

Integrating and automating the software environment for the Beam and Radiation Monitoring for CMS

Olga Filyushkina
CERN, HAAGA-HELIA University Of Applied Sciences

Geneva, April 16, 2010



Acknowledgements

I am heartily thankful to my supervisor, Dr. Richard Hall-Wilton, whose encouragement, guidance, and support from the initial to the final level enabled me to develop an understanding of the subject. I would like to show my gratitude to Dr. David Stickland, Patrice Siegrist, Matthew Hollingsworth, Steffen Mueller, and the Beam and Radiation monitoring group for their help. I am also grateful to CERN Technical student program that made this thesis possible.

I also would like to show my gratitude to my thesis coordinator Jukka Juslin at HAAGA-HELIA University of Applied Science for his guidance and support in a number of ways. I also would like to thank Juhani Valimaki, Markku Somerkivi and HAAGA-HELIA University for the support during my work.

Lastly, I offer my regards and blessings to my family and friends who supported me in any respect during the completion of the project.

ABSTRACT

The real-time online visualization framework used by the Beam and Radiation Monitoring group at the Compact Muon Solenoid at Large Hadron Collider, CERN. The purpose of the visualization framework is to provide real-time diagnostic of beam conditions, which defines the set of the requirements to be met by the framework. Those requirements include data quality assurance, vital safety issues, low latency, data caching, etc. The real-time visualization framework is written in the Java programming language and based on JDataViewer—a plotting package developed at CERN.

At the current time the framework is run by the Beam and Radiation Monitoring, Pixel, Tracker groups, Run Field Manager and others. It contributed to real-time data analysis during 2009-2010 runs as a stable monitoring tool. The displays reflect the beam conditions in a real-time with the low latency level, thus it is the first place at the CMS detector where the beam collisions are observed.

CONTENTS

1. <i>Introduction</i>	7
1.1 The European Organization for Nuclear Research	7
1.2 Large Hadron Collider	8
1.3 Compact Muon Solenoid	8
1.4 Beam and Radiation Monitoring for the CMS	8
2. <i>Scope of the Assignment</i>	10
2.1 Requirements imposed upon the software architecture	10
2.2 Technical Tools	11
3. <i>Project Framework</i>	13
3.1 The Beam and Radiation Monitoring system	13
3.2 Arcitecture of the BRM system	15
3.2.1 Network	15
3.2.2 Computing	15
3.2.3 Protocols	17
4. <i>Data Transport</i>	21
4.1 Data transport	21
4.2 Initial resources	24
5. <i>Visualization Package</i>	26
5.1 Implementation of Visualization Package	26
5.1.1 Special features	31
5.1.2 Problems	39
6. <i>Conclusions</i>	41
6.1 Deliverables and their validity	41
6.1.1 Security Audit	48
6.1.2 Reliability Analysis	48
6.2 Usage	51

7. <i>Outlook and Recommendations</i>	55
7.1 Assessments of the working process	55
7.2 Reflection on the working process	56
7.3 Recommendation	56

LIST OF FIGURES

3.1	CMS Detector.	13
3.2	BRM Network.	16
3.3	Data Interchange protocol model.	19
4.1	Data Transport.	22
5.1	Visualization framework, class digram.	26
5.2	DipCacheVisualizer class.	28
5.3	DataViewerConfigurator class.	28
5.4	ConfigurableDataViewer class.	29
5.5	LogChartInteractor.	32
5.6	Example of LogChartInteractor.	32
5.7	HistogramDataSet.	33
5.8	Histogram types.	35
5.9	Histogram example.	36
5.10	Header Applet, class diagram.	36
5.11	NumberMonitoringApplet class.	37
5.12	ReadHeaderXml class.	37
5.13	Header class.	38
5.14	Header example.	39
6.1	Start-up latency.	42
6.2	Data transport measurement.	43
6.3	Clock synchronization assessment.	44
6.4	BCM2 clock phase.	45
6.5	BCM1 clock phase.	46
6.6	Interaction tools.	50
6.7	Luminosity Scan, BRM Summary monitor, 2010 run.	52
6.8	7 TeV Collisions at 24 Hours Summary Monitor, 2010 run.	53
6.9	LHC Activity at LHC Summary Monitor, 2009 run.	54

1. INTRODUCTION

1.1 The European Organization for Nuclear Research

The European Organization for Nuclear Research, known as CERN[1], is the world's largest particle physics laboratory. It was established in 1954 and is situated in the northwest suburbs of Geneva on the FrancoSwiss border. The organization has twenty European member states, and is currently the workplace of approximately 2,600 permanent employees and in total approximately 4500 employees, as well as 7,931 scientists and engineers as collaborators representing 580 universities and research facilities and 80 nationalities. CERN's main function is to provide the particle accelerators and other infrastructure needed for fundamental physics research. The output of the organization is nominally pure research and trained professional staff experienced in research activity; a large proportion of whom subsequently contribute significantly to a wide variety of economic activity typically within Europe. Whilst there is little direct financial benefit to CERN from spinoffs, CERN generates significant economic support and effectively R+D funding to the European high-technology industry and numerous small company startups through the technological developments required to build the large experiments. Lastly, there are numerous byproducts of the research which benefit society as a whole, the most obvious examples of which are the World Wide Web, GRID and cloud computing, and also large contribution in the medical physics arena. Numerous experiments have been constructed at CERN by international collaborations to make use of them. As an international facility, the CERN sites are officially under neither Swiss nor French jurisdiction. Member states' contributions to CERN for the year 2008 total CHF 1 billion which is approximately €664 million. CERN is also noted for being the birthplace of the World Wide Web. The main site at Meyrin has a large computer center containing very powerful data processing facilities primarily for experimental data analysis, and because of the need to make them available to researchers elsewhere, has historically been (and continues to be) a major wide area networking hub. Grid computing is developed at CERN, which is the combination of computer resources

from multiple administrative domains applied to a common task, usually to a scientific, technical or business problem that requires a great number of computer processing cycles or the need to process large amounts of data. Grid computing is distributed, large- scale cluster computing, as well as a form of network distributed parallel processing.

1.2 Large Hadron Collider

The Large Hadron Collider(LHC)[2][3] is the main project at the moment. It is the world's largest and highest-energy particle accelerator(the record achieved on 30 November 2009), intended to eventually collide opposing particle beams at an energy of 7 TeV per particle. LHC is being built in a circular tunnel 27 km in circumference. The tunnel is buried around 50 to 175 m. underground. It straddles the Swiss and French borders on the outskirts of Geneva.

1.3 Compact Muon Solenoid

The Compact Muon Solenoid(CMS)[4] experiment is one of two large general-purpose particle physics detectors built on the LHC. It is designed to see a wide range of particles and phenomena produced in high-energy collisions in the LHC. The CMS experiment is 21 m long, 15 m wide and 15 m high, and sits in a cavern that could contain all the residents of Geneva. Detectors consist of layers of different material, which stop and measure the different particles, and use this key data to build up a picture of events at the heart of the collision. CMS involves approximately 3,600 people from 183 scientific institutes, representing 38 countries that form the CMS collaboration who built and now operate the detector.

1.4 Beam and Radiation Monitoring for the CMS

The Beam and Radiation Monitoring group is a part of the CMS collaboration. The CMS Beam and Radiation Monitoring system is composed of 6 different subsystems that monitor the beam conditions and radiation field in and around CMS over time scales that range from bunch-by-bunch to long term monitoring. The remit of the BRM group is to providing monitoring of the beam-induced radiation field and also real-time fast diagnosis of beam conditions, initiating protection procedures in the advent of dangerous conditions for the CMS detector. The BRM working group consists of

approximately 50 people (7 people located at CERN, the rest is in Germany, USA and New Zealand).

2. SCOPE OF THE ASSIGNMENT

The scope of the assignment includes developing and supporting the web-based real-time visualization framework, which divides into the following tasks:

- Displaying information in real time with low latency is the main function of the framework and the final goal of the development process.
- Developing and implementing the display environment to support the visualization monitors. This stage of development includes interaction with a “DIP-Cache” [5] server that is capable of serializing the cache’s data into XML files and combining different monitors into one web page visible anywhere in the CMS private network.
- Data quality assurance is an important part of the framework intended to satisfy the data quality standards. The data is stored and executed in XML format, which ensures the consistent and coherent data processing.
- Safety issues are vital. As a real-time system, the monitoring framework should run 24/7/52. Availability of the monitoring directly affects the CMS and LHC availability for data taking.
- Web-based “DIP-Cache” Administrator service is a web interface for “DIP-Cache” server. It is an important part of the system, that is intended to increase the usability and security of the framework.
- Security issues and protocols have to be defined to protect data from unauthorized access and also to ensure safety of the system.

2.1 Requirements imposed upon the software architecture

- Displaying information in real time with low latency is the main function of the framework and the final goal of the development process.

-
- Developing and implementing display environment to support the visualization monitors.
 - Data quality assurance is an important part of the framework intended to satisfy the data quality standards.
 - Safety issues are vital. As a real-time system, the motoring framework should run 24/7/52.
 - Security issues.
 - Usability issues. The monitors should provide clear representation of data. They should be easy to use and configure.
 - The framework should be interactive, which means that the end-users should be able to set the monitor according to their need without making constant changes to configuration files.

2.2 *Technical Tools*

- scientific Linux operating system [6]. It is a Unix-like operating system produced by CERN and Fermi National Accelerator Laboratory (USA). It is free and open software based on Red Hat Enterprise Linux. The latest version 5.4 called Boron was released 04-11-2009. Scientific Linux includes software suited for the scientists and people working with scientific data.
- Eclipse environment, Gallileo.
- Programming languages: Java version 1.5 and 1.6.
- Markup Languages: XML, HTML and Cascading Style Sheets (CSS).
- Subversion as a version control tool.
- Apache Maven building tools.
- Standard communication protocol, such as HTTP.
- Data Interchange Protocol (DIP) [7], which is a system that allows relatively small amounts of real-time data to be exchanged between loosely coupled heterogeneous systems. The availability of C++ and Java client-server pairs allow the guaranteed flexibility for the protocol.

-
- DIP browser as a supporting tool for Data Interchange protocol. It is a simple java application developed at CERN and it allows to browse the DIP name space and see all the data subscriptions published by DIP. The application starts with java web start command and also it is very easy to use. It represents the structure of data subscription and give the information about subscription name, condition and the last timestamp.
 - JDataViewer framework [8] as a basic monitoring package.

3. PROJECT FRAMEWORK

3.1 The Beam and Radiation Monitoring system

CMS is a multipurpose detector designed around a 3.8T solenoid magnet[4]. Calorimeters, pixel tracker and muon chambers are set concentrically around the beam pipe and aim to measure the momenta of all particles produced by the proton-proton collisions. Many of these detectors can operate safely only when beam conditions are good. The BRM sub-detectors [9] [10] [11] are responsible for the safety of CMS and are pivotal in ensuring delicate detector systems are shut-down before beam losses can inflict serious damage by charge build-up the sensitive front-end electronics.

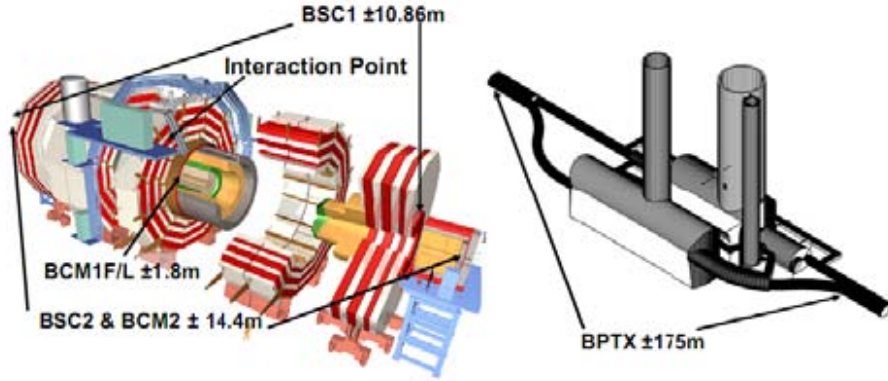


Fig. 3.1: Exploded view of the CMS detector showing the positions of the BRM sub-detectors. (Right) The CMS cavern and surrounding Long Straight Section in which the BPTX is located 175m from Interaction Point 5.

