SYNOPTIC DISPLAY—A CLIENT-SERVER SYSTEM FOR GRAPHICAL DATA REPRESENTATION

Timofei Bolshakov, Andrey Petrov, Sharon Lackey
Fermi National Accelerator Laboratory, Batavia, IL 60510-0500, U.S.A.

Abstract

Synoptic Display is a Java™ application for flexible online graphical representation of data received from a data acquisition system. It was designed as a part of Fermilab's Accelerator Applications Migration Project [1]. Synoptic Display is considered to be the next generation of such applications as ACNET Lexigraphics and EPICS MEDM. Synoptic Display projects (equivalents of MEDM screens) are rendered on major web browsers (for monitoring purposes) or launched in a web-startable console Java application (for both monitoring and control). Small bandwidth (as low as 100–1000 byte/sec) is required between client and server sides due to usage of Scalable Vector Graphics (SVG) [2] for data transfer. Synoptic Display components (data sources, processing pipes, visualization widgets) can be graphically arranged and logically interconnected in a web-startable Project Builder. Projects are stored in a server-side repository in XML format. A Runtime Project Engine (RPE) handles user requests, downloads projects from the repository, launches data acquisition jobs, and generates SVG pictures. Servlets and Java Server Pages (JSP) are used as RPE web tier. At present time, ACNET Java Data Acquisition Engine is a primary data source for the Synoptic Display, since this is the corporate Fermilab standard; there are no limitations to create new types of data sources. Projects home page is [3].

ARCHITECTURE

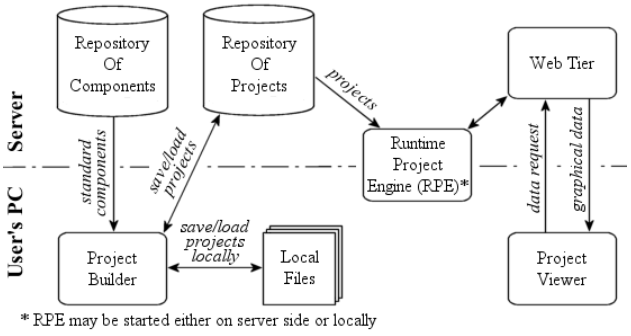

Figure 1: Synoptic Display Architecture.

The system consists of five parts: (1) project builder, (2) project viewer, (3) repository of components and projects, (4) runtime project engine, and (5) a web tier. The first two parts reside on the user's PC. Repositories are on the server side. RPE may be started either on the client or on the server side.

Project Builder

Project Builder is a client-side application dedicated to create and modify Synoptic Display projects. It is a special-purpose graphical editor that allows 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 definition of static visual components, such as immutable lines, geometrical shapes, and texts.

Project Builder works with Repository of Components in order to get a description of atomic common-purpose components. Projects are stored either in the Project Repository or in local files.

Repositories of Components and Projects

Repositories of Components and Projects are server-side program that keep and distribute Synoptic Display projects and atomic project components among multiple instances of the Project Builders and RPEs.

Runtime Project Engine

Runtime Project Engine is a central part of the system. It downloads project XML files from the repository, parses those files, creates a set of data acquisition jobs, and builds the result image. RPE may be started either locally, or on the server side. In the first case, the result is rendered on a canvas of this application; otherwise an additional client-side Project Viewer is required. Because of security precautions, the system allows device settings only if RPE is started locally inside a specific computer subnetwork.

Web Tier and Project Viewer

If RPE resides on the server, the Web Tier, a set of servlets and Java Server Pages (JSP), is employed to convert result images to an appropriate graphic format and pass them to the client-side Project Viewers. The Project Viewer in most cases is an SVG plug-in for a web browser.

Upon the first user request, Web Tier sends the full SVG image to the client. This original image is cached on both client and server. On all subsequent requests, Web Tier sends just a difference between current image and the previous one. Project viewer applies this difference to the cached image and renders it. That way requires a minimum bandwidth and eliminates visual blinking of the picture.

At this time, SVG plug-ins for prevalent browsers are provided free of charge by Adobe System, Inc. and Corel Corporation, but unfortunately the availability is limited for different platforms [4].

If a SVG plug-in is unavailable for a given configuration, Web Tier may generate JPEG or GIF images. This requires a much higher bandwidth and results in visible refreshing of the picture every few seconds.

IMPLEMENTATION

Data Format

XML, Extensible Markup Language, has been chosen as the format for Synoptic Display projects and components. 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, migration from another system, or with a text editor. Thus, XML makes Runtime Project Engine and Project Builder completely independent.

