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

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

Abstract

SDA is a general logging system for a repeated, complex process. 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 tick across different subsystems. In this article we discuss a plan to re-implement this highly successful FNAL 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 the article.

SDA IN FERMILAB

SDA is an acronym with dual meaning. Originally it was introduced in the Controls department as "Sequenced Data Acquisition" [1]. The Integration department uses it as "Shot Data Analysis" and this reading of the term became more popular. The word "Sequenced" in SDA signifies that historically most of the events come from Sequencer [5]. SDA proved to be extremely useful during tuning Fermilab accelerators chain in Collider Run II [2], [4], [5]. It allows for coordinating of effort of different groups across the Laboratory. Main disadvantage of Fermilab SDA is deep integration into Fermilab Control System (ACNET). Now we are trying to implement the portable SDA system, based on experience, collected in Fermilab.

GENERAL VIEW

Sequenced Data Acquisition is a logging system for description of starting, developing, and finishing of complex multistage process. On every stage of the process different set of properties and conditions are collected. Start and stop time of every stage define 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 study the behavior of some particular subsystem across several stages or cooperation of different subsystems during some particular stage. As an example we can refer to article [2].

From this perspective SDA become an important tool for studies of repeatable multistage processes in complex systems like accelerators, thermonuclear facilities, space rockets, hurricane research etc.

Terminology

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

Atoms have name, data type, and request, defining how to collect data. Atoms can be different in different SDA systems.

Events define time or condition for data collection. Different control systems may have different events.

Collection is a set of atomic values collected on specified events. Events are described in the configuration. One particular atom can be present in the collection only once. Collection describes the stage of the multistage process. It also has type and name. For example collection type 4 has name "Inject Protons" in shot named "Collider Shot".

Shot contains certain types of collections and rules to start and stop data acquisition for those collections. It also has name and type – shot type 1 in Fermilab has name "Collider Shot". Shot describe a whole process. An instance of shot (data for particular processes) has "shot alias" and "shot index". Shot index is unique across the whole SDA system and is acquired automatically. Shot alias is specified by the operator in the beginning of the shot. So, there may be several shot with the same alias (for example if some of them where unsuccessful).

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

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

IMPLEMENTATION

To implement the system we should define how structure and data are stored, describe data acquisition process, design basic tools for editing structure, viewing and accessing data. Java was selected as an

*by Universities Research Association, Inc., under Contract No. DE-AC02-76CH03000 with the US Department of Energy.

implementation language for its portability, Object Oriented Design, and rich APIs.

Data Storage

Structure of SDA is hierarchical: Shot contain Cases, Case contains Sets (Collections), Set contains Atoms. Obvious representation of such structure is XML. That is true for both structure (configuration) and data. XML Schema's was created to describe XML for configuration (structure) and data. In both Schema's details of atoms and events were not specified, because they may change from system to system. In configuration Schema "atom data request", "atom type" and "event" are specified as a string. In data Schema atom content is leaved unspecified.

Shot structure (configuration) is an XML document. Shot instance (data collected for shot) is also an XML document.

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

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

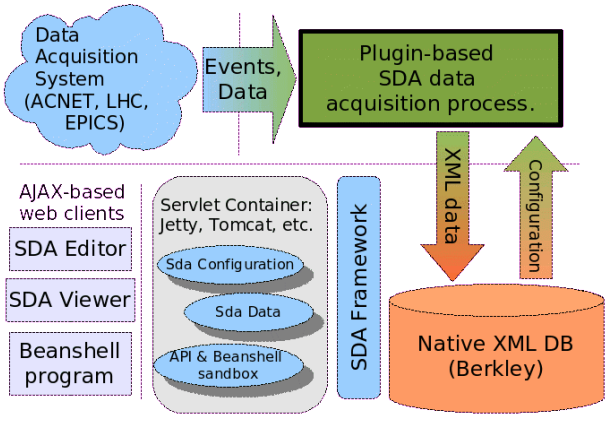

Figure 1: Portable SDA Block Diagram.

Because actual storage mechanism in the code is defined by Java interface, native XML database in the future can be substituted by relational database, despite we consider usage of XML database as a success – it does greatly simplify the development and proved to be fast and reliable.

Basic Tools.