Figure 3.1 shows the positions of the BRM sub-detectors in CMS. The Beam Pickup Timing for the eXperiment (BPTX)[12] electrodes are mounted on the beam pipe in the long straight sections leading into the CMS cavern. Entering the cavern, the beams then pass by the Beam Conditions Monitor 2 (BCM2)[13] [14], Beam Scintillation Counter (BSC1 & BSC2)[15], Beam

Conditions Monitor 1 Leakage (BSC1L)[16] and Beam Conditions Monitor 1 Fast (BCM1F)[17][18][19][20].

Protection of any high energy physics experiment is of vital importance. The purpose built, sensitive electronic detectors throughout CMS are required to operate in a high radiation field and high magnet (up to 4 Tesla) environment. However should the instantaneous radiation field greatly increase, many of these systems could be destroyed due to excess currents induced by the beam. It was the task of the CMS BRM group to design and install the safety systems which will predict the loss of beam control and trigger a beam abort if pre-determined thresholds are met, signaling that conditions are deemed unsafe. The abort signal simultaneously causes the LHC kicker magnets to deflect the beams into the beam dump and shuts down all the at-risk detectors. Additionally, extensive monitoring is installed to allow diagnosis of adverse beam conditions.

The BRM system sub-detectors measure every possible aspect of the radiation entering the cavern. The tasks include monitoring of beam timing, intensity and position (BPTX), beam profile and losses (BCM1F, BCM1L, BSC, BCM2), beam halo and minimum bias event triggering (BSC) and ambient radiation dose to the surrounding region (RadMon [3]). The CMS experiment is one point away from the LHC beam dump at point 6 of the LHC. This puts CMS in a relatively dangerous position. The kicker magnets which remove the beam from the LHC and direct it in to the dump, must do so during the $3\mu\text{s}$ abort gap in the orbit train. Should an asynchronous beam-dump occur, the CMS detector could be showered by 10^{12} protons within $< 0.3\mu\text{s}$ potentially causing catastrophic damage to the sensitive inner tracker electronics. In such a case, the BRM systems will record data which will be used to give estimates of the radiation flux through many areas of the CMS cavern. This data will in turn give an indication of the detrimental effects (e.g. reduction of lifetime) on the surrounding instrumentation.

To reduce electronic noise and ground currents, strict CMS grounding rules had to be followed. In the cases of the BSC and BPTX, the signals are transferred to the readout electronics by coaxial cables and so special attention to the grounding scheme of these detectors was required. The BCM1F and BCM2 use optical fibers between their front-end system and the readout electronics, reducing the risks of ground loops.

3.2 Architecture of the BRM system

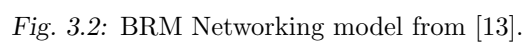
3.2.1 Network

The core of the BRM Network is a switched Ethernet connection. The network interrelation between different elements are shown in Figure 3.2. The main local network at CERN is called General Purpose Network or GPN. It provides the service for the office work stations and the WI-FI service for laptop on CERN sites. GPN devices may only access LHC and Experiment Network resources through dedicated gateways nodes. The .CMS network is the private network for most of the CMS systems. It provides security and independence to the CMS experiment and allows the use of more vulnerable network devices, such as oscilloscopes, temperature monitors, PLC's, and voltage controllers. The CMS experiment runs on the .CMS network only. It includes only few machines on CMS Experiment Network, few workstations and no laptops. The technical network is the private network of the LHC machine. It is also a part of the BRM network, because BCM1L and BCM2 systems use LHC readout electronics for the data transmission. That means that BRM comprises the tightly controlled gateway between CMS Experiment network and Technical network.

3.2.2 Computing

The BRM makes use of various different computing resources:

- The Control Room computers are the workstations that are physically located at the CMS control room. They are used by the shift crew at the control room and run scientific Linux 5. The majority of on-line visualization monitors are run locally there. Those computers are part of the CMS technical network.
- BRMBCMCTRL1 is a control node that is part of the LHC technical network. This computer has the role of a bridge to .CMS network from LHC network. The data is transported from BRMBCMCTRL1 station with the help of General Unilateral Transfer Server [21] to BRMBCMCTRL2 station located in .CMS technical network. This process is critical for the BRM system, because BCM1L and BCM2 detectors, that are part of the BRM System, use LHC readout electronics and the data that they produce has to be securely transported to the .CMS network.



- BRMBCMCTRL2 as it was described earlier is another core component in data transporting process. This node has a secure connection to BRMBCMCTRL1.
- cmsusr0-1 machines are located in the border between .CMS network and General Purpose network. They are connected to the BRMBCMCTRL2 with standard Ethernet cabling. Also the real-time visualization software is been developed and run on those machines. The operating system installed on cmsusr0-1 machines is Scientific Linux 4.4 (Beryllium).
- “Nodes 1 and 2” are used for the data processing. “DIP-Cache” web server runs on “node1” for collecting XML file. On “node2” the web server for data representation runs.

3.2.3 Protocols

On the BRM Networking figure 3.2 there are four type of connection specified. Those are Signal, Data, “PVSS” connection and DIP, that will be described in a separate section. “Signal” represents the analog data transmission between detector and front-end electronics. “Data” is any digital data transmission. PVSS [22] is used to connect to hardware (or software, such as voltage control and monitors) devices, acquire the data they produce and use it for their supervision. The BRM system uses PVSS for setting voltage to the detectors, monitoring the state of detector channels or setting alarms and warnings, and archiving the state of the detector channels.

Data Interchange Protocol

Data Interchange Protocol [7] is a system which allows relatively small amounts of real-time data to be exchanged between loosely coupled heterogeneous systems. These systems do not need low latency and the data is assumed to be mostly summarized data rather than low-level parameters from the individual systems.

The following points detail further its capabilities and intended use:

- DIP is a simple and robust publish/subscribe system which supports an on-change data exchange.
- It supports primitives, arrays of primitive and complex structures of primitives.

- The DIP API is supported on Windows and Linux and for C++ and Java.
- There is a negotiated contract between the consumer and the provider. Hence, the consumer is expected to know the name of the data item, its meaning and its data type. There will, however, be a formal repository describing the data items.
- Providers are expected to update the data at a rate which is suitable for the consumers. The provider is responsible for passing changes to DIP. DIP only transfers the data to the consumer when it receives it.
- DIP takes care of byte swapping, etc., transparently.
- Consumers and providers connect to DIP only via its API.
- It is not possible to have more than one provider/consumer per domain.
- There can only be one publisher per item.
- The DIP API allows the client on reconnect to be able decide either to get automatically the current value for all the data items he subscribes to or not to get them until they are updated.
- The DIP data format includes a timestamp and quality flag.
- A naming convention is defined and can be seen below.
- The following security measures are provided:
 - Only publishers from within the CERN domain are allowed.
 - Only one publisher per item should be allowed.

The purpose of the components identified in the figure are as follows:

- Data source represents the source of the data that is to be sent via DIP. It may be internal or external to the DIP server. The DIP server is responsible for accessing the data source and writing it out to DIP through the Publication.
- The publication, is a named object that represents an atomic piece of data that is published in DIP. The writer of the server must write the code that obtains the data from the Data Source and writes it into the publication object. A client subscribes to the publication by providing DIP with the publications name.

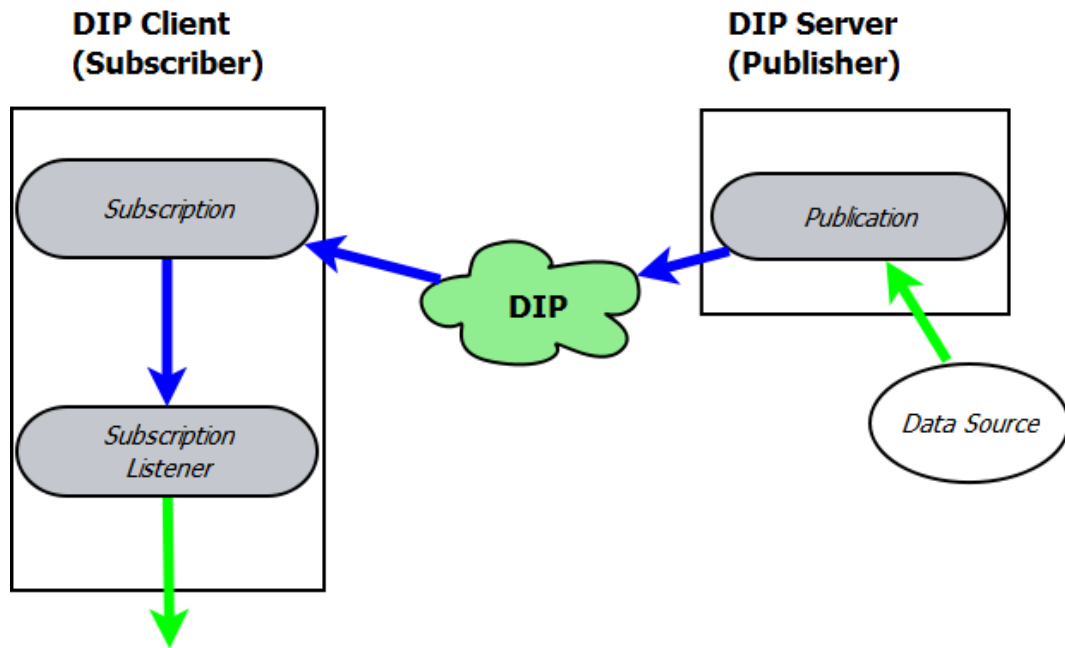


Fig. 3.3: In the above diagram the interaction between DIP servers and clients described. The arrows show the flow of information between the producer and subscriber. The green arrows indicate some action is required by the writer of the Publisher or Subscriber in order for the information to flow. The white arrows are used where the flow of information between components is handled by DIP.

- DIP provides the mechanism by which data is passed from the publisher to the subscriber.
- The subscription, is an object given to the client by DIP when that client subscribes to a publication. With this object a client may request the most recently published value of the publication or unsubscribe from a publication when that client had previously subscribed to.
- SubscriptionListener acts as a wrapper, containing several callbacks. The most important of which handles data from those publications subscribed to when it arrives.

4. DATA TRANSPORT

4.1 *Data transport*

The initial target of the visualization framework is real-time data monitoring. To achieve this goal, it is important to understand the data transport process, which significantly affects the the monitoring framework’s performance.

The Data flow Figure 4.1 describes the data transmission path for the BCM1L [18] and BCM2 [13] detectors. The path that data takes to the on-line monitors is marked with the red arrows.

The analog signal from the detectors goes to the VME-crates in the LHC network [2]. There data gets converted to a digital signal and goes to CERN Middleware (CMW) and also is provided to CERN Control Center (CCC) and to BRMBCMCTRL1, where it get published to DIP within the LHC technical network. At the next step the data is transported to BRMBCMCTRL2 in the .CMS network. This process is done by the GUTS package described later. The same package sends data to the rest of the .CMS network. At BRMBCMCTRL1 data gets published to DIP again, within .CMS network. The “DIP-Cache” server is built on top of DIP, so it caches the data from there. At the final stage data is retrieved from “DIP-Cache” server and displayed with the visualization framework.

The Controls Middleware

The controls Middleware (CMW) [23] is a complex infrastructure that provides a common software communication infrastructure for the CERN accelerator controls. In particular it supports the Accelerator Device Model and device I/O services, the publish/subscribe paradigm synchronized with Accelerator Timing, and interoperability with industrial control systems.

