Operational Perspective on Maintaining the Java-Based Shot Data Analysis System for the Tevatron Complex

E. McCrory[1], T. Bolshakov[1], V. Papadimitriou[1], A. J. Slaughter[1], A. Xiao[2]
[1]*Fermilab, Batavia, IL (USA);* [2]*Argonne National Laboratory, Lemont, IL (USA)*

ABSTRACT

The Tevatron's Shot Data Analysis (SDA) system is based on a data acquisition mechanism crafted specifically to collect data generated during the course of Tevatron proton/antiproton stores, including their setup. SDA is responsible for calculating important store quantities, which experts at Fermilab study in order to improve the performance of the Tevatron. These results are archived permanently in databases and in user-accessible summary files. This paper presents an overview of this system and a perspective on the operational issues relevant to keeping it up and running.

INTRODUCTION

The operation of the Fermilab Tevatron Complex depends strongly on manipulating antiprotons. The process of removing antiprotons from an antiproton source and directing these particles to be stored and to collide with protons in the Tevatron is complicated and is subject to many variances. This process is called a *shot setup*, and the event that represents this is called a *shot*. The term *store*, which is often used interchangeably with shot, refers to the results of the shot setup in which we provide proton/antiproton collisions for the physics experiments. Early on, we began storing data collected during a shot into relational databases indexed by the sequence number of the shot and by the markers that define the time evolution of the shot. We have also stored summaries of these data into simple, user-accessible tables. These databases and tables have proven to be very useful.

An important part of this system is the Java packages that have been developed to manipulate, analyze and display shot data. The packages perform calculations that are either too complicated for the data acquisition systems to perform or that evolve with our understanding of the Tevatron Complex.

The acronym SDA has several meanings at Fermilab. One is "Sequenced Data Acquisition:" The system responsible for most of the data acquisition during a shot. Another is "Shot Data Analysis:" The set of high-level analysis tools that analyze the data collected by Sequenced Data Acquisition. This latter interpretation is the one implied in this paper.

One of the most useful products of SDA is the Supertable, which is a summary of the best available information about the shots and stores. It is available as a spreadsheet, an AIDA file (for use in Java Analysis Studio [1]) or as an HTML document. It is updated three times during every shot.

The outline of this paper is as follows. First, we describe the structure of SDA: The databases, programs, and summary tables, including the Supertable. Then some of the architecture of the Java class packages are discussed. Lastly, we describe the issues we have encountered, and the lessons we have learned for this system [2].

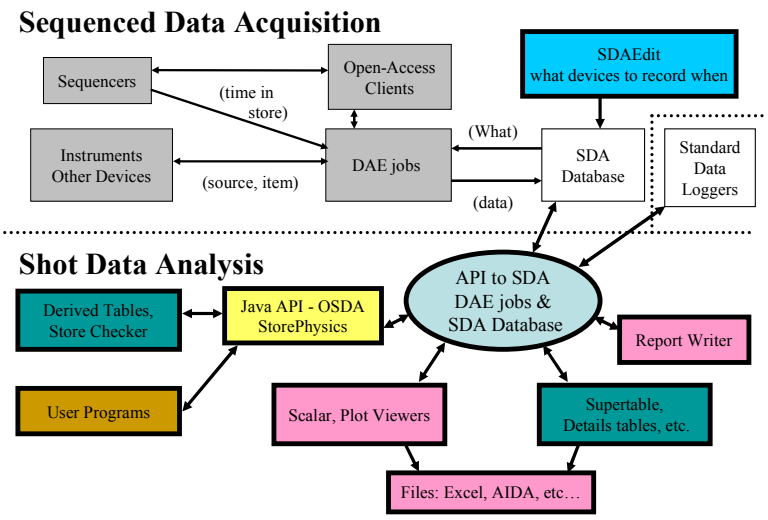


Figure 1, Overview of the SDA System.

SDA OVERVIEW

The architecture of both SDA's is shown in Figure 1. All low level data acquisition tasks are performed by DAE (Data Acquisition Engine) Jobs [3]. The SDA database is filled through the rules created in SDAEdit. The analysis portion of the system is centered on the Java API in the OSDA and StorePhysics packages [4]. DAE Jobs are also responsible for run-

ning these classes at the appropriate time and seeing that the relevant tables are created (in particular, the Supertable). All of the programs here are written in Java.

