# CROSS-PLATFORM AND CROSS-ACCELERATOR MACHINE PHYSICS PROGRAMS WITH DATABUSH

I. Kriznar, M. Plesko, A. Pucelj, A. Zupanc (JSI and Cosylab), J. Galambos (SNS)

*Abstract*

Databush, a collection of Java classes and utilities for machine physics, serves well at the synchrotron light source ANKA. Machine physics applications with a visually appealing user interface can be written with Databush quickly and efficiently and thanks to Java they are cross-platform. Modular architecture and use of Abeans, a communication and application Java framework, has extended the usability of Databush even across different control systems. With improvements and upgrade to latest Abeans R3 libraries the Databush applications have been tested on EPICS based control system at SNS. Additional applications for specific needs of the SNS have been written in short time. Lattice data was obtained through the XAL Java library of SNS. Databush applications have kept robust and appealing user interface, which was polished up at ANKA in order to be used by operators without background in physics. We plan to use the same applications also for the upcoming SESAME light source.

## INTRODUCTION

Databush is a Java package with applications, GUI and calculation tools for writing machine physics applications. Applications made with Databush, such as OrbitCorrection, are running at ANKA accelerator. Recently, Databush package has been adjusted to the new version of Abeans library, release 3.0. With this improvement Databush applications are now available for EPICS channels. OrbitDisplay application was tested at SNS accelerator with their front-end structure. Databush has two layers of abstraction of a control system, that makes it able to quickly adopt different kinds of remote or local data sources. With Java, Databush is automatically runnable at all platforms where Java Virtual Machine can be run.

## DATABUSH

Databush physical model of accelerator structure is optimized for fast calculations of linear optics and delivers results in structure of machine optics objects. Databush subscribes as data change listener to dynamic value sources, thus provides a quasi real-time state of a machine. A machine lattice is organized in hierarchy of collaborating machine elements (magnets, BPMs, etc.), as seen in UML diagram 1. These elements may be tagged as virtual (simulated without external data source) or connected to external data source via Property interface. Databush structure is self-contained. An application can make a snapshot of the machine state, render state in Databush model and apply changes back to the machine. Since Databush is a Java package, its applications are platform independent. Plug-in additions allow fine-tuning of Databush to meet the needs of a particular implementation.

Control system and communication protocol specifics are hidden from Databush applications with two levels, Datatype Properties, and Abeans Plug. Each of these levels can be replaced with an alternative implementation. Two major benefits result from this architecture:

- Databush machine physics application can be used on other accelerators without changing a single line of code, with appropriate modification of plugs, of course. This then enables sharing accelerator software among different accelerators.
- Physicists can concentrate on machine physics problems. Since all the communication with control system is handled by Databush and Abeans, no special knowledge of the communication protocols and control system is needed.

In addition to the machine model Databush also has a toolkit with GUI and calculation components, designed to be independent or dependent on Databush only in certain aspects. They can display calculation results or perform various correction algorithms and similar. Following the same paradigm as Abeans, Databush also reuses as much code from other libraries as well as within itself. With modular structure, it is easy to create new machine physics applications from existing components or extend existing application for new tasks. For example, lattice loading modules have been made to load lattice from Databush native file structure, from MAD input file (both notations) or from XAL hierarchical structure.

### *Databush at ANKA*

Databush was primarily developed for ANKA accelerator. Key emphasis was on fast calculation of linear optics based model in constant energy situation. From the beginning it was designed with the same spirit as Abeans, to be extensible and as modular as possible. GUI interface of Databush application was designed to be understandable and intuitive for operators, who have no physical background. For instance, OrbitCorrection application is used in everyday operation at ANKA by technician in manual or automatic mode.

The following applications was designed for ANKA and are used in everyday operation:

- *OrbitCorrection* is an extensible application framework with modules for various beam optimization tasks: closed orbit correction, correctors strength reduction, dispersion correction, 3/4 corrector bump.

- *OrbitDisplay* shows chart with BPM measurements and lattice synoptic display.
- *MachineFunctionsDisplay* shows the results of linear optics calculation with one second update rate on the chart with lattice synoptic display.
- *MachineFunctionsManipulator* is similar to MachineFunctionsDisplay with slider controls for lattice parameter manipulation. Machine model is updated from the machine on demand and calculation result (power supply currents) can be set back to the machine on demand as well.