At the heart of CMW is the Remote Device Access (RDA) system, which defines the client and the server API and provides the communication on top of Common Object Request Broker Architecture(COBRA), that enables software components written in multiple computer languages and running on

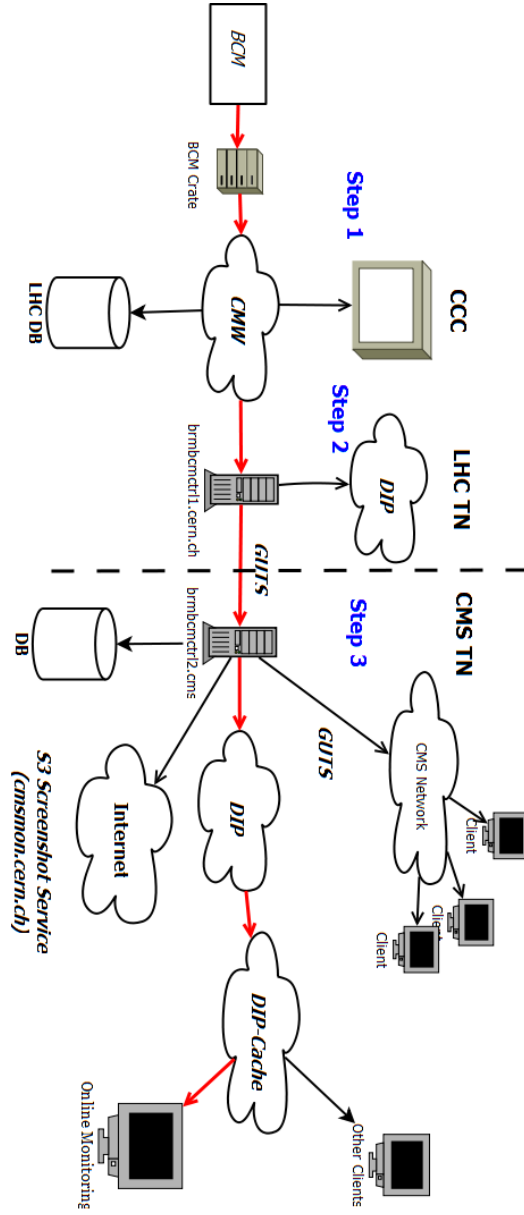


Fig. 4.1: Data flow digram.

multiple computers. The Java control programs constitute, the main category of CMW clients.

The other component integrated with CMW and RDA is called FESA [24]. The real-time front-end software architecture FESA is a framework used to fully integrate equipment such as power supplies, PLCs or beam diagnostic devices at the front-end level into the CS. It provides JAVA based graphical user interfaces to design, deploy, instantiate and test the equipment classes. FESA provides the connection to CMW and RDA and also it provides required services for data acquisition, programmable timing, amplification gain settings etc

Generalized Unilateral Transfer Server

Generalized Unilateral Transfer Server (GUTS) [21] is a Java framework that is designed to solve a specific problem of transporting the data through network. It implements a simple user API that allows users to send any Plain Old Java Object over the network to an arbitrary number of connected clients. The main function of GUTS is to be “unilateral” framework which means that data transfer only goes one way—from the server to the client. Data (other than a “keep alive” 0) is never allowed to go from the client to the server, this increases the security of the data transmission. This transfer service is used when CMW can not be used because of security reasons, for instance between LHC technical network and .CMS technical network.

The DIP-Cache server

The DIP-Cache server [5] is a a real-time data caching server capable of serializing the cache’s data on demand into XML. DIP-Cache is the key component in the on-line monitoring framework that plays the role of a transfer point between DIP protocol and the visualization framework. It is able to subscribe to the DIP subscriptions and cache the incoming data into the memory for a short period of time in binary format. Since the server is designed as a temporary caching tool and it keeps only a relatively small amount of data, it does not cause any problem to keep memory.

The DIP-Cache server starts up by reading XML configuration file that defines the set of default subscriptions and their maximum cache depth. It also can be configured using HTTP queries. For example, HTTP query that subscribes to a DIP publication looks like:

```
http://{hostname}:{port}/brm-wbm/dipProvider/subscribe
?dipName={dipName}&maxCacheDepth={cache depth}
```

At startup the DIP-Cache Server initially subscribes to 11 predefined subscriptions, which can increase depending on the required number of displays. Default cache depth size is 100 data points, but it is possible to change it at any time. The summary subscriptions of BRM detectors have a bigger size (from 3600 to 43200 data points), which makes them the main memory consumers. When new data arrives, it is added into a queue, so it is guaranteed that only the newest data is kept in the memory. Also any incoming data is identified by the subscription name, which provides an easy method to transport it to the client program later.

The client, which is the on-line visualization framework, is capable to subscribe to the DIP publications using the name of the publication. When the visualization framework makes first call to the DIP-Cache, it starts receiving the data that is already cached in a small chunks starting from the oldest data. After it catches up with the cache, the client gets the newest piece of data only.

The DIP-Cache Administrator page is currently under development and will be released soon. The page provide the users with the easy web interface to interact with the DIP-Cache server. It allows to monitor the condition of the server, add or remove subscriptions and modify the cached depth.

4.2 Initial resources

The on-line real-time visualization package is based on JDataviewer package [8] developed at CERN embedded inside of a Java Applet.

JDataviewer (JDVE) is a plotting package that creates a simple representation of data. The main reason why JDVE package was chosen from other technologies, is that it is developed at CERN and also it has the implementation of all necessary components for data monitoring, which includes data model (DataSource and DataSet objects), DataViewer class (the simple display object and its basis part), Chart class (simple chart object), different renderers (Bar, Scatters, etc), chart interactors (Zoom Interactor) and scalers. The package has a nice user interface and the ability to develop its functionality, but on the other hand it is not developed as a web package, which requires to use some additional tools to integrate the displays with a browser.

Since JDVE is a java package, it was rather easy to integrate it with Java Applet. Java Applet is a special type of a program that can be embedded to HTML page using `<applet></applet>` tag. It is a part of Swing package and it provides interactive features to web applications, like mouse input, button and check boxes, which is important part of the visualization

framework. Also it allows to run different applets within one page, which is important for the monitoring framework.

One of the main requirements for the framework is high usability level, which also means that it should provide easy and reliable way to create and configure a web monitor. This goal is reached by using XML configuration files, because they have very logical and clear structure, which describe the display, and also it can be easily parsed in Java.

A simple HTML page is used to assemble the display from different packages. The visualization framework has two different applet: DipCacheVisualizer and HeaderApplet. Also it is useful for setting title to the display and other simple operations.

5. VISUALIZATION PACKAGE

5.1 Implementation of Visualization Package

The architecture of the visualization framework is built on the principle of maximum simplicity and integration. Re-using the existing functionality makes the system light-weighted and stable, which is one of the most important requirements for the real-time monitoring.

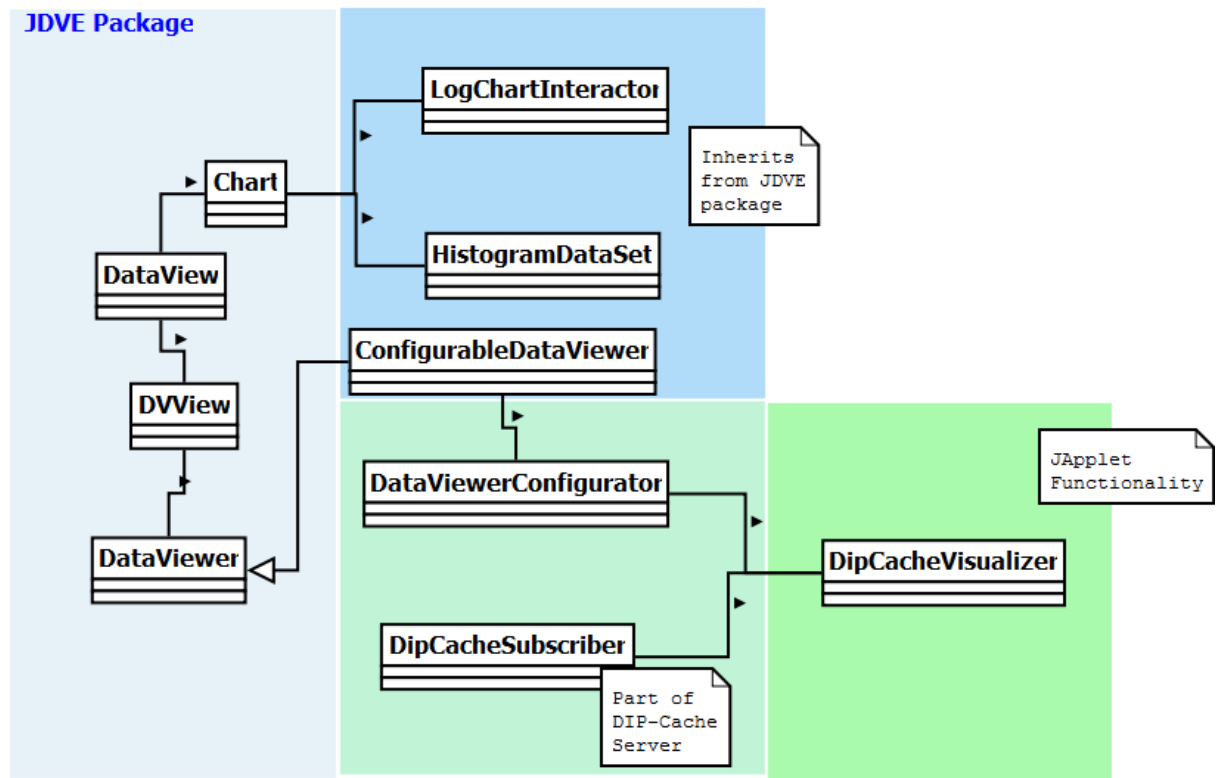


Fig. 5.1: The Class diagram that reflects the dependencies inside the framework.

The framework is developed around four main classes, but it also includes a number of auxiliary ones.

DipCacheVisualizer is an entry point class. It inherits from Java Applet class [25] and is responsible for integration of the monitors based on JDVE package to Java applet. The class contains the private class named DataGenerator, which implements DipSubscriptionListener interface, that is defined inside of the dip package. The internal class handles new data coming from DIP-Cache and puts it into a proper data-set. To retrieve the data, DataGenerator class has a method called extractData(DipData data, String field, int index), which extracts DipData into a suitable format.

In order for DataGenerator class to perform its task correctly, it is critical to have a robust and easy way to match a DataSet object to a DipSubscription. To meet this goal DipCacheVisualizer class has a private Hash map that matches the keys represented in String format to object arrays that contain a DataSet object and other relevant information.

Another important step that is crucial for the visualization framework is embedding a monitor inside of a Java Applet object. Inside of the DipCacheVisualizer it is done by the createChart() method, which makes a call to DataViewerConfigurator.configureDataViewer (String XMLFile) static method. DipCacheVisualizer gets the path to XML configuration file as a parameter from the HTML page and parses it to DataViewerConfigurator class that is capable of reading XML. The path to XML configuration file should be valid and reachable in order the framework to complete the task successfully. If the configureDataViewer() method fails to return an object the program can not continue executing.

DataViewerConfigurator.configureDataViewer(String XMLFile) method returns ConfigurableDataViewer object that is assigned to the internal field inside of DipCacheVisualizer. At this point the Hash map inside of ConfigurableDataView gets retrieved and reassigned to the local Hash map. At the same time for each data set new DipCacheSubscriber object is created that makes a call to DIP-Cache and gets the latest data, and then passes it to DataGenerator class.

