A PARALLEL CONTROLS SOFTWARE APPROACH FOR PEP II: AIDA & MATLAB MIDDLE LAYER *

W. Wittmer†, W. Colocho, G. White,
SLAC, 2575 Sand Hill Road, Menlo Park, CA 94025, USA

Abstract

The controls software in use at PEP II (Stanford Control Program - SCP) had originally been developed in the eighties. It is very successful in routine operation but due to its internal structure it is difficult and time consuming to extend its functionality. This is problematic during machine development and when solving operational issues. Routinely, data has to be exported from the system, analyzed offline, and calculated settings have to be reimported. Since this is a manual process, it is time consuming and error-prone. Setting up automated processes, as is done for MIA (Model Independent Analysis), is also time consuming and specific to each application. Recently, there has been a trend at light sources to use MATLAB[1] as the platform to control accelerators using a "MATLAB Middle Layer" [2] (MML), and so called channel access (CA) programs to communicate with the low level control system (LLCS). This has proven very successful, especially during machine development time and trouble shooting. A special CA code, named AIDA (Accelerator Independent Data Access [3]), was developed to handle the communication between MATLAB, modern software frameworks, and the SCP. The MML had to be adapted for implementation at PEP II. Colliders differ significantly in their designs compared to light sources, which poses a challenge. PEP II is the first collider at which this implementation is being done. We will report on this effort, which is still ongoing.

INTRODUCTION

When new methods to correct the optics of an accelerator emerge, an implementation can take considerable effort and time. For machine development (MD), ad hoc experiments, and where resources are scarce, fully integrated applications are not developed to integrate these new methods. Experiments and measurements, which are performed irregularly, are not automated. These "manual" measurements are time consuming and error prone. Analyzing the data from these measurements is usually performed after the MD has finished and if parts of the data are unusable it is not possible to quickly repeat a partial measurement.

In light sources this problem has been overcome by using MATLAB. Several CA tools have been written to import data directly into MATLAB and enable device set points to be changed. These mostly interact with EPICS, which provides the low level control. Setting up the basic data acquisition, processing and hardware control, as pre-prepared matlab scripts, permits MDs to be performed efficiently and without time consuming preparations.

Consequently, a suit of MATLAB functions, referred to as the MML, has been developed, which handles many of the common tasks. This allows one to rapidly create scripts for non-routine tasks. The biggest advantage of this package is that it is easily portable between different accelerators. It has been ported from SSRL, where it was originally developed, to more than ten different light sources all over the world. The MML interface to the LLCS, is handled by two simple functions for input and output, and the machine dependent features (lattice, naming convention, et cetera) are handled centrally by a machine specific configuration file. Adapting these files to a given machine, make the basic functionality immediately available and extensions are simple.

IMPLEMENTATION AT PEP II

The above described characteristics of the MATLAB MML package, make it suitable for used as a platform for MD at PEP II. A further advantage is that the MIA package is also based on MATLAB, so communication between MIA and the machine is greatly simplified through this approach. The main difficulty to overcome was to establish the communication between the MATLAB MML and the LLCS. Since EPICS is not used at PEP II greatly, a different route, similar to the one at NSLS, had to be taken. Fortunately, a software project was already under way at SLAC, called AIDA, which provides a unified mechanism for scientific applications to access the various control systems of the different accelerators at SLAC. This had to be expanded to provide the needed functionality.

AIDA - ACCELERATOR INDEPENDENT DATA ACCESS

AIDA is middleware for fast data exchange. It connects online physics applications software to the various control systems and data sources these programs require to do analysis and optimization of scientific machines. Such data sources include device control, online beam model parameters, archived process variable history, relational database access, and so on. As such, it is used by programs like orbit correction, emittance, and general purpose experimental tools like "Correlation Plots" [5], etc, to connect them to control systems such as EPICS [6], legacy control system programs, and "persistent" data stores like Oracle, EPICS archiver and so on.

* Supported by US DOE under Contract DE-AC03-76SF00515
† wittmer@slac.stanford.edu

AIDA is not for real-time control, but it is fast (2ms round trip for simple data over a 100mbs, network as is typical in an accelerator laboratory). Its client side is Java [7]. This means of course it can be used as the data access layer of modern software. Notably though, this feature means it can be accessed directly from MATLAB, with no intermediate wrapping such as Mex files necessary [8].