The format of a Synoptic Display project is given by XML schema [5]. Briefly:

- Every project includes three types of components: active components, passive components and links. **Active components** can accept and produce data, and each of them has a specific runtime implementation. **Passive components** are immutable SVG elements. **Links** are used to define logical flows of data between active components.
- Every active component may contain some set of internal components (active, passive and links). Projects themselves are active components.
- Every component has a set of properties. There is a set of required properties for each type of component. Every property has type, name, value, and may have the default value. Currently supported types are: double, integer, string, boolean and color.
- Active components may have inputs and outputs. For certain components the amount of inputs and outputs may be changed at user discretion.
- A Java class responsible for the runtime implementation must be specified for every active component.

Project Builder

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

Builder's graphical user interface is based on Swing classes, however it has been found that the generic behaviour of JComponent and successors differs very much from the desired behaviour of Synoptic Display's atomic components (active, passive, and links) at design time. A complex set of custom components has been developed to implement specific operations of the graphical editor. Virtually all operations which are usual for graphical editors (except zoom operation, so far) are supported. In addition, the Builder provides editing of a component's properties, operations with inputs/outputs and logical links.

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, listed in [6]. The same module is used by RPE to show the immutable lines, shapes and texts.

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

Runtime Project Engine

Runtime Project Engine is an operational environment, used to start Synoptic Display projects. The runtime behaviour of the project is implemented by RunTimePanel class, an extension of JComponent. Though each kind of atomic components included in project can have its own behaviuor, all of them implement a RunTimeComponent interface, as follow:

```
public void init( Element root )
public ObjectPipe getSink( int i )
public void setSink( int i, ObjectPipe pipe )
public void start()
public void stop()
```

Here, ObjectPipe is an object pipe abstraction, Element is a JDOM element [7].

Object pipe implementation classes are created by a factory. Special data classes are travelling through those pipes: TimedDouble, BunchOfTimedDouble, TimedErrors, TimedInteger, and TimedBoolean.

During initialization of RunTimePanel all classes that implement runtime behaviour of included components are being instantiated using *reflection* and interconnected with object pipes.

The existing set of components is divided into three logical groups with different functionality. Each individual component may belong to only one of these groups:

- **Data sources**—system-dependent components provide readings and settings for real physical devices. At this time only ACNET data sources are implemented. System independent test data sources are provided, as well.
- **Data processors** provide conversions (i.e., expression-driven scaling), filtering, averaging, consolidation, data streams distribution, etc.
- **Visual components** provide data visualization. They are Numeric Display, Alarm Display, bar charts, simulations of an oscilloscope, knobs, thermometer, barrel with floating level of liquid, gauge, etc.

Web Tier

Web Tier is a set of servlets and Java Server Pages deployed on the Tomcat Servlet Engine [8].

The heart of the Web Tier is a servlet responsible for loading and launching of Synoptic Display projects. This servlet considers the project as a JFrame with parameters. After such a JFrame is instantiated, it is painted on SVGGraphics, a custom extension of Graphics2D class. SVGGraphics returns a SVG picture as a JDOM tree. Depending on user request, the servlet sends to the client either a zipped SVG document, or just a difference between current SVG document and the previous one. Usage of the differences for high level vector graphics elements leads to very small network bandwidth. On the client side, a short JavaScript code executed in a browser requests those differences and updates the SVG DOM tree inside a SVG plug-in with new values.

Another servlet implements access to the repository. It supplies Project Builder and RPE with XML data of projects and components.

A set of JSPs surrounds those two servlets in order to provide the user interface.

CONCLUSION

At the current time, Synoptic Display provides a complete solution to create, modify, store, and launch data acquisition projects. An open, portable, and human readable data format, based on XML, is used to convey and store projects, components, and graphical data. The central server-side repository is used to store common component library and projects, that allows sharing of data among multiple developers. Projects stored there immediately become available for execution. The graphical result is rendered on the client side using a web browser with a SVG plug-in, and a very small software installation is required.

Three main parts of the system: Project Builder, Runtime Project Engine, and Web Tier can be used independently. Web Tier can be easily rewritten to support other types of JFrames. Runtime Project Engine framework can be extended to meet specific user needs (i.e., simulation) or in order to support different data acquisition standards (i.e., EPICS or industrial SCADA systems). Even all of the Runtime Engine classes can be rewritten completely by other developers to provide desired meaning to the XML project files. XML project files can be generated from JSP pages—this approach was used for the Slow Plot sub-project [9].

REFERENCES

[1] S. L. Lackey, F. X. Zhang , “Fermilab Accelerator Applications Migration Project,” ICALEPCS’03, Gyenongju, October 2003.
[2] http://www.w3.org/Graphics/SVG/
[3] http://www-bd.fnal.gov/synoptic/
[4] http://www-bd.fnal.gov/synoptic/doc/guide/overview.html#2
[5] http://www-bd.fnal.gov/synoptic/schemas/project.xsd
[6] http://www-bd.fnal.gov/synoptic/doc/guide/svg.html
[7] http://www.jdom.org
[8] http://jakarta.apache.org/tomcat
[9] http://www-bd.fnal.gov/synoptic/SlowPlot4.html