Rendering XML is done within DataViewerConfigurator class. Since XML files can get rather big and complicated, but at the same time all configuration files have the same structure, Simple API for XML(SAX) technology [26] is used. The DataViewerConfigurator has a static method configureDataView(String xmlUrl), which plays the role of entry point to the class. This method performs two important operations: first it loads and parses XML document, and also creates an instance of ConfigurableDataViewer class that gets shaped while XML execution and then returned

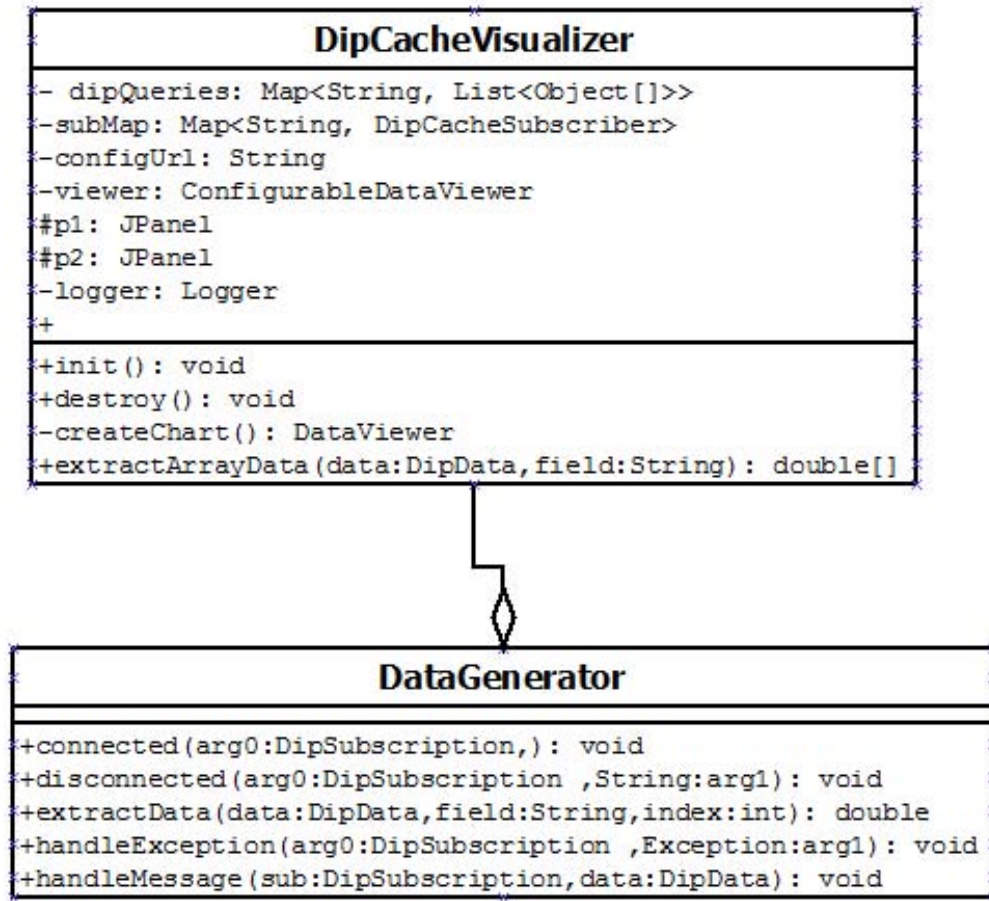


Fig. 5.2: DipCacheVisualizer.

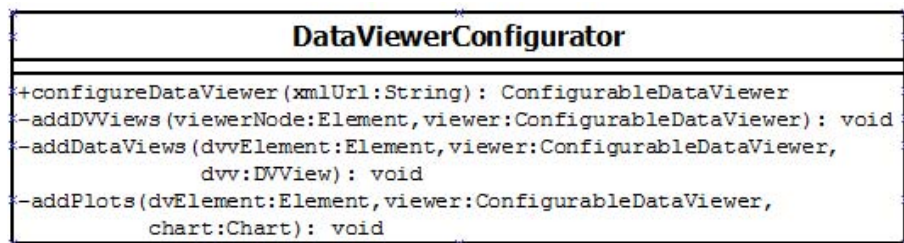


Fig. 5.3: DataViewConfigurator class.

to the DipCacheVisualizer class.

ConfigurableDataViewer is derived from DataView class that is the part of JDVE package. It contains a private map that lists datasets to the object array, which is very important process for data transport inside of the visualization framework. This map is used as a key inside of the DipCacheVisualizer class to match Dip Subscriptions to Dataset objects.

Also ConfigurableDataViewer is capable of making difference between a single field, array data or a certain field inside array. It is important to make to differentiate between data types in order to use suitable data extraction method and avoid data losses.

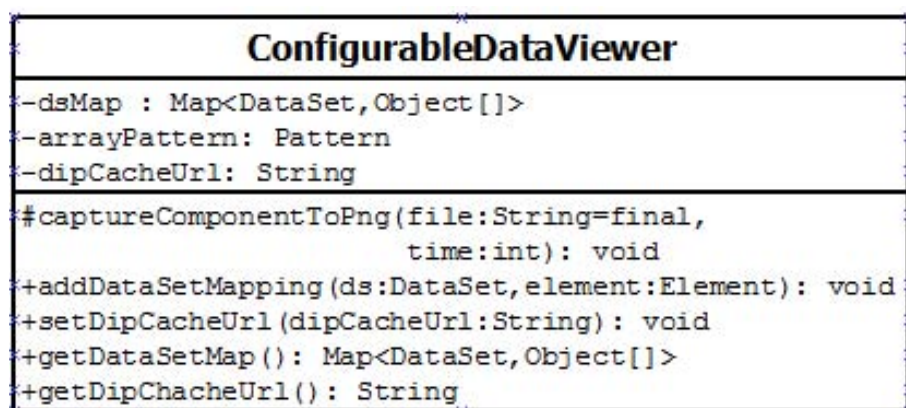


Fig. 5.4: ConfigurableDataViewer class.

The XML file represents the structure of the data to be displayed and the display itself at the same time, so it must be consistent and follow the pre-defined hierarchy.

Main tag is <Viewer> ... </Viewer> represents the container frame, which contains all charts and the toolbar. It might have any number of the second level element, but there may be only one Viewer tag per XML file. Second level tag is called <DVView> .. </DVView>. It is responsible for one tab in the display. It is possible to have multiple tags within one display instance, but it is not recommended due to the significant increase of data to be transmitted, but it is only possible to observe one tab per display at a time. So having several tabs is resource consuming and might slow down the applet. The third level element is <DataView> ... </DataView>, which represents a single chart within the display. Several number of DataView tags form a DVView element. The final level element is called Plot. A

single `<Plot> ... </Plot>` is responsible for one plot on a chart and there no limitations for number of plots within one chart.

Each tag contains a certain number of attributes, that describe the display and date to be monitored.

Viewer's attributes are:

- “Name” defines the name of the display.
- “dipCacheUrl” give URL to dipCache server, which contains cached dip data.
- “showExplorer” toggles data explorer on or off.
- “capture ” sets capturing function if true.
- “path” is auxiliary attribute used when the capture is set true. It specifies the directory where png files produced by capture function are saved.
- “Time” is another auxiliary attribute of capture function. It set the time period when one png file is made. By default it is set to 100000 milliseconds.

DVView has only one attribute called “name”, which defines the name of a tab in the display.

MapView element contains:

- “Name” is the name of a chart within display.
- “axisType” defines the axis type. Default values can be set as a number or timestamp. If it is left blank, then the value of axisType will be set to number automatically.
- “multiAxis” is boolean value which is set to be true when the multiple Y axis are needed.

The plot tag defines the single plot within the display and it has complicated attribute structure.

- “Name” is the name of the plot to be displayed.
- “dipName” defines a dip subscription that contains the data to be displayed.

- “xField” is the name of dip field to be plotted on X axis. When xField is not used, dip time stamp is plotted on X axis.
- “yField” is the name of dip field to be plotted as Y values. It is possible to specify a dip field that contains a scaler, array or an array element.
- “type” is an attribute that defines the type of the plot to be displayed. It has 5 default values that it can be set to, which are described in detail in a table below.

HTML configuration file is a basic HTML file that contains `<applet></applet>` tag. Its purpose is simply passes the path to XML file to the DipCacheVisualizer applet, and then embeds it into a web page. Also it is useful for combining a few applets in one web page and other simple HTML operations.

The other component that is used within the visualization framework is CSS definitions. The simplest way to use it is just embed CSS definition into XML file. It is not the cleanest method, but it insures that CSS doesn't get overwritten somewhere on the lower levels of the framework. The CSS styling supports at the moment 6 different elements (selectors): chart, chartArea, scale, grid, axis and legend. The scale, grid and axis elements may depend on the axis (X or Y). To specify style for selected a conditional selector with an argument axis set to X or Y is used. The properties defined for a specific axis override properties defined for both axis i.e. properties defined by selector `scale[axis='X']` override these defined by scale selector. The “dataset” selector requires a condition attribute: name of the data set which is the same with yField name in XML file.

5.1.1 Special features

During the development some additional features were added to the visualization framework.

- “LogChartInteractor” was originally planned as a class derived from ChartInteractor (JDVE), which means that it is non-graphical component of the framework that is used to interact with chart object. LogChartInteractor initially is responsible for switching between logarithmic and linear scale types. But later it got some extra functionality: adding different annotations and reset method to histograms and also implementing data picker.

Since “LogChartInteractor” inherits from ChartInteractor, it has to implement the parent methods: `chartConnected()`, `chartDisconnected()` (Chart

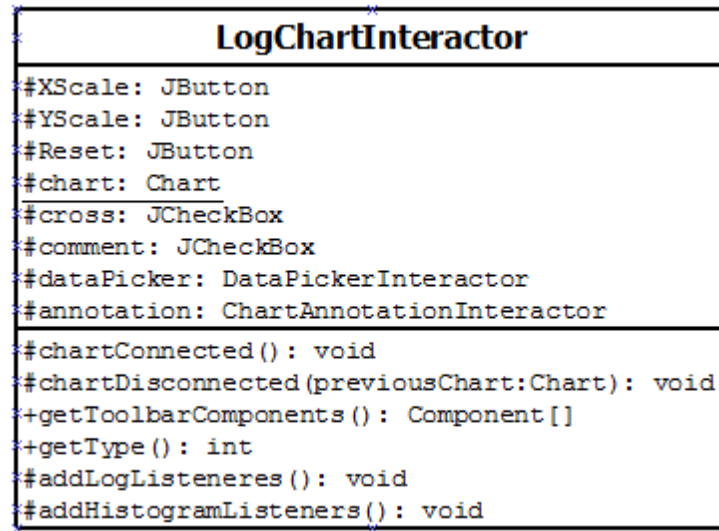


Fig. 5.5: LogChartIntector class.

previousChart) and getToolBarComponents(). chartConnected() is the method that initiates the interactor by assigning the chart object to the interactor. It also sets the buttons to the chart toolbar that are responsible for interactor's functions.

The LogChartIntector's functions depend on the types of the dataset objects as well as on toolbar assigned to the chart, thus the integration of the interactor should happen after the datasets are created and the chart object is configured. The integration is done by simply adding new instance of the LogChartIntector to the chart.

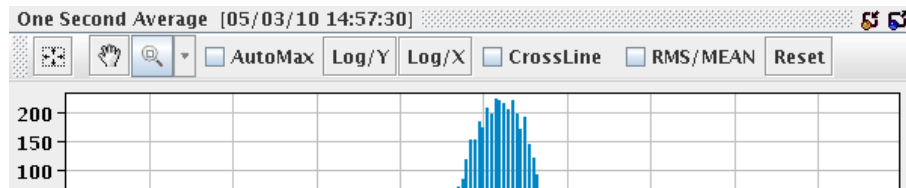


Fig. 5.6: The functionality of LogChartIntector directly in a chart's tool bar.

- "Histogram dataset" is one of the most important features in the vi-

sualization framework. HistogramDataSet is a child class of DefaultDataSet (JDVE). The main function of it is to arrange incoming data in a way that it could be rendered as a histogram.

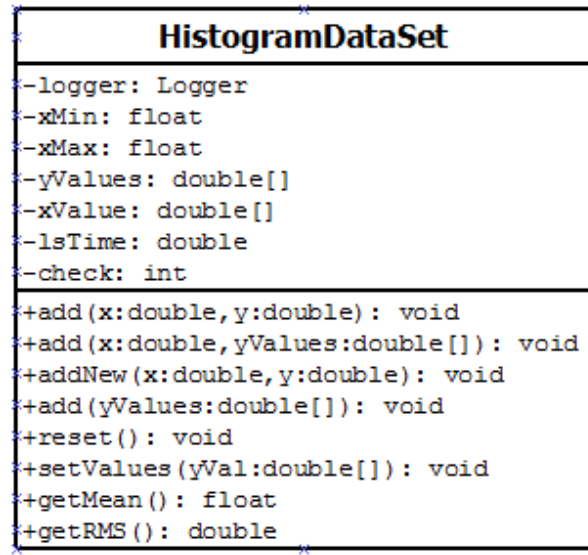


Fig. 5.7: HistogramDataSet class.

The first constructor is used for creating a dataset that sets the bins for when the first data arrives. It used for array data type only. The number of bins in the histogram then will be equal to the number elements in the array. There is a check field inside the constructor to make sure that the bins will be calculated when data arrives.

The second constructor is used when minimum and maximum value of bins are predefined by the user. After some simple calculations inside of the constructor, the xValues and yValues arrays are created and set as a starting point for the histogram.

