PORTABLE SDA (SEQUENCED DATA ACQUISITION) WITH A NATIVE XML DATABASE*

T. B. Bolshakov, E. McCrory, Fermilab, Batavia, IL 60510, U.S.A.

Abstract

SDA is a general logging system for a complex process that evolved in a regular way in time. It has been used as one of the main logging facility for the Tevatron Collider during Run II. It creates a time abstraction in terms understood by everyone and allows for common time ticks across different subsystems. In this article we discuss a plan to re-implement this highly successful Fermilab system in a more general way so it can be used elsewhere. Latest technologies, namely a native XML database and AJAX, are used in the project and discussed in this article.

SDA IN FERMILAB

SDA is an acronym with dual meaning. Originally it was introduced by the Controls department as "Sequenced Data Acquisition" [1], although some think of SDA as meaning "Shot Data Analysis." The word "Sequenced" in SDA signifies that historically most of the events come from Sequencer [5]. SDA has proven to be extremely useful during the tuning of the Fermilab accelerators chain in Collider Run II [2], [4], [5]. It allows for coordinating of effort of different groups across the Laboratory. The main disadvantage of Fermilab SDA is its deep integration into the Fermilab Control System (ACNET). Now we are trying to implement a portable SDA system, based on our experience with it at Fermilab.

GENERAL VIEW

Sequenced Data Acquisition is a logging system for the definition and description of the starting, developing, and finishing of a complex multistage process. Each stage of the process defines a different set of properties and conditions that are collected. The start and stop times of every stage define a common time tick across the system. The difference between SDA and "usual" logging is like difference between CSV (Comma Separated Values) and XML text files.

Shot Data Analysis is a set of libraries, routines and reports that use data from Sequenced Data Acquisition. Shot Data Analysis studies the behavior of some particular subsystem across several stages or the cooperation of different subsystems during some particular stage. As an example we can refer to article [2].

SDA has become an important tool for studying repeatable multistage processes in complex systems. We use it for studying accelerators, but any complex, multistage process can be analyzed with this system, for example, thermonuclear facilities, space rockets, hurricane research, etc.

Terminology

SDA is based on rules. The most important terms of those rules are atom, event, collection, shot, case, and set.

Atoms have a name, a data type, and a request which defines how to collect data. Atoms can be different in different SDA systems. A specific atom can be present in a collection only once.

Events define the time or the condition for data collection. Different control systems may have different events. Events are described in the configuration.

A Collection is a set of atomic values collected on specified events. It delineates the stage of a multistage process. It also has a type and a name. For example collection type 4 for the Tevatron has the name "Inject Protons" in the shot named "Collider Shot".

A Shot contains certain types of collections and the rules to start and stop data acquisition for those collections. It also has a name and a type – shot type 1 in Fermilab has the name "Collider Shot". Shot describe a whole process. An instance of shot (data for particular processes) has a "shot alias" and a "shot index". The shot index is unique across the whole SDA system and is acquired automatically.

Collections in a shot with the same type are called a Case. If a collection is repeated several times the Case may have Sets – several instances of the same collection.

Shots, Cases and Sets define time ticks for the whole process.

IMPLEMENTATION

To implement the system we must define how the structure and data are stored, describe the data acquisition process, create the basic tools for editing the structure, and for viewing and accessing the data. Java was selected as an implementation language for its portability, Object Oriented Design, and rich APIs.

Data Storage

The structure of SDA is hierarchical: a Shot contain Cases, a Case contains Sets (Collections), and a Set contains Atoms. This structure and the data storage lend itself well to XML. An XML Schema has been created to describe the XML for configuration (structure) and for data. In both Schema, details of atoms and events were not specified because they may change from system to system. In the configuration Schema "atom data request", "atom type" and "event" are specified as a string. In data Schema the atom content is left unspecified.

The shot structure (configuration) is an XML document, as is the shot instance (data collected for shot).

XML documents may be stored differently – as a plain file, mapped into Relational database, or in an XML

database. We decided to utilize a native XML database for storing the structure and the data. Berkley XML DB was selected as the database.

