnetic elements in real-time.

Class THandle and its derivatives are used to provide linking between control channels of different types. Objects of this class convert beam parameters (for example coordinates and angular deflection of the electron beam at a given accelerator azimuth) to control settings for magnetic elements (e.g., corrector magnetic fields) and then to appropriate power supply settings. The local bump, betatron tune control and chromaticity control classes were implemented using this technique.

THandle class

```
class THandle
{
.....
        PTChannel*          InVector;
         int                sizeIn;
         PTChannel*         OutVector;
         int                sizeOut;
.....
        virtual  void       Connect(void* );
         virtual void       Control();

         virtual void       Compute();
         virtual void       InvCompute();
};
```

Let us consider an example of a control procedure. We construct an object of the TX4CorHandle class. The information about the magnetic elements (4 X-correctors and a virtual beam position channel) is used and objects of TMSGroup class are constructed to calculate the transport matrix between elements.

Then executing the SetValue() procedure of the beam position channel to change the beam coordinate TMSControlChannel::Control() method involves appropriate methods of TX4CorHandle. Then the necessary simulation information is calculated (transport matrix and matrix binding the beam coordinate and

angle with the strengths of the correctors in this case). After that we can execute the Control() procedure for these correctors. This procedure uses objects of the TSimpleHandle class for binding to specific power supply control channels. The TX3CorHandle and TX4CorHandle classes support local bumps using both 3 and 4 correctors.

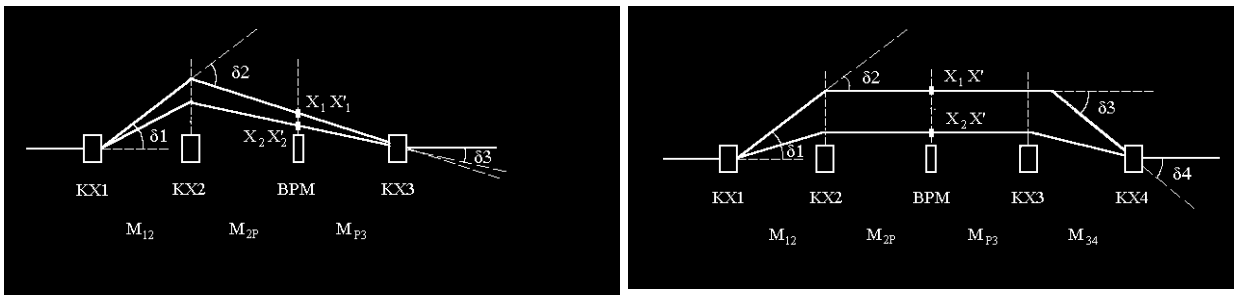

Figure 2. Three- and 4- corrector local bumps.

3. EXAMPLE

The library was successfully ported to the HPUX platform at the NSLS facility. The software complex was developed for on-line modeling. It is based on the GLISH software bus [2] which provides interprogram communication via events. The programs include:
"Readdbf" - reads the description of a MAD file and sends ring structure information to GLISH;
"Model" - gets a structure from MAD, calculates Twiss parameters of the ring (for each element in-out or for a selected group of elements with defined step in s-direction), gets readback information from the control system for energy and values of the quadrupole lenses, updates Twiss parameters, sends information to GLISH;
"QHandle" - the manager for tune on-line modeling - requests the MAD file, sets on-off status for readback, displays working-point tune diagram;
"ReadMes" - receives requests for getting readback of tune measurement values, sends information to GLISH;
"Dtwiss" - the manager for Twiss on-line modeling - requests the MAD file, selects the part of the ring, gets information from GLISH, displays graphics of the Twiss parameters and the structure of the selected part of the ring. An example of a graphic implementation using Motif is shown in figure 3.

