DRAG AND DROP DISPLAY & BUILDER*

Timofei B. Bolshakov, Andrey D. Petrov, FNAL, Batavia, IL 60510, U.S.A.

Abstract

The Drag and Drop (DnD) Display & Builder is a component-oriented system that allows users to create visual representations of data received from data acquisition systems. It is an upgrade of a Synoptic Display mechanism used at Fermilab since 2002. Components can be graphically arranged and logically interconnected in the web-startable Project Builder. Projects can be either lightweight AJAX- and SVG-based web pages, or they can be started as Java applications. The new version was initiated as a response to discussions between the LHC Controls Group and Fermilab.

OBJECTIVES

Synoptic Displays, first introduced at Fermilab in 2002, were aimed to become the main presentation tool for monitoring of accelerator parameters. The system had a component-based design, an AJAX-based web representation layer, and a convenient Project Builder. It was an illusion that such a combination of modern technologies would make the Synoptic Displays a "killer application" in the ACNET–based control system at Fermilab.

The reality, however, was different. Synoptic Displays become the main presentation tool only in one (though important) department, for the Cryogenic control system. It also did not make its way out of Fermilab.

In Autumn 2006, a LAFS ("LHC at Fermilab Software) collaboration was formed to participate in the design of software for LHC. Besides helping CERN to with software development and sharing out experience in controlling the Fermilab collider complex, the group members had an goal to learn new trends and ideas from CERN. Also, we wanted to reassess our older projects for future use in the Fermilab control system.

Synoptic Displays is one of these projects. It was renamed into "Drag and Drop Builder" and was being greatly refactored.

ARCHITECTURE

The Drag and Drop architecture consists of 4 parts:

- Project Builder,
- Repository of components and projects,
- Runtime Project Engine
- Web tier.

Our goal was to break this architecture into independent parts, an base them on standard software components as much as possible. As a result, some of those pieces can now be used separately from DnD.

The low-level interfaces and the XML file formats were also changed to accommodate new experience acquired since 2002.

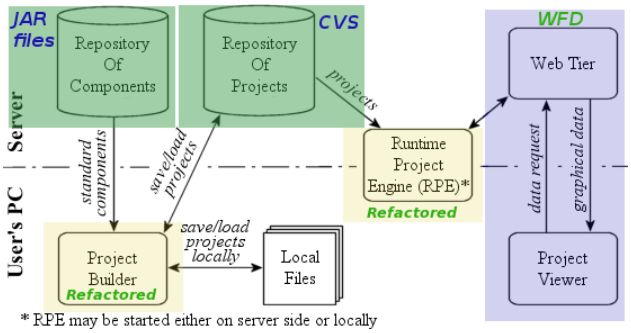

Figure 1: DnD Architecture.

Project Builder

Project Builder is a client-side application dedicated to create and modify DnD projects. It is a special-purpose graphical editor that allows the users to define logical flows of information from data sources to data consumers through data handlers and pipes. The second function of the builder is defining static visual components, such as immutable lines, geometrical shapes, and texts.

In the older version, the Project Builder got descriptions of atomic common-purpose components from the Repository of Components. Projects were stored either in the Project Repository or in local files.

Now, we eliminated the separate Repositories (as described below), and the Project Builder was changed accordingly.

Repositories

The old architecture of Synoptic Displays had two repositories, Repository of Projects and Repository of Components. Both of them were server-side processes. In DnD, the Repository of Components is replaced by a jar file containing all component libraries. The Repository of Projects is replaced by CVS.

Runtime Project Engine

The Runtime Project Engine (RPE) is a central part of the system. It downloads project files from a repository, parses them, creates sets of data acquisition jobs, and builds the resulting images. RPE may be started either locally, or on the server side. In the first case, the result is rendered on a canvas of the application; otherwise a web-tier is used to represent the image on the client side in a browser. Because of security precautions, the system allows device settings only if RPE is started locally inside a specific computer network.

WFD – Web Fixed Displays

The former Synoptic Display web-tier was refactored into a general purpose Web Fixed Display. WFD may be used independently, as a general purpose software tool. Its

*Fermilab is operated by Fermi Research Alliance, LLC., under Contract No. DE-AC02-07CH11359 with the U.S. Department of Energy.

goal is to represent any arbitrary Java AWT- or Swing-based applications remotely in standard web browser. Modern AJAX and SVG technologies are used to suffice that goal. XMLHttpRequest and SVG implementation are greatly available now – both of them are natively supported by popular Firefox web browser.