Initially AIDA was prototyped to verify whether projected ILC online physics applications programs, could be written independently of the accelerator subsystem control systems with which they would have to interact. The rationale was that there would probably be a number of such control systems in ILC, each quite different, and high level applications would be required to talk to all of them in their own language. After a successful prototype was developed, it was deployed for the B-factory accelerator at SLAC.

AIDA is now being used for commissioning LCLS. For LCLS it bridges the legacy control system and modeling environment, which is implemented on HP Alpha machines running VMS, to Unix and Windows users running MATLAB based GUI applications.

The AIDA Programming Interface

Technically, AIDA acts as a name service and common Applications Programming Interface (API). AIDA presents the same API to client applications for all data sources. That is, whether you are getting the value of a magnet, or the value of all Beam Position Monitors in a beamline, or the transfer matrix from one device to another, the methods you call are the same. Each request to get data or set it, includes a ”name//attribute” pair to identify the data source to be contacted (notice that more than 1 service may be involved with a device - the present value may be acquired by AIDA via EPICS, its Twiss may come from a model server, its history from Oracle etc). Since AIDA can get any type through the same API, including structured values, the client side type must be specified. Other AIDA API methods allow this to be done by casting.

Rich Server Side, not Rich Client Side Middleware.

The important characteristic of AIDA for its use in middleware for optimizing the B-factory, has been that it allows the use of high level applications code on the server side (as opposed to the client). That is, rather than simply access a database of controls API like EPICS, AIDA servers can offer rich functionality, like beam synchronous acquisition of a whole accelerator including subtraction of a reference orbit, accessible in one client call. For instance:

```
da = DaObject;
% Specify a timing definition
da.setParam('BPMD=38');
% Require diff to the "GOLD" orbit
da.setParam('CNFTYPE=GOLD');
% Set num ring turns to average
da.setParam('N=1024');
% Make the actual acquisition
DaValue v = da.getDaValue('P2BPMHER//BPMS');
```

Two things are of note for middleware implementations, are illustrated in this example. Firstly, the acquisition is parameterized by timing, by a reference orbit (so a ”difference” orbit shall be returned), and by an indication of turns to average. Secondly, notice that the data returned will be structured. In this specific case it's a 2d vector of vectors. The inner vector gives the name, x, y, z, tmit, hardware status, and an acquisition diagnostic value, for each beam position monitor in the beamline whose orbit was acquired. The outer vector includes them all, sorted by z position.

A DaValue is in fact a dynamically constructed self-describing data structure. On the wire it's a CORBA[9] Any type, though AIDA wraps it into a much more user friendly dynamic type called a DaValue. If a client program knows the structure of what it's going to get, it can access the elements directly. Additionally, DaValue includes facilities for a client to dynamically inspect it (like Java ”introspection”) recursively.

This facility for pushing the smarts into the server (so called Service Oriented Architecture), has technical and organizational advantages. The technical advantage is the facility to interface modern applications to mature control systems that already include much analysis code and therefore can't be accessed through a simple name/value paradigm. Organizationally, it allows computer professionals to implement the server side in a rigorous way, while allowing the physics professionals to concentrate on the data analysis.

MML - MATLAB MIDDLE LAYER

As preparation to the adaption of the two interface files and the machine specific configuration file of the MML, the new interfaces provided by AIDA had to be tested. The following capabilities, provided through AIDA, have been tested: synchronized orbit data acquisition, magnet control, analog channel data acquisition, SCP archive and EPICS channel archiver data access. The capability to change the master oscillator has been recently added and its test is being prepared.

This testing also provided important insight how to perform this adaption. Contrary to the other CA codes, AIDA provides two different approaches to setup the interface between MML and LLCS as described above. We have chosen the method that allows to use the MML without having to change interface.

The machine specific configuration file has to be written from scratch and cannot be derived from the files for other light sources due to the fundamental differences between colliders and light sources. This effort is ongoing. First, applications have been written for simple tasks using the schematics of this approach and will be included once the setup of the MML for PEP II has been completed. These

first implementations already show the benefit from this approach.

Simultaneous Orbit Measurement with two BPM Systems. As an example, part of the data analysis for the newly installed BPM (beam position monitor) electronics in the LER (low energy ring) [10] is presented below. The simple task is to plot the orbit data off the old PEP II BPM electronics plus those newly installed, in one figure. To perform this in the SCP would be an excessive project. This is due to the different underlying data structure for the different systems.