## *Databush and XAL at SNS*

The Spallation Neutron Source (SNS) is an accelerator-based neutron source being built in Oak Ridge, Tennessee, by the U.S. Department of Energy. The SNS will provide the most intense pulsed neutron beams in the world for scientific research and industrial development. The SNSs machine consists of front-end system which produces a pulsed beam of negative hydrogen (H-) ions at energy $2.5MeV$, a large linear accelerator with normal conducting and superconducting radio-frequency cavities, which accelerates the H- beam up to $1GeV$, an accumulator ring structure and target area with experimental lines.

SNS is using a Java based framework for accelerator physics programming interface for the accelerator, called XAL [2]. XAL provides Java wrapper for underlying connection to the EPICS control system. An on-line model is included in this framework for quick beam tracking. Online model uses powerful and flexible calculation model for calculation of machine physics parameters which includes important higher level effect on proton beam in complex SNS accelerator.

At SNS, OrbitDisplay Databush application is running with Abeans EPICS plug. A bump application has also been made and will be tested when enough of the linac transfer line will be operational. Databush application at SNS will gradually be transfered to XAL framework. Databush has a fast and lightweight linear optics machine model and will be replaced with XAL's on-line model, which presents far better fit for the complex accelerator structure at SNS. GUI features, tools and physical algorithms from Databush will help make XAL a better machine physics package.

## ABEANS

The Abeans contains a wide range of Java tools which cover communication modelling, applications framework, GUI widgets, services, resource handling and more. Abeans library is reused and improved with each new Java project Cosylab deploys [1]. It was natural for Databush to use as many features from the Abeans toolbox as possible, when it was transfered to the new Abeans release.

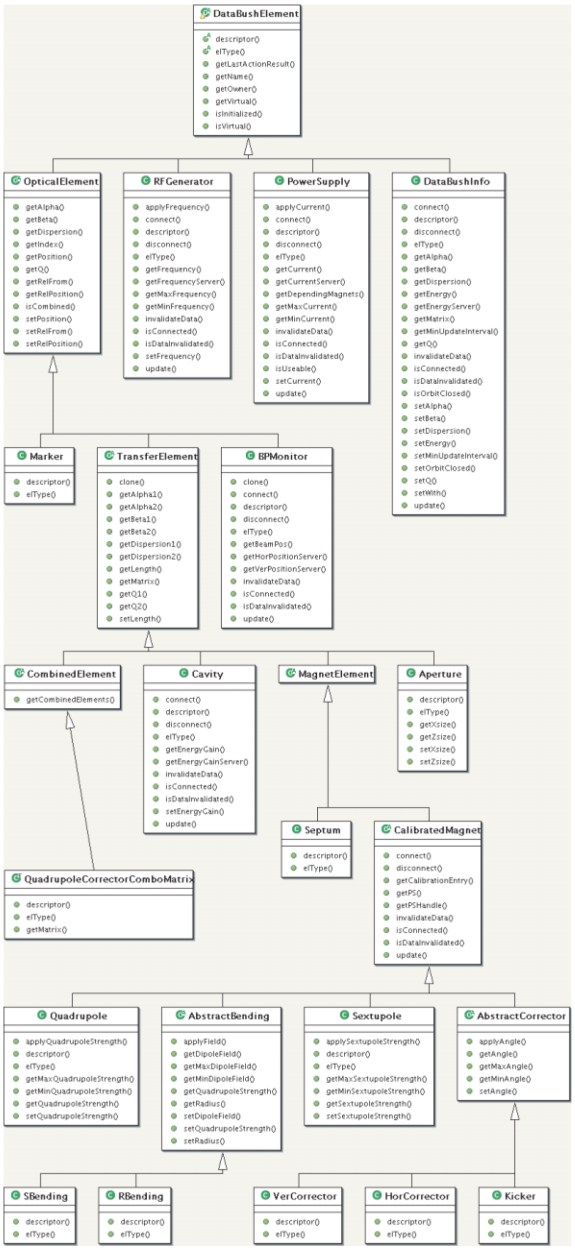

Figure 1: UML diagram of Databush element hierarchy.