Sequenced Data Acquisition

The Java data acquisition scheme invented for Tevatron Collider Run II makes abstract several basic concepts: *Source* (where the data come from), *Disposition* (where are the data go to), *Item* (what facet of the Device is required) and *Event* (when the data should be collected). A DAE Job defines concrete instances of these four abstractions, in addition to the user-supplied actions to be taken by the job. Sequenced Data Acquisition is a DAE Job. It additionally has the concepts of *collection* (of Devices) and *shot*. For the work described in this paper, the Disposition of the SDA DAE Job is the SDA database: The data are collected automatically, based on events during a shot.

The results of the SDA DAE Job are stored in the SDA database, indexed by the shot number and the collection. *Case* is the word used to describe a collection that is gathered at a specific time during a shot. A case is identified by a name or a number, for example Case 6 for a collider shot has the name "Inject Pbars." A collection may be gathered multiple times during a case, and this is called a *set*. During the case "HEP" ("High-Energy Physics;" when we are delivering colliding beams to the high-energy physics experiments), a new set is generated every ten minutes. Since our stores last up to 40 hours, there will be up to 240 sets in an HEP case.

A Java tool has been created to examine the stores, cases and sets collected by Sequenced Data Acquisition. This general tool facilitates the examination of the integrity of the database and it also provides some rudimentary data analysis mechanisms. A screen shot of this application after selecting a specific shot number is shown in Figure 2. Clicking on one of the cases shows the sets recorded for that case. When you click on a set and select "get data," a new window appears with the data gathered for that set.

In summary, Sequenced Data Acquisition defines a clock that marks the time-evolution of a store, from shot setup to collisions to the end of the store. Consequently, the store number may be translated to a range of times. The cases always happen in the same order; for example, Case 2 always follows Case 1. Furthermore, the numbered sequence of sets always falls in order. Notice in Figure 2 that the time stamps for the Cases are in chronological order. While the actual time and the duration of a case or of a set may change, the order of them is always fixed, and therefore defines a reliable and useful clock for a shot. This clock abstraction has been applied to other important sequential processes in the Tevatron Complex. For example, the process of transferring antiprotons from our Accumulator ring to our Recycler ring has a similar SDA clock defined for it.

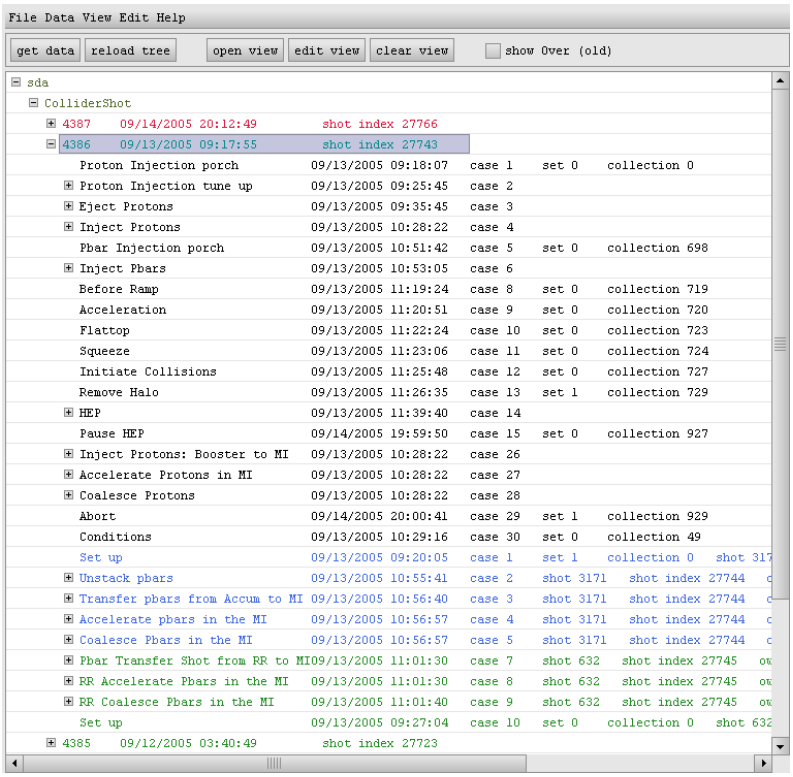


Figure 2, SDA Viewer, Shot detail

This definition of a clock is crucial to the simplicity and usability of the SDA system. Every expert can determine easily which point on this "clock" is important (for example, "there was an emittance blowup at the Coalesce Protons Case for store number 4386"). Looking up the information is also simple, since this is the way it is stored. Moreover, it is a simple matter to correlate these SDA entries with data that only have a raw time stamp (as in our data logger).