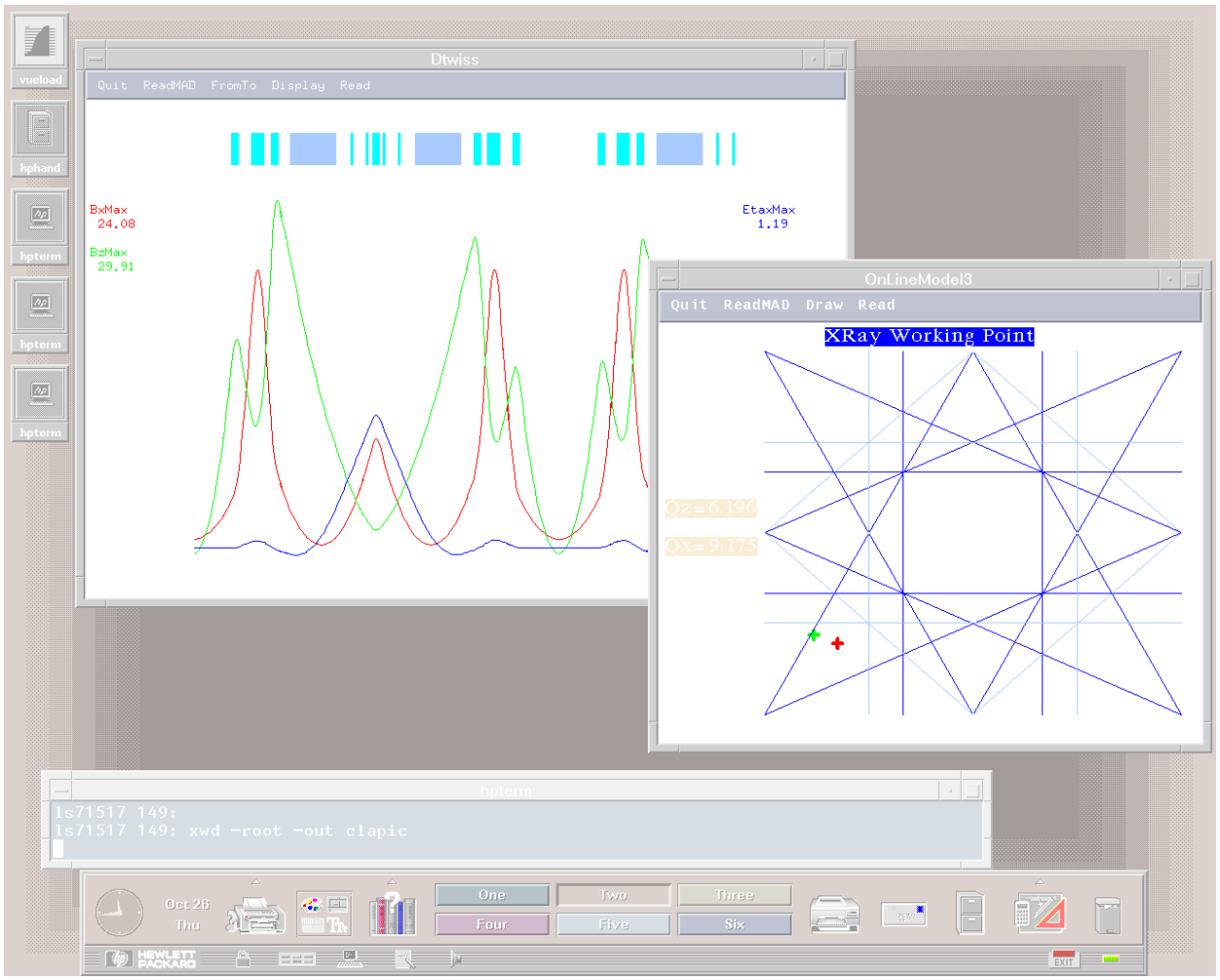

Figure 3. Example of on-line modeling.

4. REFERENCES

[1] H.Nishimura, "Dynamic Accelerator Modeling", Proc. of PAC'93, Washington, 1993, pp.111-113.
[2] L.Schachinger, V.Paxon "A Software System for Modeling and Controlling Accelerator Physics Parameters at ALS", Proc. of PAC'93, Washington, 1993, pp.1940-1942.
[3] M.Bickley, et al.,"Orbit Correction Implementation at CEBAF", Proc. of PAC'93, Washington, 1993, pp.1895-1897.
[4] M.Plesko, “A Complete Data Structure for High Level Software of ELETTRA”, Proc. of EPAC’94,London, 1994.