Berkeley DB XML is an embedded XML database with XQuery-based access to documents stored in containers and indexed based on their content [3]. Berkeley DB XML is built on top of Berkeley DB and inherits its rich features and attributes. Like Berkeley DB, Berkeley DB XML is a library, not a server; it exposes a programmatic API for developers, and runs in process with the application. Berkeley DB XML supports flexible indexing of XML nodes, elements, attributes and meta-data to enable the fastest, most efficient retrieval of data.

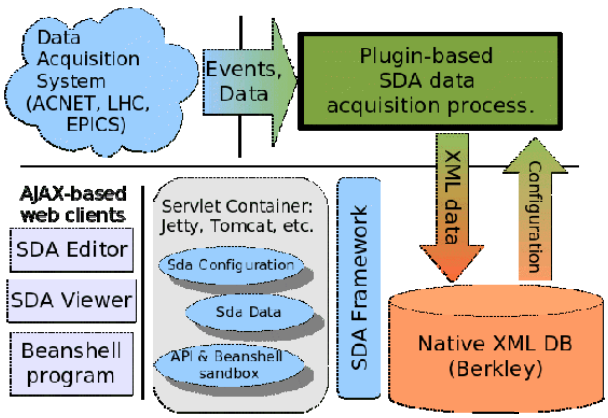

Figure 1: Portable SDA Block Diagram.

Because the actual storage mechanism in the code is defined by a Java interface, the native XML database can be replaced by a relational database. Nevertheless we consider the native XML database as a success – it greatly simplifies the development and has proven to be fast and reliable.

Basic Tools

Basic SDA tools include the SDA Editor and the SDA Viewer. The SDA Editor allows for creating and editing the configuration and the SDA Viewer is used to browse the collected data. Both of these tools allow for plugins in order to allow for Atom data requests and Events to be different for different control systems. The Portable SDA Editor and SDA Viewer are implemented as web applications, based on AJAX (Asynchronous Javascript and XML) - a Web development technique for creating interactive web applications.

Plugins for creating and editing Atom Data Requests and Events are implemented as JSP (Java Server Pages). The basic SDA Editor represents them as strings and cannot verify their validity. Atom data renderer for SDA Viewer is implemented as a Java Interface and can be additionally tuned by providing JavaScript editor on the client side. Web application and AJAX have been selected over Java Web-Startable program because it imposes less limitation on the client computer and, simultaneously, provides more flexibility on the server side. A complete, working SDA Viewer and SDA Editor for Fermilab has been created.

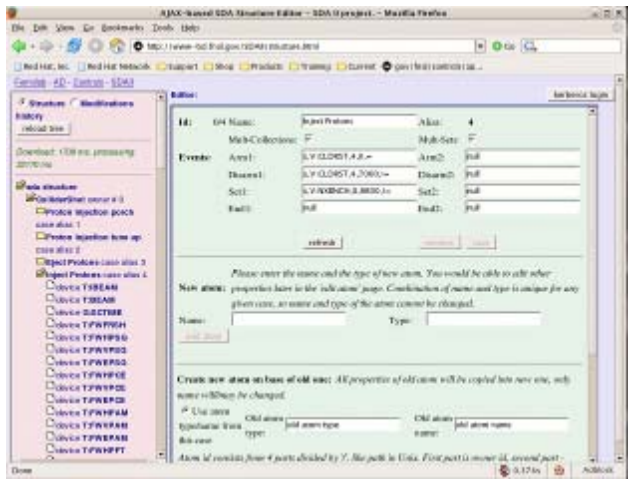

Figure 2: SDA Configuration Editor.

Recently AJAX has become even more attractive because SVG (Scalable Vector Graphics) is natively implemented in Firefox 1.5. Using SVG makes it possible to represent on the Web page any rich graphical information.

Despite the decision to use AJAX, there is still room for web-startable Java programs because data comes as XML over HTTP.