The Supertable

The Supertable is a systematic tabulation of intensities, efficiencies, transverse emittances and their growth rates, bunch lengths, lifetimes and other important data on the shot and on the store. The current version of the Supertable may be found at http://www-ad.fnal.gov/sda. The Supertable is available as HTML, as a spreadsheet or as an AIDA/JAS file.

The Supertable allows easy study of accelerator measurements from shot to shot over a long period of time. There is one line per shot. The first column (column 0, out of 224) is the shot number and is the index into the table. A snippet of the Supertable is shown in Figure 3.

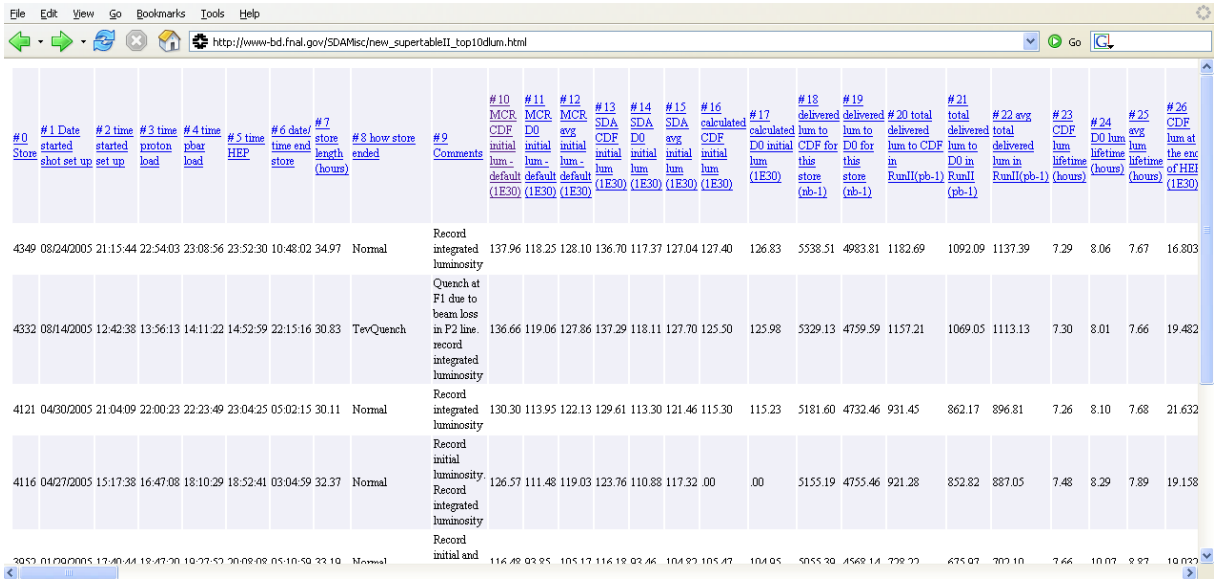
The data in the Supertable generally fall into two categories: Fundamentally important data on the shot or on the store (for example, the initial luminosity, the start time, the end time and the reason for ending the store), or data that require some extra calculations (for example, transmission efficiencies and emittances). These calculations are subject to change as our understanding of the facility changes. For example, different technicians are responsible for the beam current measurements in different machines, so the relative calibrations are systematically different. The Supertable calculations have been adjusted to take into account the differences in these calibrations so that the transmission efficiencies are accurate.

A particularly important example of an aspect of the SDA database and the Supertable that is subject to change is the emittance calculation in the Tevatron. Briefly, the emittance, ε, of a Tevatron beam is determined by measuring the width and the momentum spread of the beam,

$$\varepsilon = (K/\gamma)\,(w^2 - (\delta p/p)^2 D^2)\,/\,\beta$$

K is a constant, γ is the relativistic gamma factor, w is the measured beam width, $\delta p/p$ is the measured momentum spread of the beam, D is the dispersion value of the machine lattice at the measurement point and β is the beta function of the machine lattice at the measurement point. The lattice functions have been poorly determined until recently, owing to the inaccuracy of the beam-position monitor (BPM) system in the Tevatron. As the new BPM system has come online [5], improved measurements of the beta and dispersion functions in the Tevatron are being developed, and the calculation of the emittance is affected. Moreover, the values chosen for D and β are surprisingly controversial. As the accepted values for these quantities change, it has been necessary to recalculate the Supertable emittance values.

