DEPLOYMENT OF ABEANS RELEASE 3 FOR THE TINE CONTROL SYSTEM AT DESY

M. Kadunc, J. Kamenik, A. Košmrlj, I. Križnar, G. Pajor, A. Pucelj,
G. Tkačik, I. Verstovšek, D. Vitas, K. Žagar, Jožef Stefan Institute and Cosylab
P. Duval, H. Wu, DESY

Abstract

Cosylab has created Abeans, a Java™ API for easy development and deployment of Control System applications. Abeans is a result of a continuous five-year development process. It consists of visual and non-visual libraries that support retrieval of remote data, offer various graphical visualization components and provide commonly used application services. In collaboration with DESY, a data acquisition module was implemented to use the TINE control system, accessing its device servers, properties, their characteristics, history data and the TINE archive servers. Many new graphical components, such as charting components and tables, have been included and adapted for use in rapid application development tools. Some general-purpose applications, as well as prototypes of specific accelerator control applications were developed. The main benefits of Abeans at DESY are its portability, which allows running the same applications in all the different operating systems that are used at DESY, and the flexible deployment of the applications enabled by the Java™ Web Start technology. The system has been recently upgraded to allow remote access to the DESY control system using Web Services technology.

INTRODUCTION

Abeans R3 is a library that provides simple Java beans for connection with the control system when building control applications [1, 2]. At the same time, it provides several useful services: logging, reporting, exception handling, configuration and data resource loaders, authentication and policy management.

As 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. At DESY we used the Abeans "channel" model (i.e. a narrow interface access model), which consists of simple Channel classes, to create a plug connecting the TINE Java driver to the Abeans.

Meta information about the control system, such as its namespace hierarchy and property types, is stored in the Abeans Distributed Directory [3], an implementation of the Sun's JNDI (Java Naming and Directory Interface) which provides standardized namespace browsing and searching functionality for different architectures. Graphical components for displaying this structure have also been developed.

TINE PLUG

In the past two years the TINE plug for Abeans [4] has been improved significantly. It now supports all the primitive TINE types and most of the complex data types used at DESY. TINE data types are mapped to suitable Java types as shown in Table 1.

Table 1: TINE to Java type mappings

TINE Type	Java Type	seq Type
SHORT LONG BYTE	long	long[]
FLOAT DOUBLE	double	double[]
NAME**	String	String[]
LNGINT	{long, long}	long[][]
DBLDBL	{double, double}	double[][]
FLTINT DBLINT	{Double, Integer}	Number[][]
INTFLTINT	{Long, Double, Long}	Number[][]
NAME*INT	{String, Long}	Object[][]

The plug also supports the new features of the TINE version 3.30 [5], such as overloaded properties, property/device precedence in static characteristics retrieval etc. The plug's performance has been optimized by caching frequently accessed static information about properties and device servers. Access to history data has also been improved and now allows retrieval of properties' short-term history, read from the device servers, or the long-term history data stored in central archive servers (Figure 1).

A significant benefit of connecting Abeans and TINE is the TINE Remote Directory, which is included as a subnode of the Abeans Distributed Directory. Application developers can now access all of the TINE namespace information, as well as structure of the archived properties, through a common interface and display this information easily in graphical user interface.

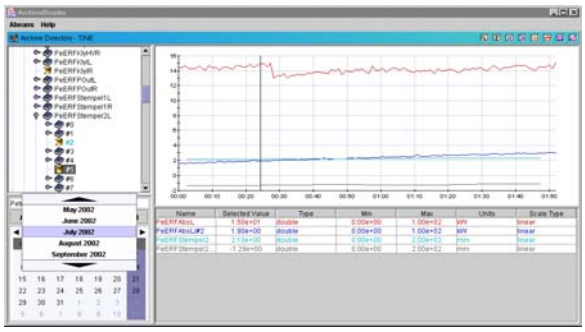

Figure 1: Archive Reader application, using Abeans DistributedDirectory to retrieve archived data

GRAPHICAL COMPONENTS

By using Abeans as a wrapper library around the TINE communication protocol and its namespace hierarchy we are able to display the data in generic graphical components, which we call *displayers*. They implement a standard interface through which they obtain values of control system properties (*dynamic values*). They can also obtain static characteristics, such as minimum and maximum values, to adapt their visual appearance for the given property.