The HistogramDataSet class also has four different methods to set incoming data that allows to have two different types of histogram. Different types of histograms are described in details in Table 5.8. “First type” sums incoming values with the data that is already in the data set. There is reset() method for this kind of histogram to set the bins to zero. The other type is called “instant” histogram, that contains the latest value only. When new data is coming, it overwrites

the values that were in the dataset before.

The type of the histogram is specified in the XML configuration file, so then dataset objects are created inside of the `DataViewConfigurator` class according to the configuration file.

- “Header Applet”: The `DipCacheVisualizer` applet is targeted to contain the graphical displays. There is a requirement to create an applet that could also display simple status messages such as text or numbers. It is implemented as a simple header applet displayed above the graphical monitors on a web page. To meet this goal the framework is using the `Header` or `NumberMonitoring Applet` class.

To simplify the monitoring of the status messages, the `NumberMonitoring` applet has the functionality that applies certain color schemes to the applet’s background or foreground. For example, the normal background color for the header is white, but if status message contains the word “ALARM”, the background of the header switches to red.

The design and implementation of the `Header Applet` is almost identical to the `DipCacheVisualizer Applet`. The only basic difference is that the “`ReadHeaderXML`” class, that parses XML configuration files, creates an object called “header” and then parses it to the header applet. Java Spring class was used to create a header object from an XML file.

The XML files has three basic types of tags: `header`, `subscription`, `dipField`. `<header> ... </header>` tag is responsible for the header object. It describes the name of the header and the URL to the DIP-Cache server.

`<subscription> ... </subscription>` specifies the DIP subscription. There could be one than one subscription in a single header. The subscription attributes are an url (provides full path to the DIP subscription), a state (a flag that shows whether the subscription has a field that reflect the state of it) and the `titleSize` (allows to arrange the header object in the best possible way).

`<dipField> ... </dipField>` tag defines the DIP field to be displayed as a field attribute, and also the title that should be given to the DIP field as a name attribute.

The HTML configuration file is defined the same way as for the `DipCacheVisualizer`. The only difference is that the path to the spring

Syntax	Purpose	Functionality	Usage	Example	Dataset Naming
history:Number of data points to be displayed in a chart	gets the number of data points defined from dipChace server and plots it on a time scale	Time scaler with X axis set to timestamp	It can be used for any single value	history:1000	The name of dataset is the name of the subscription
histogram:Xmin:xMax: x: Number of Bins	creates histogram within specified range	Histogram that will add new data points to existing data set	It can be use for arrays and a single value	histogram:-3:4:8	The name of dataset: histogram_<the name of the subscription>
histogram:-1	creates histogram with bins from 0 to N	Histogram that adds new data points to existing data set	It can be used for array values ONLY	histogram:-1	The name of dataset: histogram_<the name of the subscription>
instant:xMin:xMax:n umber of bins	creates histogram within specified range	Histogram that sets new data to dataset per unit of time	It can be use for arrays and a single value	instant:0:10 0:10	The name of dataset is the name of the subscription
instant:-1	creates histogram with bins from 0 to N	Histogram that sets new data to dataset per unit of time	It can be used for array values ONLY	instant:-1	The name of dataset is the name of the subscription

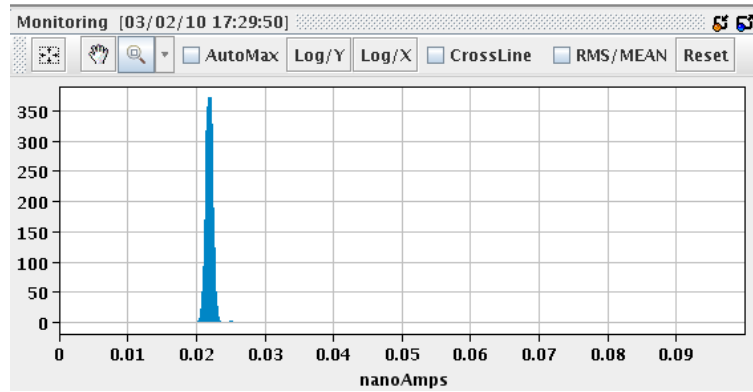


Fig. 5.9: An example of "summing" histogram.

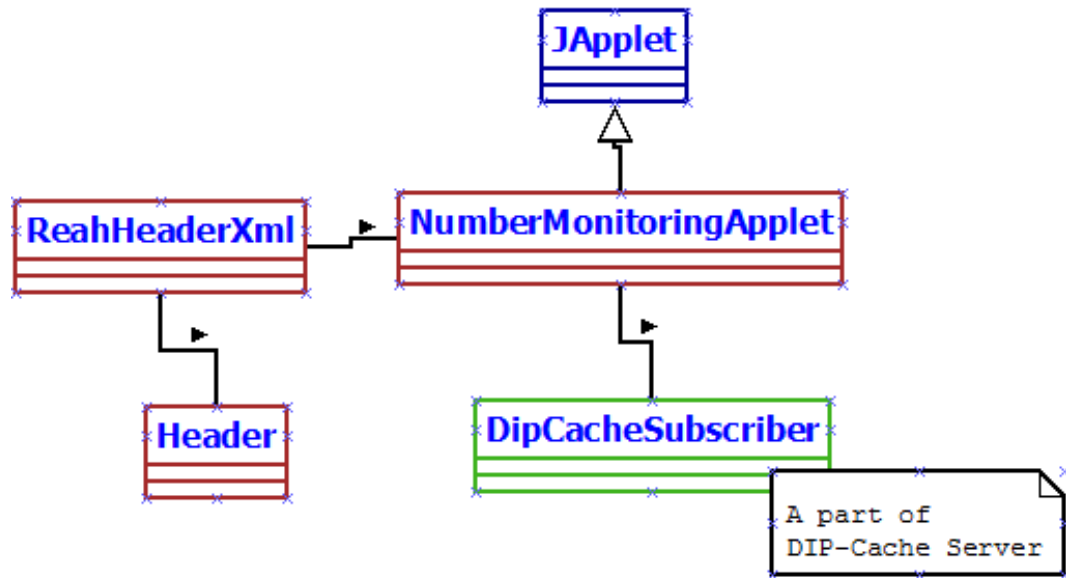


Fig. 5.10: Class Diagram that represents the relations of Header package.

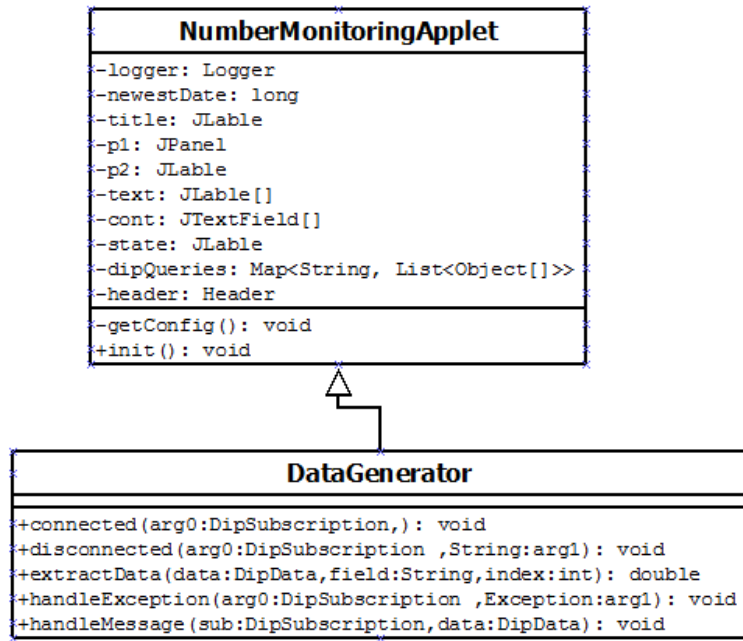


Fig. 5.11: NumberMonitoringApplet Class.

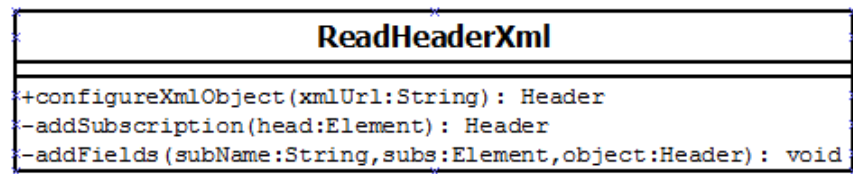
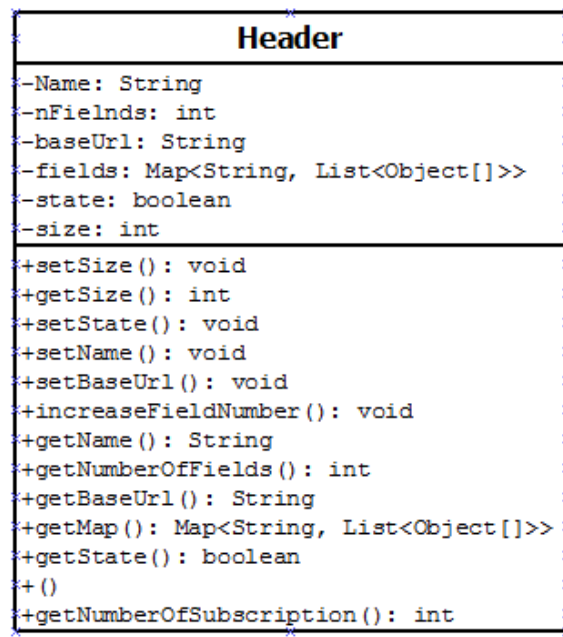


Fig. 5.12: ReadHeaderXml Class.

*Fig. 5.13:* Header Class.

framework should be added to the archive attribute.

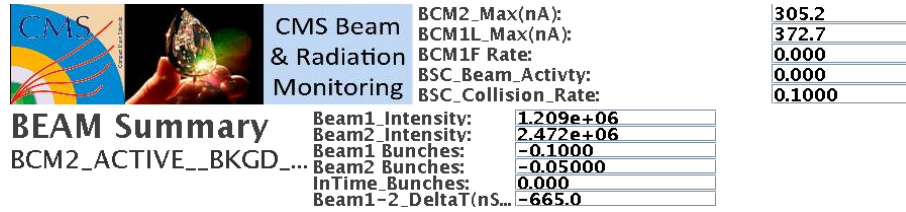


Fig. 5.14: An example of the Header Applet and BRM Logo.

- The capturing method is the method inside of visualization framework, that allows creating PNG images of the applets. The basic principle of the method is taking a capture of the certain applet and writing an image to a directory where the files can be accessed outside of .CMS Network.

5.1.2 Problems

During the implementation a number of problems occurred. All of the them can be divided in following categories:

- Java related problems are mostly referred to java version conflict between a java used inside of .CMS network and the java that the JDVE package is written on. At the development starting time java version 1.5 was installed inside of the .CMS network, but the JDVE package requires to use java 1.6. So the JDVE package was recompiled to java 1.5, which limited the possibility to get the support from the JDVE developers. This problem caused a losses of time and performance during the working process. At moment the java version was updated inside .CMS network to 1.6, so it is planed to switch the framework back to the newer java version as soon as possible.

The other problem that significantly affected the development process comes out of the JApplet class and its specifications. JApplet is a small program that should not take much of heap space, but since the visualization monitors could process rather large amounts of data, the heap size has to be manually increased. Also java security setting does not allow JApplet to read data from the local computers. Thus it is necessary to manually grant all necessary permissions to the end users, which is not efficient on the large development scale.

- JDVE package related problems come from the fact that the visualization framework does not use the original JDVE package, but the adapted version. All of them slow down the working process and apply some limitations to the development. For instance, the framework does not set the CSS definitions to the chart renderers, when using external CSS files. Thus all CSS configurations are added inside of XML config file, that makes it larger and more complicated to read.

The other JDVE related problem is that the chart object does not add more than one external interactor object. The interactors that are pre-defined in the JDVE package do not cause any problems, but all of the visualization framework functionality, that requires some communication with the chart, is forced to be assigned with single interactor object.

- Also various browser related problems affect the stability of the visualization framework. The browser used to run the monitors is Mozilla Firefox/3.0.7 for Linux. Considering the fact that the development nodes are running Scientific Linux 4, there is no opportunity to install newer Firefox version. The most common problems caused by the browser are that it does not destroy the applet object correctly and loading a new applet instance configured with a newer version of XML file. Also the crash rate of applets running by the Firefox is much higher, that the crash rate of the standalone application.