With AIDA this was a fast and simple exercise finished in one afternoon. The interface for importing the orbit data into MATLAB of the older BPM electronics was already existing from the testing. The new units are set up as analog channels in the SCP. The access for these were also existing and a function to import these specific channel had to written and tested. These two functions were combined and the MATLAB tools to plot both data sets in one figure used. The result is shown in fig.1 The main goal of

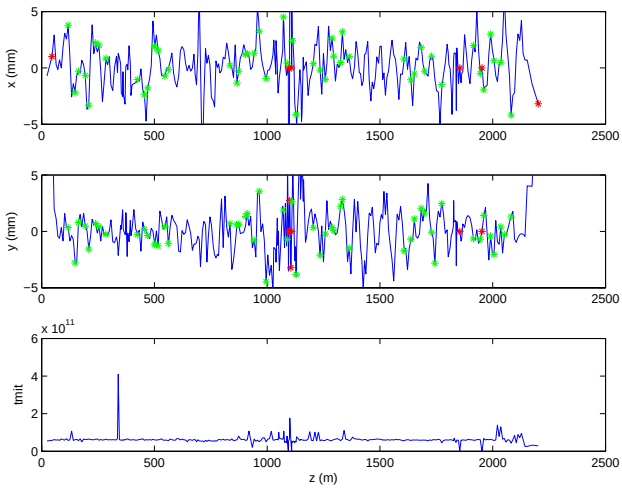

Figure 1: Live orbit data from the LER in PEP II. The blue trace represents the orbit measurement from the older system. The red stars are units flagged as showing bad data. The green stars are the orbit measurements with the new system.

this analysis is to track the difference between the position readings of the two systems. One important aspect is to make sure only measurements from working units are used for this comparison. Since this information is part of the data structure imported into MATLAB all needed information is available. Processing and displaying was added to the function. This script is used to commission the new system. Using the SCP to perform this task would have implied checking the data manually every time.

Vacuum monitoring A collider as large as PEPII has thousands of analog channels monitoring multiple subsystems. Another application of the AIDA/MATLAB software package is to plot a large number of analog channels on a condensed way. For example, to plot all the HER vacuum pumps for a given time period.Anomalous activity can then be found at a glance. One figure can show 100s of channels with time in the x axis, channel number in the y axis and vacuum amplitude as the color scale. Vaccum activity arround the time of PEP II beam abort is sometimes easily identified by these plots.

LER Sextupole Orbit Feedback The SCP provides an orbit feedback in the LER sextupoles. This existing feedback only monitors and corrects orbit changes in the vertical plane. Its adaption to correct both planes simultaneously would have been too large an effort. The MML provides the capability of orbit correction so to write a new routine handling this feedback task would not be a large effort.

STATUS AND OUTLOOK

So far the major preparation for introducing the MML for MD studies at PEP II have been successful. The basic communication between MATLAB and the LLCS provided by AIDA is completed and all tests passed. The adaption of the MML is ongoing and will be completed in the following months. First applications, based on the package AIDA-MML have been developed and are being used. After the completion of the package MIA will immediately benefit through its shared MATLAB platform. Small applications, like the LER sextupole feedback, are planned for normal operation. Once the MML is being used during MD shifts we anticipate its use also during normal operation.

REFERENCES

[1] The MathWorks. http://www.mathworks.com/

[2] J. Corbett, G. Portmann and A. Terebilo, “ACCELERATOR CONTROL MIDDLE LAYER,” PAC’03, June 2003, Portland, p. 2369, http://www.JACoW.org.

[3] Accelerator Independent Data Access (AIDA). http://www.slac.stanford.edu/grp/cd/soft/aida/

[4] The Linac Coherent Light Source Conceptual Design Report, SLAC-R-593

[5] Correlation Plot Facility in the SLC Control System. L. Hendrickson, N. Phinney, L. Sanchez-Chopitea, S. Clark(SLAC). SLAC-PUB-5685, Nov 1991. 3pp.

[6] Experimental Physics and Industrial Control System (EPICS). http://www.aps.anl.gov/epics/index.php

[7] Java Technology. http://java.sun.com/

[8] Using AIDA from Matlab. http://www.slac.stanford.edu/grp/cd/soft/aida/aida_matlab.html

[9] Common Object Request Broker Architecture, http://www.omg.org/docs/formal/04-03-12.pdf

[10] W. Wittmer, A.S. Fisher, D.J. Martin, J.J. Seebeck’ “Detection of Instumental Drifts in the PEP II LER BPM System,” these proceedings.

