BEYOND ABEANS

Igor Kriznar, Jaka Bobnar, Cosylab, Ljubljana
Matthias R. Clausen, Philip Duval, Wu Hong Gong, DESY, Hamburg
Günther Fröhlich, GSI, Darmstadt

Abstract

Java Abeans libraries have been successfully started in 1999 as part of ANKA control system. The goal was to provide a universal solution for building high level control system applications in Java for any control system. The arrival of Java 1.5 in 2005 was an excellent opportunity to review Abeans and CosyBeans (GUI components and widgets part of Abeans). Cosylab has put experience and new features of Java 1.5 into new projects which superseded what has been done so far by Cosylab. The key element for success of the projects is the collaboration between different laboratories. The CosyBeans components have found their usefulness as a base for development of ACOP GUI components for TINE at DESY. Similarly Abeans' non-visual libraries were replaced by DAL (Data Access Library) and CSS (Control System Studio) projects developed in collaboration with DESY. DAL was also successfully used at GSI, Darmstadt, to model device layer on top of middle-ware CORBA layer. New Java applications were build with DAL and renewed CosyBeans components and are already used in commissioning of new beamline at GSI.

ABEANS

Abeans version R3 is a library that provides simple Java beans for connection with the control system when building control applications. At the same time it provides several useful services: logging, reporting, exception handling, configuration and data resource loaders, authentication and policy management.

Because Abeans are designed to run multiple applications in a single JVM (Java Virtual Machine), libraries are loaded only once and the memory footprint is modest compared to applications running in separate JVMs. This feature also allows the same applications to be run individually or from within an applet in a web browser.

In Abeans, different models are used to represent the structure of the control system. Models use plugs to get data from a specific control system. There are two models: Abeans “device” model for controls systems like ACS, where connection is made to device CORBA objects; and Abeans “channel” model (i.e. a narrow interface access model), which consists of simple Channel classes, to create a plug connecting to a single process variable or channels.

Meta information about the control system, such as its namespace hierarchy and property types is stored in the Abeans Distributed Directory which is an implementation of the Sun’s JNDI (Java Naming and Directory Interface) which provides standardized namespace browsing and searching functionality for different architectures.

Abeans are being used in production environment on several sites, sometimes from the beginning:

- ANKA (Germany)
- Diamond (UK)
- UVSOR (Japan)
- ASP (Australia)
- Spring-8/JASRI (Japan)

As long as this sites will be running Abeans support will be provided. In case of necessity for features which are not available in Abeans there is a scenario for easy upgrade to latest CosyBeans and DAL libraries [1].

Abeans Strong Points

During years of development and improvement of Abeans a lot of experience has been collected inside several libraries, which include Abeans and CosyBeans open source products. Bellow is a table which was made in year 2002 by making a crude effort estimation in man-years from number of code lines.

Table 1: Effort for development of Abeans and CosyBeans libraries till year 2002.

Product	*Lines of Code*	*Estimated Work*
Common	7,343	1.58 my
Abeans	26,715	6.30 my
CosyBeans	44,931	10.54 my
Integration	4,214	0.91 my

Abeans are especially successful in this areas:

- Building application can be very effective with professional looking GUI components. With little effort Abeans and CosyBeans components allow building an application which has professional grade features and look & feel. All components are available under open source terms.
- Reuse of components, design patterns, and more: experiences and best practices gained by Cosylab. It was always policy of Abeans that it should adopt any new design pattern or feature which had proved successful in practice of control programming. Not only code itself but also enhanced programming practices are important part of Abeans framework.
- Collaboration with and contributions from many sites: ANKA, ESO, SNS, DESY, RIKEN. Abeans are unique in a way that they were made in multi-institute collaboration, which was distributed in time scale and effort goals. Each lab has contributed its own view on challenges

that Abeans had to solve and of course own solutions, which somehow had to fit into Abeans concepts. Binding point was Abeans team with their ideas and expertise.

Abeans Weaknesses

Two major weaknesses came obvious while working with Abeans. This two weaknesses seems to be general to any framework:

- Constraints: building has to be done in a certain way to accomplish the task.
- Complexity: maintaining a complex framework is expensive.