The displayers that have been previously used in Abeans R2 have been completely rewritten and adapted for use in the new release of Abeans. These are Gauger, Slider and Ledder. New components, developed speciffically for Abeans Release 3, are:

- Wheelswitch – allows changing values one digit at a time.
- Piper – displays the value as a vertical bar.
- LabelDisplayer and NumberField – display the value in its textual representation.
- DialKnob – allows changing the value by rotating a circular knob using the mouse

All the graphical components use the latest Java graphical features, such as gradient fills, anti-aliasing and partial transparency. These advanced features can be disabled to improve application performance.

We have written our own pure Java charting component, called SpikeChart, the implementation of which specializes in performance rather than features. For advanced features, such as zooming, selecting data and enhanced visual appearance, we use Sitraka's commercial JClass Chart component. Both charting components share the same data access interface and can thus be interchanged during development or even in run-time, which eliminates performance vs. functionality trade-offs that otherwise have to be made when designing charting components. An attempt has also been made to connect the Java version of DESY's ACOP chart component [6] to the same interface.

A table component which uses similar data access interfaces was also developed to facilitate display of large sets of properties. It can be used to display and manipulate sequential properties as well as arrays of simple dynamic values.

Apart from control-system-specific graphical components, we created some other general-purpose widgets for use in applications, such as enhanced tree control, about dialog, resizable label etc.

ABEANS AS A WEB SERVICE

We have developed technology which allows us to access TINE data from the web without extending the control system outside the controlled network environment of the DESY institute.

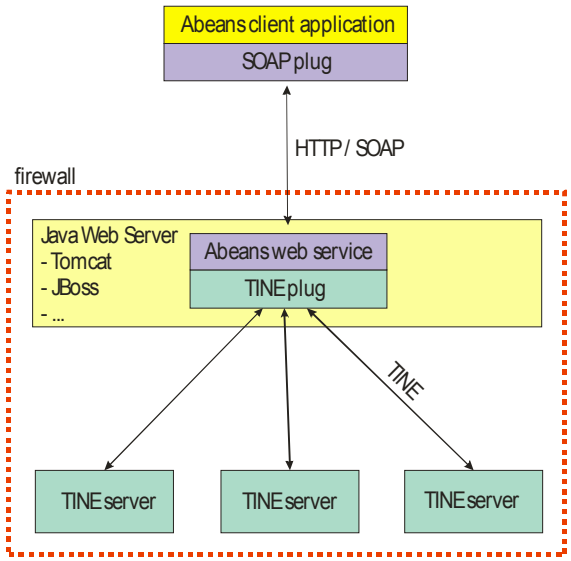

Figure 2: Architecture of Abeans Web Services

A thin communication layer has been introduced between the client application and the TINE servers. This layer is split into two parts, one of which runs on a server inside the DESY network, and the other in the client's Java virtual machine. We decided to use HTTP protocol for communication between the two parts, as it is widely supported in most of the operating systems, programming languages and network applications, and is allowed to pass through the firewall. SOAP (Simple Object Access Protocol [7]) was used for serializing and deserializing the data. The client part of the connection was implemented as an Abeans Plug to achieve effortless migration of existing control system applications, running in DESY, to the new communication protocol. The server part was implemented using Abeans connected to the device servers by the TINE plug. In order to simplify implementation of the network communication, the server application is deployed as a Web Service [8] and runs inside a Java Web Server such as Jakarta's Tomcat [9].

The implementation of the new layers was fairly simple, mostly due to the flexibilty and simplicity of the data communication engine in Abeans. Most of the functionality could be achieved by converting requests to some serializable form on the client side, unpacking them and forwarding them to the TINE plug on the server side. More effort had to be put into retrieval of TINE's namespace hierarchy and information about properties, which Abeans holds in its DistributedDirectory. Since

SOAP WebServices do not support asynchronous calls, updating of the values had to be implemented with polling.