Management at Fermilab decided in 2003 that the numbers in the Supertable would henceforth be the official numbers that describe the shots. Corrections have been made to numbers were found to be in error, or that disagree with an otherwise accepted number. It has been our experience that people throughout the laboratory, and in our parent organization in Washington, DC, look at data from the Supertable to gauge the operation of the entire facility.



#0 Store	#1 Date started shot set up	#2 time started set up	#3 time proton load	#4 time pbar load	#5 time HEP	#6 date/ time end store	#7 store length (hours)	#8 how store ended	#9 Comments	#10 MCR CDF initial lum - default (1E30)	#11 MCR D0 initial lum - default (1E30)	#12 MCR avg initial lum - default (1E30)	#13 SDA CDF initial lum (1E30)	#14 SDA D0 initial lum (1E30)	#15 SDA avg initial lum (1E30)	#16 calculated CDF initial lum (1E30)	#17 calculated D0 initial lum (1E30)	#18 delivered lum to CDF for this store (nb-1)	#19 delivered lum to D0 for this store (nb-1)	#20 total delivered lum to CDF in RunII(pb-1)	#21 total delivered lum to D0 in RunII (pb-1)	#22 avg total delivered lum in RunII(pb-1)	#23 CDF lum lifetime (hours)	#24 D0 lum lifetime (hours)	#25 avg lum lifetime (hours)	#26 CDF lum at the end of HEI (1E30)
4349	08/24/2005	21:15:44	22:54:03	23:08:56	23:52:30	10:48:02	34.97	Normal	Record integrated luminosity	137.96	118.25	128.10	136.70	117.37	127.04	127.40	126.83	5538.51	4983.81	1182.69	1092.09	1137.39	7.29	8.06	7.67	16.803
4332	08/14/2005	12:42:38	13:56:13	14:11:22	14:52:59	22:15:16	30.83	TevQuench	Quench at F1 due to beam loss in P2 line. record integrated luminosity	136.66	119.06	127.86	137.29	118.11	127.70	125.50	125.98	5329.13	4759.59	1157.21	1069.05	1113.13	7.30	8.01	7.66	19.482
4121	04/30/2005	21:04:09	22:00:23	22:23:49	23:04:25	05:02:15	30.11	Normal	Record integrated luminosity	130.30	113.95	122.13	129.61	113.30	121.46	115.30	115.23	5181.60	4732.46	931.45	862.17	896.81	7.26	8.10	7.68	21.632
4116	04/27/2005	15:17:38	16:47:08	18:10:29	18:52:41	03:04:59	32.37	Normal	Record initial luminosity. Record integrated luminosity	126.57	111.48	119.03	123.76	110.88	117.32	.00	.00	5155.19	4755.46	921.28	852.82	887.05	7.48	8.29	7.89	19.158
3952	01/09/2005	17:40:44	18:47:20	19:27:52	20:08:08	05:10:59	33.19	Normal	Record initial and	116.48	93.85	105.17	116.18	93.46	104.82	105.47	104.85	5055.39	4568.14	728.22	675.97	702.10	7.66	10.07	8.87	19.032

Figure 3, Supertable Sample.

Other SDA Tools

In addition to the SDA viewer described above, there is a rich set of tools for working with raw data from the Accelerator Complex. We have an SDA Edit tool for editing the rules, an SDA Plot Viewer for browsing of blocks of time/value data in the Sequenced Data Acquisition database, an SDA Reports tool for implementing simple expressions involving the scalars in the data logger database and several others. All of them can be found in Fermilab index pages [6]. JAS (Java Analysis Studio) plugins for obtaining SDA and data logger data have also been developed.

One useful application of these tools is the Shot Scrapbook. The electronic logbook system at Fermilab has been extended so that one instance of the logbook, the Shot Scrapbook, is filled automatically [7]. Some of the items that are entered into this logbook are luminosities, emittance and intensity tables, graphs of relevant device readings during the shot setup, etc…

JAVA PACKAGES

Shot data analysis is based on Java packages called OSDA (Open SDA) and StorePhysics. OSDA retrieves data from the SDA databases via HTTP (outside of the Accelerator Division firewall) or through a fast binary protocol (inside the firewall, via Java RMI [8]). Requests for data are consolidated and partially cached to improve performance.