In order to use all strength of a framework a programmer should solve certain problems in certain ways. And this must be learned. Abeans framework can solve very complex problems in an elegant and simple way, like interfacing several control systems at same time. But there is certain cost to this. One is that some simple tasks need a little more effort, for instance you have to extend certain classes to make a simple application.

When a programmer wants to use only certain features in a way not foreseen by the framework, then framework may appear limiting in ways in which it can be used.

Abeans were ahead of it's time for many Java features. Logging and exception services for example became part of official Java only after they have been developed especially for Abeans. At some point Abeans need to be upgraded to latest Java and possibly merge or replace Abeans special services with the one available in standard Java JDK.

BEYOND ABEANS

Development of Java technology moved forward to version JDK 1.5 and latest 1.6 with important new changes in Java language capabilities and performance improvements. The Following considerations influenced decision what to do with Abeans in the next step:

- Abeans works well and are stable in Java 1.4.
- A lot of features which were parts of Abeans framework are now available in recent Java technology. For example multi JVM instances can share native resources so framework for sharing JVM is not necessary any more in Java 1.5.
- To use new the Java capabilities most efficiently (like generics and enums) Abeans API should be rewritten.
- There are several libraries inside Abeans, especially parts of CosyBeans, which were interesting for use outside of any framework.
- Several application frameworks appeared in recent time like Eclipse RCP (Rich Client Platform), which have huge support from industry and open source community.

Conclusion was that Abeans can safely stay at the current development level, doing their nice job by solving problems for which they were designed. It has always been Abeans' policy to avoid re-inventing and to adopt and use tools and libraries which are already available and reliable. General plan is to further develop and follow or advanced Java technology with those Abeans libraries, which are still unique in the field.

Libraries and technologies, which can be used instead or beside Abeans, with examples are described in the next subsections.

CosyBeans

CosyBeans libraries are open source Java libraries for data visualization [2]. They contain Java widgets especially designed for displaying live data from a control system. They were developed as a GUI part of Abeans. From the beginning the basic widgets were designed as standard Java Beans. They are organized in the following packages:

- Java-Common
  Contains non-visual utilities and common classes.
- CosyBeans-Common
  Contains GUI components and widgets as pure Java Bean objects made in Swing. All widgets are defined here as independent beans which can be used in standard Java visual editors.
- CosyBeans
  Contains widgets from CosyBeans-Common package with addition of Displayer interface, which makes widgets easier to connect with the data providing libraries from control system.
- CosyBeans-DAL
  Extension of Displayers from CosyBeans package which connects them to DAL API.

CosyBeans components, beans and widgets are flexible and easy to use. They can be used just like other Swing components because they have a clean and documented API and they conform to Java Beans specifications. They work perfectly with other Java Beans or Java development environment.

CosyBeans are integrated into ACOP beans at DESY [3] as base for several ACOP components.

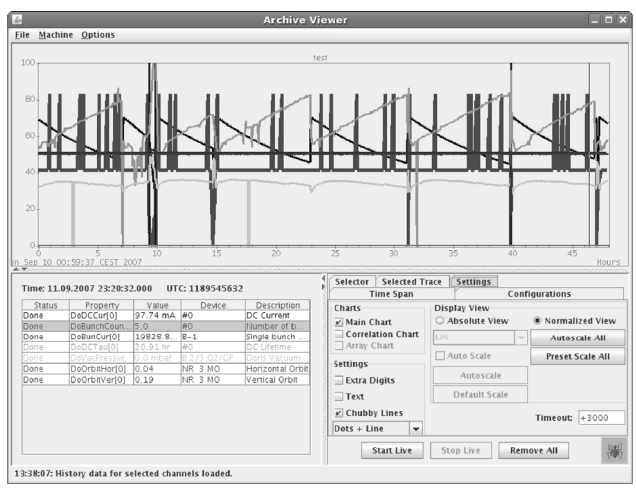

Figure 1: ArchiveViewer for TINE archive server in Java and ACOP in development at DESY.

ACOP Beans

ACOP beans are Java Beans which are developed for connection with TINE control system [4]. They provide customizers and browsers and other tools which help programmer and users to make smooth and easy connection of ACOP components to TINE. They are building blocks for TINE Java control system applications and panels. TINE Java applications are made with pure Swing and Java with NetBeans or Eclipse. ACOP components are available inside visual building tools (NetBeans and Eclipse VE) from custom ACOP palette [5].