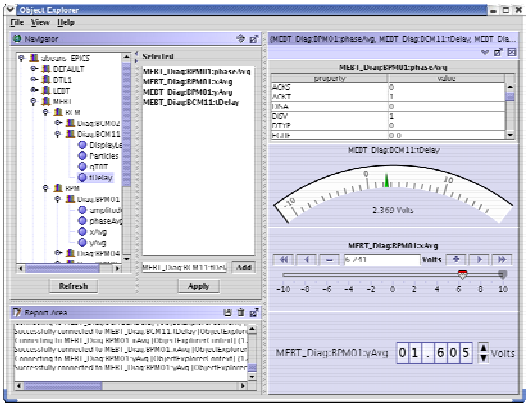

Figure 3: Object Explorer application

Results of this year's developments are encouraging: Complex applications, which were designed for controlling the accelerators from inside DESY using TINE Abeans plug, run equally well by using the HTTP/SOAP. We can also run generic applications, such as Archive Reader (Figure 1) and Object Explorer [10] (Figure 3) without any noticeable differences. No changes have to be made to the applications or the core Abeans library to achieve this, one just has to change the plug used by the applications in a configuration file. Performance of the existing implementation is acceptable, but many things could be done to improve it without losing any of the functionality:

- Abeans server could act as a proxy for monitoring the dynamic values on the server, offering its dynamic value data to multiple clients and thus minimizing communication inside the DESY network.
- The polling that is currently done by the client applications should be grouped, retrieving all the updates in one remote call.

Performance limitations of the current implementation are mostly due to the complex serialization of objects to the XML SOAP protocol. The Web Services technology grows more and more popular, and solutions that overcome this problem are constantly being developed. One of these is Fast Web Services initiative [11], whose solutions are reported to be up to 10 times more efficient than existing ones. Because our implementation is not critically dependent on the encoding we do not expect any problems with migrating to this or other such technologies, when they are ready.

DEPLOYMENT

The flexible design of the Abeans framework enables us to run applications in different environments – as a stand-alone application, as a Java applet or using Java WebStart – without changing any of the application code.

Java™ WebStart [12] has been used to achieve easy deployment of control system applications. An XML file with a description of the application is made available to the users on a web server, and the application is run simply by clicking a link to that file. When an application is started for the first time, all of the necessary libraries are automatically downloaded to the user's computer and checked for validity. This eliminates the need for manually configuring the classpath, which is often a nuissance. In subsequent launches the application uses locally cached resources, as long as they are the same as those on the server. When new versions are uplodaded to the server, user's applications are automatically updated.

Resources that are needed by the application can be accessed from many different locations using the Abeans' loaders. In this way the application settings can be stored either at the user's desktop machine, inside the JAR files in the application distribution, on a web server or even in a remote database. Locations of the resource files can be specified in the Abeans configuration.

REFERENCES

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

[2] I. Verstovšek et al., "Abeans: Application Development Framework for Java," ICALEPCS 2003, Gyeongju, Korea

[3] G. Tkačik, M. Pleško, "A Reflection on Introspection", ICALEPCS 2003, Gyeongjou, Korea

[4] P. Duval et al., "New Abeans for TINE Java Control Applications," ICALEPCS 2001, Stanford, Ca;

[5] http://desyntwww.desy.de/tine

[6] http://desyntwww.desy.de/acop/

[7] http://www.w3.org/TR/SOAP

[8] http://www.w3.org/TR/wsdl

[9] http://jakarta.apache.org/tomcat/

[10] M. Kadunc et al., "Object Explorer – A Pluggable Generic Testing and Diagnostic Tool", PCAPAC 2002, Frascati, Italy

[11] http://developer.java.sun.com/developer/technicalArticles/WebServices/fastWS/

[12] http://java.sun.com/products/javawebstart/