Accessing Data

OSDA (Open SDA) API is used for Shot Data Analysis in Fermilab. This API is simple and easy to learn, but powerful enough to write analysis programs. It was originally based on XML over HTTP and because of that has been easy to integrate into new system. Using OSDA it is possible to write Java programs that access SDA data over HTTP.

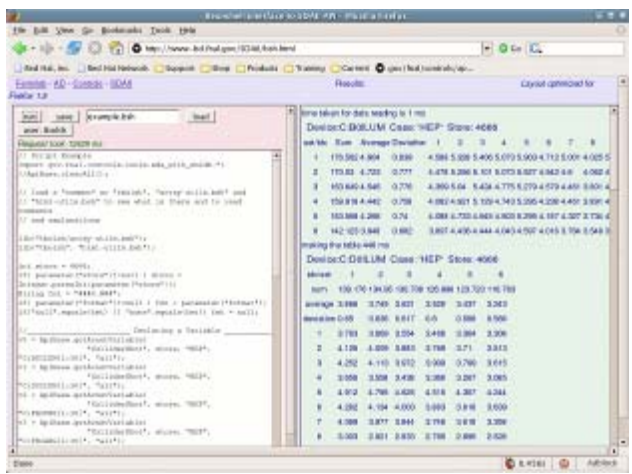

Figure 3: Beanshell sandbox.

In addition an OSDA-like API has been developed that accesses data on the server. Because Berkley XML DB is a library rather then server, this API utilizes direct calls to Berkley DB and is significantly faster then OSDA. To run user programs on the server, a Beanshell [6] sandbox has been implemented. Using Beanshell different types of reports can be generated without compromising security.

Data Acquisition

The Data Acquisition part of portable SDA depends on the control system. Interfaces for Data Acquisition have been designed for portable SDA. The intent is to make it lightweight, flexible, extensible and scriptable (using Beanshell). Usage of SCF (Secure Controls Framework) for the Fermilab implementation will provide support for both ACNET and EPICS.

Testing the System

In order to test SDA applications, the performance of native XML database, the convenience of AJAX and the Beanshell sandbox, all Fermilab SDA configuration data and all SDA data collected in Fermilab Collider Run II for shots with type "Collider Shot" has been imported into new system. You can see the results at http://www-bd.fnal.gov/SDAII .

Fermilab implementation of Data Acquisition is to be tested this summer.

CONCLUSIONS

Portable SDA implements a general and powerful paradigm for describing a multistage processes in a complex, time dependent systems. This new approach is based on our experience during the Fermilab Collider Run II.

A Native XML database significantly simplifies SDA development, because XML is natural representation of configuration and data for such a system.

AJAX implementation provides portability and flexibility for the User Interface.

Critical implementation decisions (usage of native XML DB and AJAX) can be reversed to more standard solutions (relational DB and Java Web Startable applications).

XML DB provides sufficient performance for such an application.

REFERENCES

[1] T.B. Bolshakov, P. Lebrun, S.Panacek, V. Papadimitriou, J. Slaugher, A. Xiao (Fermilab), "SDA-based diagnostic and analysis tools for Collider Run II," PAC'05, Knoxville, USA, May 2005.

[2] A. Xiao, T. Bolshakov, P. Lebrun, E. McCrory, V. Papadimitriou, A.J. Slaughter, "Tevatron beam lifetimes at injection using the Shot Data Analysis system," PAC'05, Knoxville, USA, May 2005.

[3] http://www.sleepycat.com/products/bdbxml.html Berkley XML DB Documentation.

[4] T. Bolshakov, K. Genser, K. Gounder, E. S. McCrory, P. L. G. Lebrun, S. Panacek, V. Papadimitriou and J. Slaughter, "Data acquisition and analysis for the Fermilab Collider RunII", ICAP'04, St Petersburg Russia, July 2004.

[5] The Fermilab RunII Handbook, at http://www-ad.fnal.gov/runII/index.html

[6] Beanshell reference, at http://www.beanshell.org