6. CONCLUSIONS

6.1 Deliverables and their validity

The final deliverable can be defined as a successfully implemented real-time visualization system that meets the requirements of Beam and Radiation Monitoring group. Thereby, the quality of the end development product can be evaluated by the detailed analysis of the system characteristics against the list of the requirements presented in Chapter 2.

- Displaying information in real time with low latency is the main function of the framework.

The desired refresh rate for the visualization framework should be close to the refresh rate of DIP subscription that the display is getting data from. During the 2009 run, the refresh rate of the visualization framework on average was equal to 1 Hz, which could be seen from the latency measurement.

For the precise estimation of the framework latency it is critical to measure the data transport latency. Data transport is one of the key processes that can effect the performance of the visualization framework. Its speed depends on the many different factors such as the amount of data, the protocols that are used, the data conversion speed on the different stages of data transport, network occupancy, etc.

In the figure 4.1 data transport process is shown. The analog data arrives to the front end readout electronics, where it is converted to the digital signal and assigned the time stamp in nanoseconds. Thus data transit time is assumed to be a difference between the readout electronics time stamp and the time when the data is added to the data set inside of the visualization framework.

The measurement is based on the customized version of DipCacheVisualizer class, that is capable to retrieve the front end readout electronics time stamp from the DIP subscription cache at DIP-Cache server, and write the difference of the time stamp and the system time to the data

file. Analyzing more than five hundred entries inside the data file, it is possible to notice that the first 100 or 250 entries produces lay between 10 nanosecond and 100 seconds with average value of 60, which is reflected in the figure 6.1. This effect comes from the fact that first entries are already cached in the DIP-Cache at the moment when the programs starts, so it takes some time to process the old values and catch up with the data flow. On the other hand after 250 data points being processed, the average latency goes down to one second.

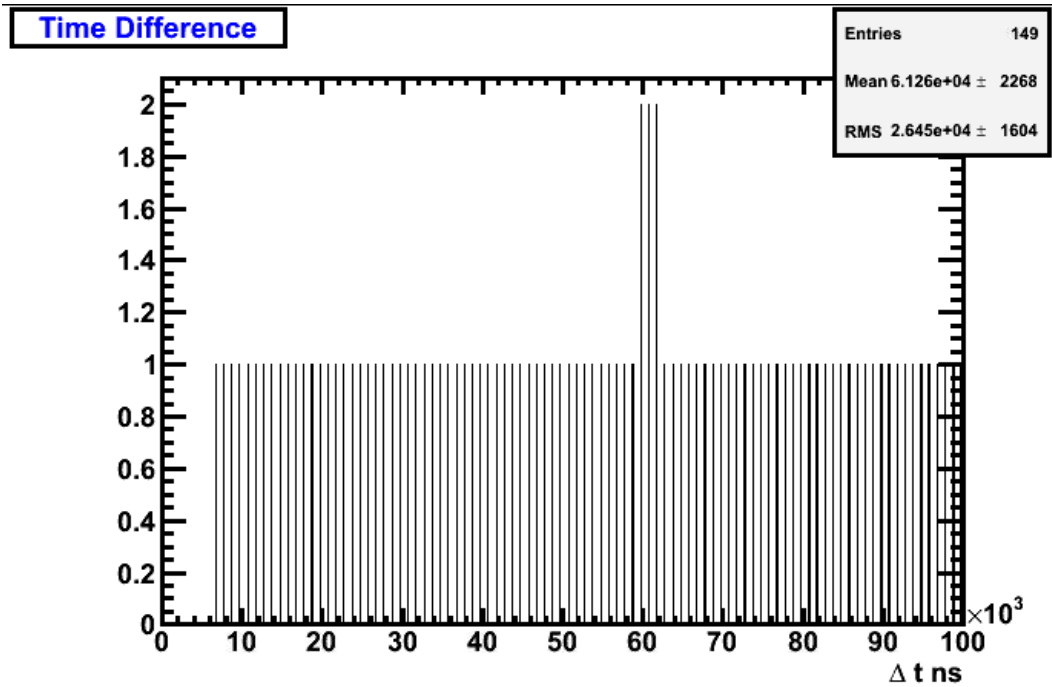


Fig. 6.1: Start-up latency (first 150 data points.)

Figure 6.2 reflects that the average working latency of the monitoring framework is around one second with the overflow of the measurement approximately 0.5 percent, which satisfies the requirements defined for the real-time monitoring framework.

The quality of the measurement can be affected by the clock synchronization, so in order to estimate the synchronization uncertainty, it is significant to make a clock phase assessment. Figure 6.3 demon-

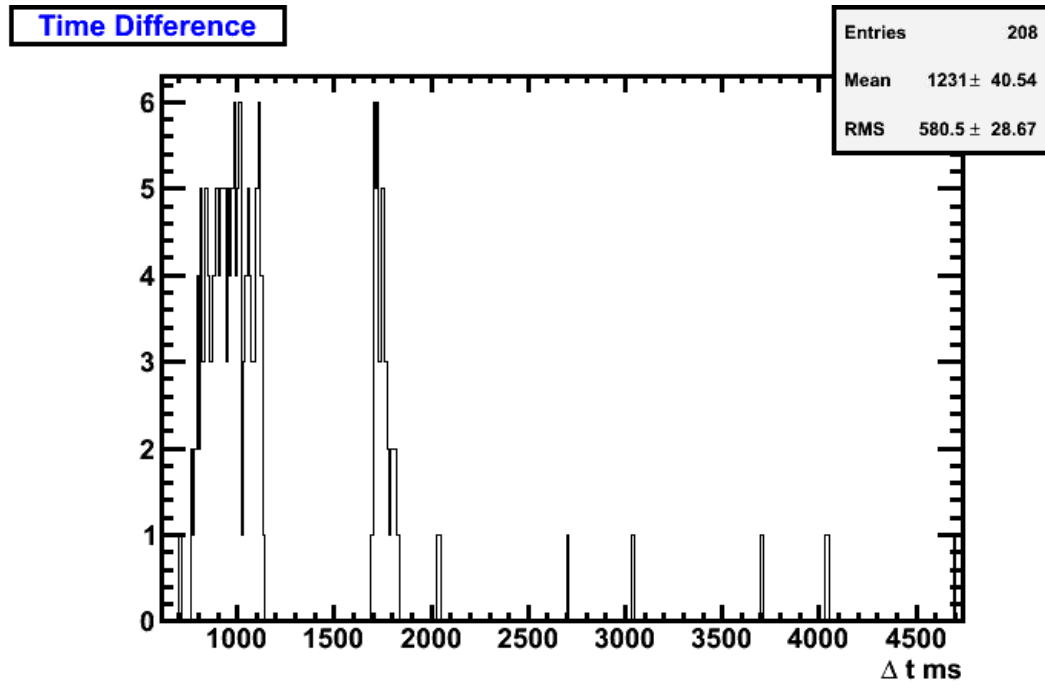


Fig. 6.2: Data Transport measurement histogram

strates that time clocks on the different computers are rather well-synchronized. The histogram is based on 110 entries and it introduces mean value of 0.003 milliseconds, which can be accepted as the measurement error. The clock synchronization inside of the .CMS network is done with the help of the time server, that can guarantee low measurement error.

On the other hand the clock synchronization of the front-end read out electronics can also affect the latency measurement. To make a clock phase assessment for front-end electronics is possible by comparing different read-out cards for one detector. The difference of the clock phase for the BCM2 detector that has two read-out cards is represented in the histogram 6.4. In order to evaluate the quality of the measurement, it is also useful to compare the BCM2 clock phase difference with the clock synchronization of the BCM1L detector (four read-out cards) represented in the figure 6.5.

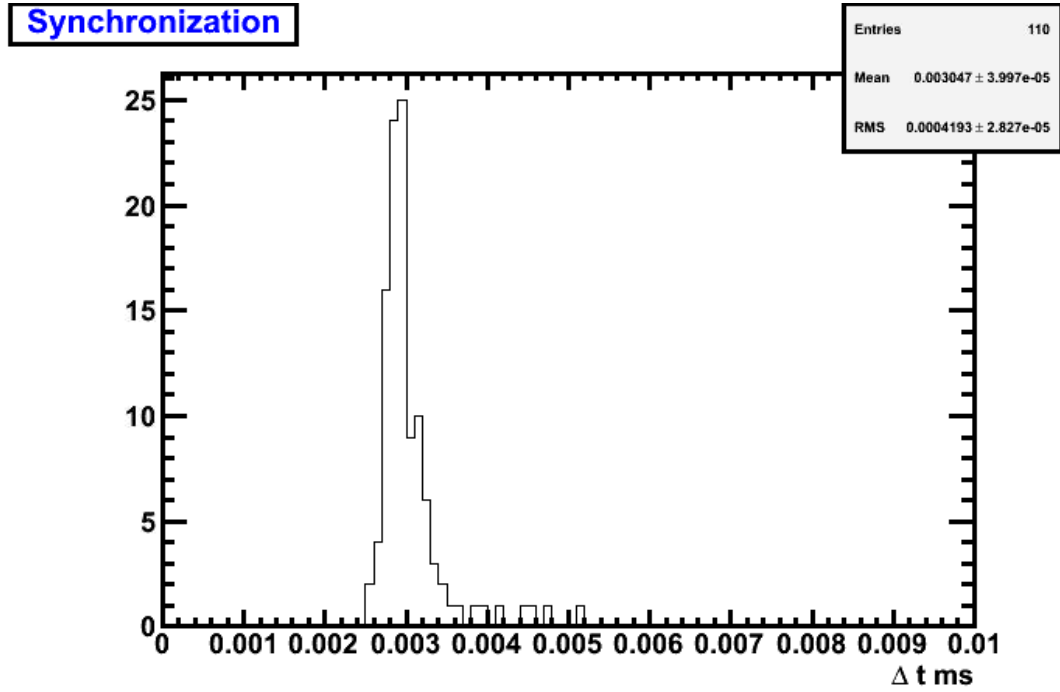


Fig. 6.3: Histogram distribution of the synchronization difference.

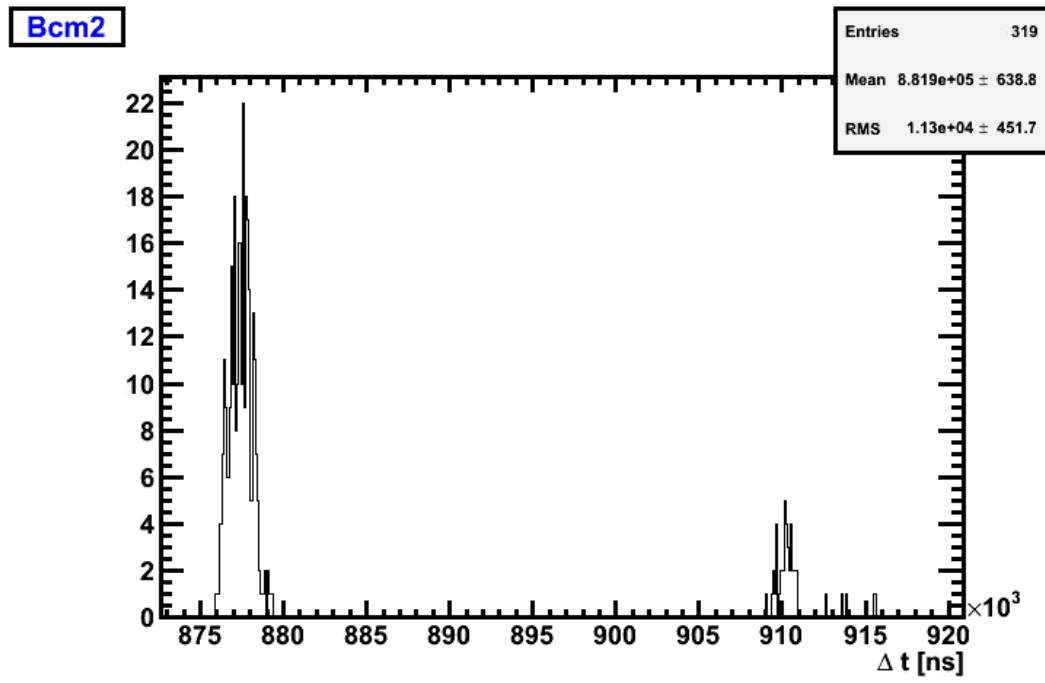


Fig. 6.4: The clock phase difference for BCM2 detector.