Physics calculations relevant to the Supertable are performed by classes in the Java package StorePhysics. This is the second package written to perform this task, written using the experiences learned from the first package. Of particular relevance are the classes that calculate the emittances and the container classes that represent the 36-bunch data that are common in SDA [9]. The classes in this Java packages are used both for offline calculations and for several online calculations.

A container class, BunchData, was added to StorePhysics to enhance Java arrays in several ways:

- The values in the container are verified before being used
- The class holds a trace-back String object that records the location in the Java code where the values in the container are changed, which is very useful during debugging.
- Iterators have been implemented, to insure that the first point is true.

Value verification in the container class is important because most calculations should verify every value every time. For example, any beam diagnostic can return the value “NaN” (for “Not a Number”) to represent the result of an illegal operation. Using NaN in a calculation will invalidate the calculation, so this must be avoided. Also, some data should only be in a specific range, for example, a beam width or an emittance must be positive.

The BunchData class is about 5% slower that the array-only calculations of the previous class package. But the previous version does not do systematic validity checks.

Another important improvement implemented in the StorePhysics package is the class LatticeFunctionFactory, which generates lattice functions for emittance calculations. In particular (as we have seen), this is important for the emittance calculations. The database from which LatticeFunctionFactory obtains the values is provided by a MySQL server, though an XML-based web communication. The values used for the lattice functions can be seen on a web page [10].

OPERATIONAL PERSPECTIVE ON SDA

The SDA system has worked well for Tevatron Run II. But it has required significant attention by experts, about 1.5 F.T.E., to keep the system running. An outline of the issues we have encountered, how we have dealt with them and the lessons learned from these experiences is presented here.

Issues

Beam diagnostics sometimes fail silently. Often, the only way to know that a diagnostic has failed is to look for bogus results in the Supertable or the SDA Viewer. We have implemented automatic monitoring of many aspects of the SDA database, but the false-positives are high and these logs have to be examined by experts. When the problem is in the diagnostic hardware, we have reverted to redundant measurements. For example, we can get the emittance in the Tevatron from two different flying wires. Although one flying wire (at E11) is preferred over the other one (at E17), we can use the E17 wire

```
DecimalFormat (
Mult( 12.345679012345679,
Add(
VectorMult(
sdaData(ColliderShot, _shot_, Case(ColliderShot, Inject Protons: Booster
to MI), -1, B:CHGB15, 0, 0),
sdaData(ColliderShot, _shot_, Case(ColliderShot, Inject Protons: Booster
to MI), -1, G:BNCH15, 0, 0)))))
```

Figure 4, an example of the output from a Functor; this is the calculation for column 53 of the Supertable: Booster Proton Intensity.

measurement if E11 fails. Furthermore, the emittance from the Synchrotron Light diagnostic, while not trusted by everyone yet, provides a redundancy check for the flying wire measurements.

The original Java code was hard to debug unless you have a detailed, experienced understanding of it. Part of the reason is that abstractions have been used extensively to write the Java code, making it difficult, sometimes, to trace where a number is actually calculated. The BunchData container class, with its trace-back storage, partially solves this problem.

It is crucial, as explained above, to ignore bad data when performing a calculation on beam bunches. For example, the average emittance of the beam is an important quantity that is recorded in the Supertable. This quantity is the average value of the 36 individual bunch emittances. If the validity of the data for each of the bunches is not verified, then the summary datum can be compromised.

This system requires several web-servers and daemons to be running at all times. There are four different servlet contexts (~20 servets and ~ 100 Java server pages (JSP)) to support and seven different daemons that recompute standard tables and graphics. All of these web-servers and daemons may misbehave due to the usual sort of software problems (overswapping on the server, incompatibility between servlet code and new version of code in DAE, ongoing upgrades of different parts of system, etc…). Constant vigilance is required.

User-friendly SDA data are stored in multiple formats in many files. Keeping consistency between the code that generates these files and the files that have already been generated is a problem. Regenerating or removing incorrect files produces a different set of problems. Our solution is to keep everything in a database and generate a file only when it is requested. But that is very slow for the user. It has been decided to generate the most commonly requested user files when possible, and to allow any file to be regenerated upon request.

As with any large software system, up-to-date documentation is hard to achieve. It is common for small changes to be implemented without updating the documentation. After several "small changes," the software is substantially different from the documentation. An automatic documentation mechanism has been implemented in some parts of the code using Functors [11]. Whenever the Functors code changes, it is possible to ask the Functor code to print itself out, which reveals the algorithm being used for the computation. A sample of a Functor printout is shown in Figure 4.