DAL and CSS

DAL (Data Access Library) design was based on specifications and input from wide control system community covering several sites and control systems. DAL purpose is to allow access to the control data from different control systems with one well defined and client programming oriented API. DAL allows building generic multi-control-system applications or provide Java wrapper for middle-ware communication libraries, like CORBA.

DAL is a part of CSS (Control System Studio) project [6] which is developed at DESY. CSS is an application suite build around RCP (Rich Client Platform), an application framework from Eclipse. The CSS suite already contains some interesting applications, like PV Probe, strip chart, EPICS archive viewer and synoptic display/editor.

DAL is completely non-visual and is written entirely in Java. Even that it is a part of the CSS it can be used also independently. Experiences and knowledge from the Abeans about multi-control-system interfacing were used in DAL. It allows fast and easy control system integration. Only a couple of days is needed to provide a basic access trough the DAL to any control system. The following control systems have been covered by DAL to various degree: EPICS, TINE, GSI IFC CORBA.

GSI IFC for HITRAP

CosyBeans in combination with DAL were used for building Java table applications and various panels for GSI HITRAP experiment. GSI have its own CORBA based control system, which was wrapped in DAL device interfaces.

CONCLUSION

We decided to keep Abeans as well designed framework at current state of development and focus on CosyBeans and other libraries coming from Abeans and update them up to latest state of Java technology development. One of the goals is also to maintain them in a less framework style. CosyBeans are now successful in a very flexible arrangement in various project as a set of loosely coupled libraries and beans. They can be used independently as single components or together in collaboration with other CosyBeans or general Java Beans.

CosyBeans continue to be open source libraries and are a common code base for many projects at different sites. They are still central point of distributed inter-laboratory collaboration.

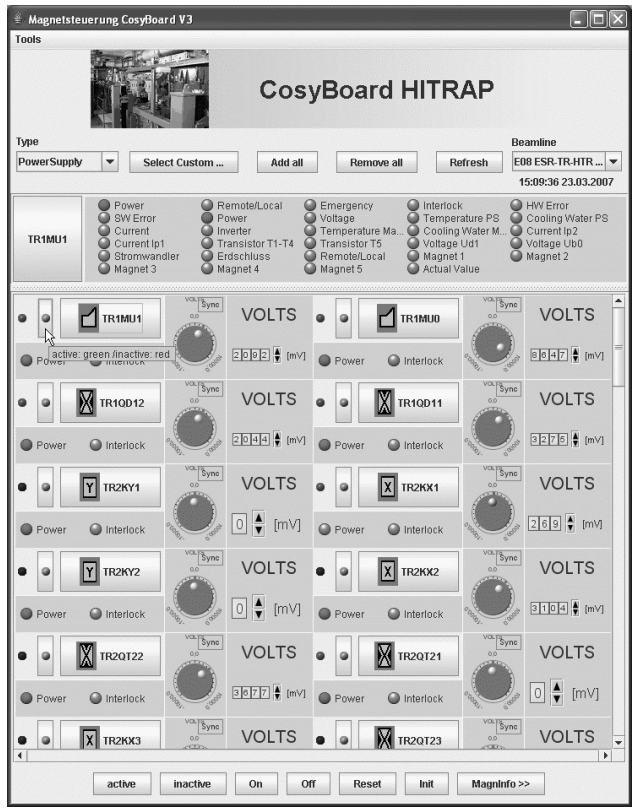

Figure 2: Magnet table application for HITRAP experiment at GSI written in Java with Swing, CosyBeans and DAL.

REFERENCES

[1] Abeans project page (http://abeans.cosylab.com/)
[2] CosyBeans project page (http://cosybeans.cosylab.com)
[3] ACOP project page (http://adweb desy de/mst/acop/)
[4] TINE project page (http://adweb.desy.de/mst/tine/)
[5] P. Duval et al., "New Abeans for TINE Java Control Applications," ICALEPCS 2001, Stanford, Ca;
[6] CSS project page (http://css.desy.de)