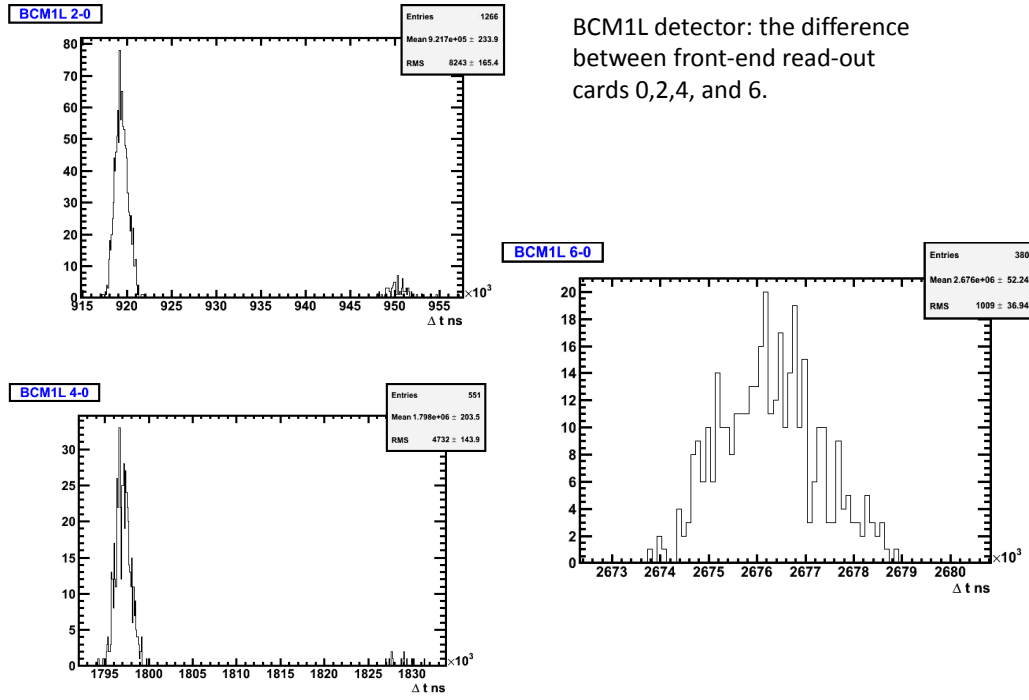


Fig. 6.5: The clock phase difference for BCM1L detector.

- Data quality assurance is another characteristic of the system that should be carefully examined in order to assess the final deliverable. Since the visualization framework is a web based application, it uses XML formatted data to ensure data quality. XML data formatting is very efficient mainly because it is a simple, logical and flexible data type that allows to transport large data amounts through the web. The other reason to use XML formatting is that it is easily parsed and processed by Java, which is significant for the application performance. It is used in the variety of different web applications, which proves it to be reliable and safe.

The visualization framework also has an additional method of preventing data losses by extracting data using different data types. Each piece of data that arrives to the framework is carefully retrieved according to its data type. This process is ensured by the data-type attribute that each data chunk is demanded to have. If the data conversion procedure has failed on the way to the monitoring framework, the value type will be set to 0, which means that this particular data chunk is corrupted and can not be used. To avoid the massive amounts of the corrupted data, every case, when the data type attribute equals 0, should be presented to the framework developers and users and carefully logged.

- As a real-time system, the visualization framework should run 24/7/52. Safety issues are vital for the system, thus it requires consistent testing and regular backup. In order to secure the backup procedure, Subversion version control system is used, that allows to conveniently manage different framework releases. Keeping up with different versions of the framework is a critical issue, because in case the fatal error occurs, using the version control system significantly decreases the down time of the system. This is an important issue due to several reasons. At first, as it was mentioned in Chapter 2, the availability of the monitoring directly inputs the CMS and LHC availability for data taking and the cost of one LHC working hour equals 10000 €. The other reason is that in case of failure, the BRM shifter or “on-call” expert is called at any time of day or night to restart the monitors.

The testing process is another important process to ensure the safety of the system, so the visualization framework testing is done in several stages. First stage is based on the various white box tests that is done by the developers as a set of manual procedures. The main purpose

of this application inspection step is to examine internal working and the structure of the visualization framework. After white box tests are completed the framework is passed to the users to implement acceptance and functional testing. This step of testing process ensures that the new version of the visualization framework meets the requirements of the end users and also validates the system behavior. Usually this stage of testing takes a longer period of time than the first one and is followed by the round of modification process, which keeps the visualization framework in the constant development process to meet the increasing user demand to the system.

6.1.1 Security Audit

Security is one of the key components of any IT system. The implementation of the visualization framework should also meet some basic security requirement. There is no reason for strict security restrictions for the monitoring because of several reasons. First of all, any service that is related to the framework is placed inside of the .CMS network and is required to use minimum amount of external resources, thus it is assumed that the access to the visualization framework is protected by the firewall. All users that can access the framework's resources have to have internal CERN authentication credentials, so it eliminates presence of any unauthenticated users of the system. At the same time the visualization framework records the implementation of key procedures to log files, that provides an opportunity to track back and analyze abnormal system activities.

There are some missing parts in the security of the framework implementation. At the moment any user that has the right to log into .CMS network, is able to write configuration files. This fact is potentially dangerous, but in practice it does not significantly decrease the security at the current state of the development.

6.1.2 Reliability Analysis

The reliability of the system can be assessed by several methods. One of the main factors that impacts the visualization framework reliability is how well the framework meets its requirements or effectiveness of the system. This factor is indicated by the quality of the design and programming solution discussed in chapter 4. The requirement for the visualization framework is compared against the actual framework characteristics were analyzed in detail in the previous sections. Summarizing the points presented there,

the general conclusion that can be drawn is that the monitoring system successfully fulfills the list of the specification at the required level. During the 2009 LHC run time, the visualization framework proved to be stable enough to successfully perform all the demanded functions.

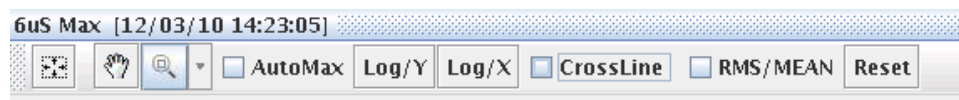
The other factor that affects the reliability of the system and closely relates to the system effectiveness is behavior assessment or efficiency. In order to be reliable, the visualization framework should consistently follow the predefined behavior of the system. This reliability factor includes the system crash rate and unexpected behavior, such as starting up using different configuration file, skipping the configurations, inconsistent update of the data, running out of cache, etc.

Behavior of the system significantly affects the usability and the user satisfaction rate. The visualization framework is an interactive real-time monitoring tool, thus it should have a clear and easy user interface in order to increase the usability. Also it is important to notice that the framework is targeted to be used by the specific group of people working at CMS control room. Thereby the visualization framework is designed and developed for the end-users who are experienced in a variety of different technologies. Widely spread and commonly recognizable technologies that are shown in the visualization framework interaction tools example 6.6 are used to simplify the interaction with the system.

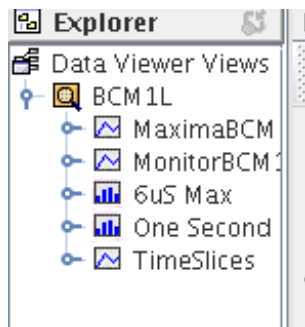
On the other hand the end user can communicate with the framework using different types of input, which are the interaction with the system through the tool bar instrument described earlier and the creation of the configuration files. The configuration files are the most significant and influential user input, so it has to have an ability to describe the monitor as thoroughly as possible and also it has to follow the usability requirements in order the system to be reliable. Thus all the configuration files are in structured document formats as XML and HTML. These kinds of document are able to present the data in the most efficient and logical way, which significantly decreases the logical errors and crash rate.

Another important criteria that reflects the success of the visualization framework is the user satisfaction rate. As it was mentioned earlier, it is closely dependent on the system performance and behavior. During the 2009 LHC run the monitoring package extensively used at the CMS control room and gathered many positive responses from the shift crew, whereby it could be concluded that the framework satisfies the main demand of the users. It is a significant result that according to the user comments, the framework has noticeably simplified the real-time monitoring of the beam of conditions and reacting accordingly to the changing situation.

Toolbar example



Tree Listing Example



DropDown Menu Example

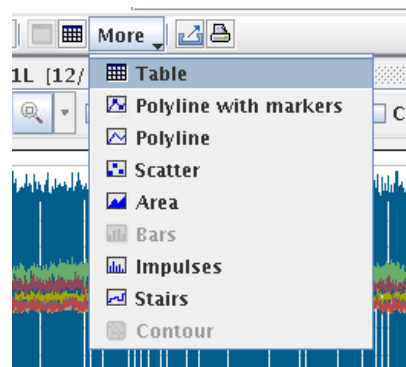


Fig. 6.6: Visualization framework interaction tools.

6.2 Usage

At the current time the monitoring framework is used by Beam and Radiation monitoring group and also tracker group, pixel group, Run Filed Manager, Shift Leaders, Technical shifters, CSC, and Luminosity group. The visualization monitors as a real-time system reflects the beam conditions in a real time with the low latency level, thus it is the first place at the CMS detector where the beam collisions are observed. Thereby it is essential to keep the monitors running all the time to provide stable 24/7 service. The crash rate of the system is low, but in case of failure, the monitors should be immediately restarted. BRM team constantly runs the set of seven displays, that uses approximately 18 DIP subscriptions. The monitors provide clear human-readable information for the real-time beam condition analysis at the Control room. They reflect the status of such BRM detector as BCM11, BCM2, BSC and BCM1F, and also provide the summary information. Examples can be seen in figures 6.7, 6.8 and 6.9. These displays are where the first collisions in 2009 were first seen and where the record high energy collisions were first observed by CMS in December 2009 and March 2010.

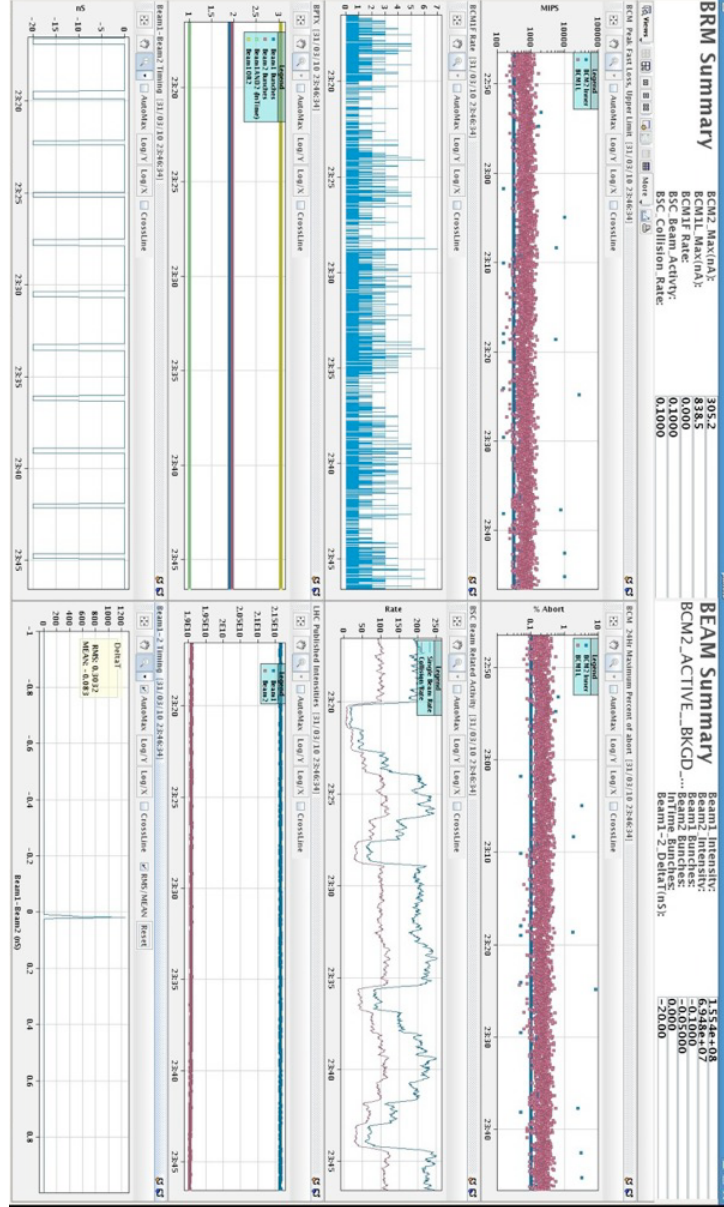


Fig. 6.7: Luminosity Scan, BRM Summary monitor, 2010 run.