Firstly, Databush uses Abeans as data access library to connect to remote data sources. Abeans provide the building blocks for constructing an object-oriented (OO) representation of a complex control system. Such representations are called *Models*. For example, *Channel* model is an OO representation of a flat control system, such as EPICS or TINE. In Channel model, a whole control system can be seen as an assemble of Channels- single sources of dynamic remote values. Implementation of a particular communications protocol is in Abeans called a *Plug*. Plug is an OO interpretation of a narrow interface. Model maps all remote operations to plug Plug as *Request*, for which the Plug returns *Response* with result of remote operation (figure 2. With this powerful approach, one model can use several Plugs (communication protocols) at the same time, or one plug can serve as communication component for several different models. Databush uses Channel model for access to EPICS channels to read magnetic fields and BPM measurements.

Databush also uses Abeans for the following tasks: error reporting, logging, resource loading end configuration management. These features are implemented as shared services in Abeans and are implemented once and for all.

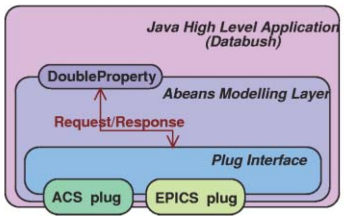

Figure 2: Abeans component diagram.

## *Datatypes*

The Datatypes library defines a set of interfaces and abstract classes that provide a consistent and extensible mechanism for access to dynamic data sources ([3] and [1]). A dynamic value is a value obtained from a data source, either local or remote, that has a unique name in the namespace of all dynamic values accessible to a Java application. Dynamic value has a well-defined type, its value can change through time and interested parties may be informed about the change. Dynamic value is described by a static context, consisting of a number of characteristics. The dynamic value can be cast into different Java types. Property incorporates concepts as characteristic, subscription (to notifications about the change in the dynamic value, the quality of the dynamic value, the status of the subscription itself) and data access (a rendering of the dynamic value into a specific Java type). Figure 3 shows UML diagram of property describing double dynamic data source as `DoubleAbstractProperty`.

The Datatypes library can be used as a set of stand-alone interfaces, because it does not depend on other Abeans libraries and is defined purely as a set of interfaces or abstract classes. Databush uses datatypes as the second level of data-flow abstraction. For Databush, Channel model with EPICS plug is just one of implementation of dynamic data access. Databush uses a dummy implementation of Datatypes interfaces for simulation purposes.

## *EPICS Plug*

Implementation of EPICS communication protocol as Abeans Plug enables object-oriented access to EPICS channels. At the moment, Channel library is used as a facade to this plug, however any other type of model, such as Device/Property paradigm oriented, could be provided. With future development of Abeans also model based on Device/Property paradigm will use same plug for communication with EPICS. This will allow to model the flat hierarchy of EPICS as a set of hierarchically organized devices with properties and thus present EPICS as a fully object driven structure for Java programming. EPICS plug was developed for SNS project and used in ObjectExplorer generic channel browser application.

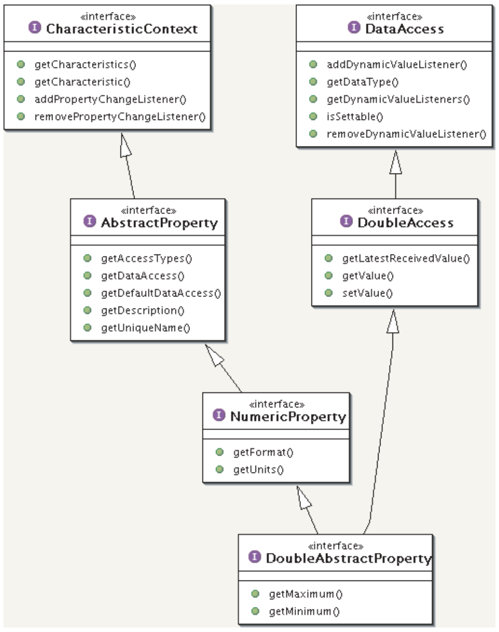

Figure 3: UML diagram of double Property, part of Datatypes framework in Abeans.

## CONCLUSION

Databush machine physics application has proven to be useful and user friendly for ANKA operators. And they were fun to work with. In combination with Abeans and latest improvements, Databush is truly becoming cross platform and cross accelerator bunch of machine physics libraries and applications. With ongoing effort, experience and code accumulated with Databush will be used at two currently largest accelerator projects, SNS and Diamond.

## REFERENCES

[1] http://www.cosylab.com

[2] http://www.sns.gov/APGroup/appProg/xal/xal.htm

[3] I. Verstovsek et al, “The New Abeans and CosyBeans: Cutting Edge Application and User Interface Framework”, PCaPAC’02, Frascati, October 2002