The Supertable was conceived as a small summary spreadsheet of only the most important quantities for a store. It started with around 20 columns. Now it has grown to 224 columns, and has become difficult for some people to use because of its size. Supertable Views partially address this problem.

Lessons Learned

The intent of the Supertable is to present the user with pre-digested data that best represents the performance of the Complex. Some data are difficult to measure with a single sample, so averaging is used to improve the precision of the recorded value. For example, the readout of the luminosity from one experiment has an RMS noise level of about 0.1 E30 [cm^{-2} sec^{-1}]. Since this actually is noise, averaging N readings will reduce the apparent jitter in the reading by a factor of the square root of N.

Redundancy has been crucial. A high-level repository like the SDA database or (even more so) the Supertable must have all the data all the time to be useful. Thus, we have encouraged the creation of redundant diagnostics hardware and we have developed mechanisms for generating SDA data in a number of different ways. Moreover, redundant data sources provide a cross-check for all the data sources. Some of the redundant diagnostics are:

- Emittances from the two flying wires, from the Synchrotron Light and from correlations to emittance measurements from the previous machine.
- Bunch length measurement, from the Fast Bunch Integrator (FBI) and from the Sampled Bunch Display (SBD).
- Intensities, from toroids, the FBI and the SBD. These intensities are calibrated automatically.

The SDA database is backed up by the conventional data loggers. With all of these redundancies, it is straight forward to regenerate the Supertable, which we have done many times.

Although the summary information provided in the Supertable is very useful, it is also useful sometimes to look at the details of these summary data. In particular, the average or the sum over all 36 bunches that is reported in the Supertable may hide interesting or important effects. One example is the luminosity lifetime. The average lifetime for the total luminosity is reported in the Supertable, but lifetimes for all 36 luminosity bunches are also available. There are ongoing studies in the Tevatron Department to understand in detail the factors that affect this lifetime, which can only be studied on a bunch-by-bunch basis.

Rapid prototyping has been a useful for creating new facets in this system. A researcher has the ability, using the Supertable in AIDA/JAS or in Excel, to create useful plots that s/he wants to be generated automatically. At this point, the researcher and the programmer can sit down together and rapidly generate the display or the data manipulation that is desired. This allows the programmer to have the problem completely specified and s/he can proceed to implement it concretely.

CONCLUSIONS

A Shot Data Analysis system has been implemented to provide a simple view into the complicated data that comes from the Tevatron Collider Complex. Two of the central aspects of this system are (1) the Supertable, which is a summary table of all the most important information about a shot and a store, and (2) the Java API that allows access to the SDA data and algorithms. Maintaining this system has required the effort of one and a half full-time-equivalent workers. Enhancements to the underlying Java class packages are intended to make this system viable to the end of Run II.

We would like to acknowledge the work of many other people at Fermilab who contributed to this work, most notably Krzysztof Genser, Paul Lebrun, Suzanne Panacek and Dennis Nicklaus, and to the Accelerator Controls Department at Fermilab.

REFERENCES

Fermilab is operated by Universities Research Association Inc. under Contract No. DE-AC02-76CH03000 with the US Dept of Energy.

[1] Java Analysis Studio is found at http://jas.freehep.org/.
[2] "SDA-Based Diagnostic and Analysis Tools for Collider Run II," T. Bolshakov, *et al.*, PAC 2005, Knoxville, TN.
[3] The DAE is documented at http://www-bd.fnal.gov/controls/public/JavaControls.htm.
[4] OSDA & StorePhysics AIP can be found at http://mccrory.fnal.gov/doc/
[5] "Tevatron Beam Position Monitor Upgrade," S. Wolbers, *et al.*, PAC 05, Knoxville, TN.
[6] Fermilab Java index page is at http://www-bd.fnal.gov/appix/.
[7] The shot scrapbook may be seen at:
http://www-bd.fnal.gov/cgi-mach/machlog.pl?nb=scrap03&load=no
[8] Java RMI: http://java.sun.com/products/jdk/rmi/
[9] The Tevatron, when we are delivering colliding beams to the experiments, has 36 proton and 36 antiproton bunches counter circulating in the beam aperture. Virtually all beam data collected from the Tevatron is collected in arrays of length 36 or 72.
[10] See http://tomato.fnal.gov/lattices.
[11] A functor is a function that can be manipulated as an object, or an object representing a single, generic function—http://jakarta.apache.org/commons/sandbox/functor