Basic SDA tools by our opinion include SDA Editor and SDA Viewer. SDA Editor allows for creating and editing configuration and SDA Viewer is used to browse collected data. Both of those tools should allow for plugins, because Atom data requests and Events should be different for different control systems. Portable SDA Editor and SDA Viewer were implemented as web applications, heavily based on AJAX (Asynchronous Javascript and XML) - a Web development technique for creating interactive web applications.

Plugins for creating and editing of Atom Data Requests and Events were implemented as JSP (Java Server Pages). Basic SDA Editor represents them as string and cannot verify validity. Atom data renderer for SDA Viewer was implemented as Java Interface and can be additionally tuned by providing Javascript editor on client side. Web application and AJAX was selected over Java Web-Startable program because it imposes less limitation on the client computer and, simultaneously, provides more flexibility on the server side. Complete working SDA Viewer and SDA Editor for Fermilab was created.

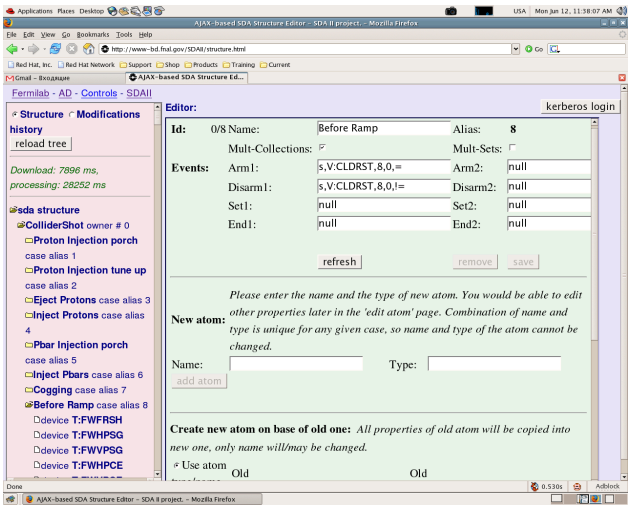

Figure 2: SDA Configuration Editor.

Recently AJAX becomes even more attractive because SVG (Scalable Vector Graphics) is natively implemented in Firefox 1.5. Using SVG makes 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 was easy to integrate into new system. Using OSDA it is possible to write Java programs that access SDA data over HTTP.

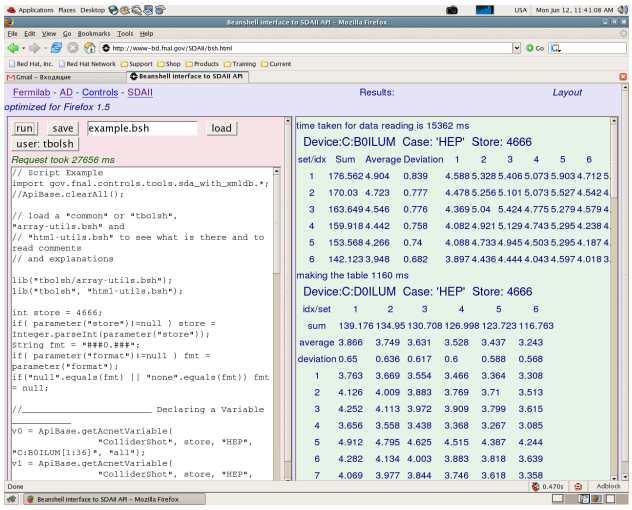

Figure 3: Beanshell sandbox.

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

Data Acquisition

Data Acquisition part of portable SDA depends on Control System. Interfaces for Data Acquisition were designed for portable SDA. The intension was to make it lightweight, flexible, extensible and scriptable (using Beanshell). Usage of SCF (Secure Controls Framework) for Fermilab implementation will provide support for ACNET and EPICS.

Testing the System.

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

Fermilab implementation of Data Acquisition supposes to be tested this summer to determine its weak and strong sides.

CONCLUSIONS

- Portable SDA implements a general and powerful paradigm for describing multistage processes in complex systems. The approach is proved by Fermilab Collider Run II experience.
- Native XML database simplified significantly SDA development, because XML is natural representation of configuration and data for such a system.
- AJAX implementation provides portability and flexibility for 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) because of the modular structure of the system.
- XML DB provides sufficient performance for such an application.
- Beta versions of portable SDA tools are available on http://www-bd.fnal.gov/SDAII

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