IMPLEMENTATION

Components

DnD projects are built from components. Component is a simple Java Interface extending Runnable.

So, each component can accept and produce timestamped data and can communicate with other components using BlockingQueues. Components can be classified into several groups:

```
import org.w3c.dom.Element;
import java.util.concurrent.BlockingQueue;
public interface RuntimeComponent extends Runnable{
   public void init(Element root);
   public void start();
   public void stop();
   public void setSink (
      int i,
      BlockingQueue <TimedData>q
   );
   public void setSource (
      int i,
      BlockingQueue <TimedData>q
   );
}
```

Figure 2: Runtime Component interface.

- Active Components – those actually accepting and/or producing data
- Producers – interfaces to underlying control system. Usually invisible.
- Pipes – data processing components, they read data from one or several input queues and writing them to one or several output queues. Usually invisible.
- Presentation components – can represent graphically data.
- Passive Components
- Immutable SVG elements
- "Self–contained" components, those representing it's own data.

When underlying data acquisition system is changed only one library should be reimplemented. Invisible producers and pipes simplifies implementation. Legacy components can be packaged as self-contained components.

BlockingQueues are serving as links between components, defining data flow in runtime.

Data Format

XML has been chosen for projects and components descriptions. XML provides a handy way to keep, convey, and process complex ordered data; it is platform-independent, open, and human readable. In general, projects can be created and modified without Project Builder: for instance, by using custom software for project generation (and that was implemented and used in Synoptic Display) , migration from another system, or with a text editor. Thus, XML makes Runtime Project Engine and Project Builder completely independent.

Project Builder

Project Builder is a graphical editor designed to create and edit DnD Projects.

Builder's graphical user interface is based on Swing classes. Virtually all operations which are usual for graphical editors are supported. In addition, the Builder provides editing of a component's properties, operations with inputs/outputs and logical links (future runtime blocking queues).

Since static components are described as Scalable Vector Graphics elements, a custom SVG rendering module is implemented. This module supports a subset of SVG elements and commands. The same module is used by RPE to show the immutable lines, shapes and texts.

Refactoring of Project Builder was done to accommodate new components description format and to make possible to read components description from component libraries jar files.

The default set of atomic components is loaded from classpath of the builder at program startup and is represented in the tree. Project Builder has a file browser to load DnD projects from the server-side repository, and to save them. In addition, projects can be stored in local files.

Repository of Projects

Repository of Projects is server-side interfaces to CVS, where project files are kept. Repository of components was eliminated – all the necessary component descriptions suppose to be kept in component library jar files.

WFD – Web Fixed Displays

WFD is a Java web application. WAR file containing complete version of WFD can be download from [3].

It includes several Java classes: ApplicationManager, SVGServlet, SVGGraphics2D and several auxiliary classes, couple JavaScript AJAX scripts and html files.

WFD can work independently of DnD. In this mode at the start of this web application ApplicationManager (AM) read properties file and starts client applications described there (with the complete class path). X Virtual Frame Buffer (xvfd) is used on Linux machines or application is started as "headless" on Windows based servers to avoid unnecessary windows on the server environment. AM creates and stores SVG representation of every window created by client applications. AM keeps about 30-60 latest SVG images in round-robin buffers. SVG is created by painting client windows on SVGGraphics2.

When user loads application web-page in browser AJAX Javascript download latest SVG image through SVGServlet and start to update browser picture requesting differences between its current SVG image and latest SVG image available in AM. Because SVG is an XML

that difference is sent as XML, as text. That leads to small network traffic.

CONCLUSION

At the current time refactoring of Synoptic Display is still not completed, but WFD [3,4] is already finished. Refactoring is planned to be finished in December 2007, but that depends on CERN discussion about Drag and Drop implementation details, that is going on around [5].

REFERENCES

[1] T.b Bolshakov, A. D. Petrov, S. Lackey, Synoptic Display — A client-server system for graphical data representation, ICALEPCS 2003, Gyenongju.
[2] Synoptic web-site http://synoptic.fnal.gov
[3] WFD web-site http://www-bd.fnal.gov/wfd
[4] T. B. Bolshakov, E. McCrory, J. Wozniak "Web Fixed Display Requirements" – CERN, EDMS, LHC-C-ES-0009
[5] T. B. Bolshakov , E. McCrory , A. D. Petrov , E. Roux , J. Wozniak "The Drag and Drop Display and Builder Requirements" – CERN, EDMS, LHC-C-ES-0011