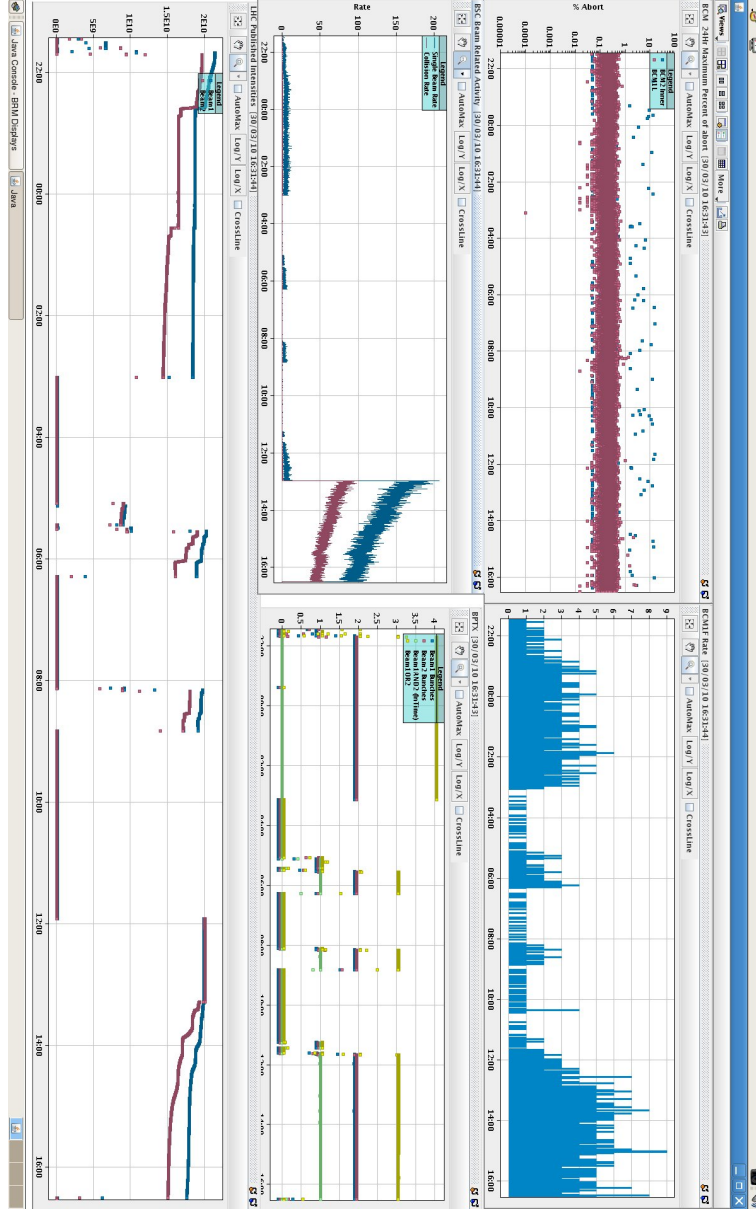


Fig. 6.8: 7 TeV Collisions at 24 Hours Summary Monitor, 2010 run.

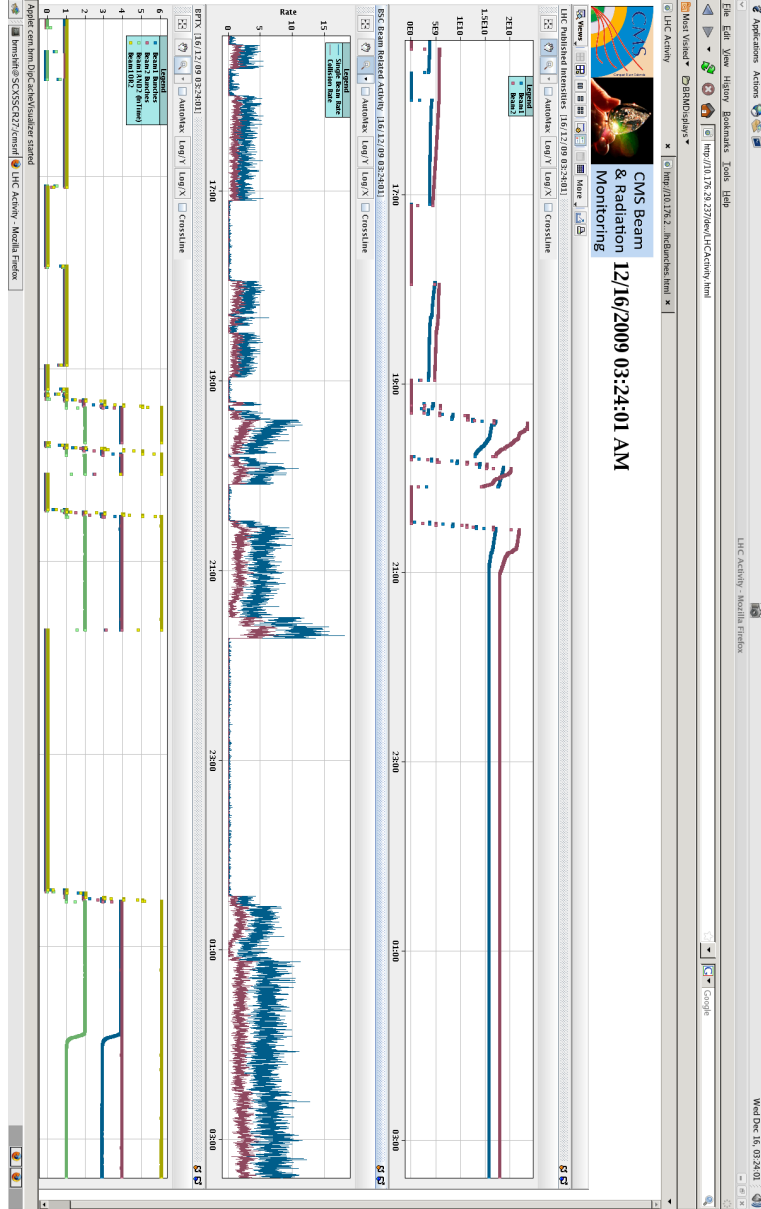


Fig. 6.9: LHC Activity at LHC Summary Monitor, 2009 run.

7. OUTLOOK AND RECOMMENDATIONS

7.1 *Assessments of the working process*

The real-time on-line monitoring framework was designed and developed as a non-commercial internal tool to satisfy the need of the Beam and Radiation monitoring system. The development of the system was fully implemented inside of the team, which set certain specific characteristics to the working process.

First of all, it is important to mention that in the confined working environment the end users of the system and the developers constantly interact with each other, which speeds up the development process and provides the stable base for the monitoring framework analysis. Each BRM team member that is involved in the shift work can be treated as a user, which also includes the developers and the designers of the system. Thus it is very difficult to clearly define project roles and responsibilities, which complicates the planing of the working process and setting the precise milestones. On the other hand close interaction between users and developers, as well as the fact that the developers had the ability to try the system in the real working environment highly contribute to the success of the project. Close communication between team members that are using and analyzing the framework constantly, reduces the verification and maintenance time.

Considering the special features of the working environment, the spiral model of software development process was used. It is the most efficient way to develop the real-time on-line monitoring framework because of the following reasons.

At first, the framework had to be in use during the LHC run at the end of 2009, so the first version of the framework was done quickly and contained only necessary features. Second reason is the ability to get the feedback from the end users and analyze the necessity of the requested features as soon as the first release of the framework is in use. It also guaranteed to have a working version of the system running all the time, while new features are being developed, which is a vital requirement for the BRM team.

Although the working process is extremely complicated and there are

many different factors that can affect the assessment of it, in the conclusion, the initial goals of the project were all met in the predefined time frame, which implies that the working process was productive and successful.

7.2 *Reflection on the working process*

As in any work of a group, development of the visualization package had points for improvement and successes. Among the advantages that speedup the production is the fact that at the moment of project initiation the BRM group already had set all the hardware resources and project basic environment. Another positive point of the working process is good communication and comfortable informal atmosphere inside of the team, that simplified the the requirement engineering and verification procedures. Also considering the fact that most of the team members are involved in the project for some small percent of their working time, good communication provided additional schedule flexibility and the time for the team to gather feedback.

As well as the positive sides, there are some disadvantages of the working process described earlier. First of all, it is the inability to plan the project from the beginning to the end. Since most of the working team is not involved in the project fully, it is quite complicated to make a initial risk studies and predict how much time a certain phase of the project will take. The same reason causes a difficulty to clearly define the milestones for each of the phase of the working process, thus the tasks that the working process consists of are not equally spread on the timescale.

7.3 *Recommendation*

The real-time on-line monitoring framework is successfully used by the Beam and Radiation Team since the end of August 2009, but new features and modifications are constantly added to the system. Currently the monitoring framework slowly switches to the java web start technologies, which significantly improves the speed and stability of the framework, and also allows to avoid many browser and security related problems. The other advantage of the the current changes is that it provides a simple way to distribute the system within CMS-network, which is very useful for the future development. For example, one of the near future goals is to develop a centralized web page that contains the links to the web start monitors, which will ensure that the end user gets the latest version of the monitors at any time.

Also the documentation for the monitoring framework is in a preparation to be released as a CMS note. It contains the short design description and

the user manual for creating the configuration files and running the displays.

The other future improvement of the monitoring system that would increase the usability and user satisfaction is the development of web interface for DIP-Cache server, that is capable to observe currently caches subscriptions, add, modify or remove the dip subscription, but the most important feature of the DIP-Cache web administration page should be the ability to create the XML configuration files for the new monitors, that would be highly appreciated by the users.

Another direction for further development is improving the usability of the system in particular implementing different set of zooming tools. The physical space that is used by the monitors is limited, so the key to better flexibility of the monitoring tools is complex and user-friendly zooming. For example, the zooming tool that is capable to zoom a certain area manually specified by the user. Another useful zooming tool the monitoring framework is missing at the moment is a zoom interactor that could zoom in some specific area and then keep refreshing then the main monitor is refreshed.

An other feature which might be a target of the future development is the ability to plot the data from a database. Of course, this is not the part of the real-time monitoring, but it could be quite useful for the analysis of the historical data.

BIBLIOGRAPHY

- [1] *CERN Official web site*, March 2010. <http://www.cern.ch>.
- [2] E. Lyndon and P. Bryant, “The CERN Large Hadron Collider: Accelerator and Experiments,” *Journal of Instrumentation* 3, vol. 3, 2008.
- [3] C. Pignard, “Online radiation monitoring for the LHC machine and experimental caverns,” *Journal of Instrumentation* 3, p. 12, 2006.
- [4] S. Chatrchyan, “The CMS experiment at the CERN LHC,” *Journal of Instrumentation* 3 S08004, 2008.
- [5] M. Hollingsworth, “DIP Cache Documentation.” in preparation.
- [6] CERN, Fermilab, *scientific linux official web site*, April 2010.
- [7] CERN, *DIP Documentation*, December 2009.
- [8] CERN, *Java DataViewer Documentation*, 2009.
- [9] C. Zamantzasl, “The LHC Beam Loss Monitoring system’s surface building installation,” *Proceedings of LECC*, vol. 4, pp. 552–556, 2006.
- [10] E. Effinger, “The LHC Beam Loss Monitoring system’s data acquisition card,” *Proceedings of LECC*, vol. 5, pp. 108–112, 2006.
- [11] B. Dehning, “The Beam Loss Monitor system,” *Proceedings of the XIII LHC Project Chamonix Workshop*, vol. 3, pp. 2667–2670, 2004.
- [12] T. Aumeyr, “Beam Phase and Intensity Monitoring for the Compact Muon Solenoid Experiment,” Master’s thesis, Vienna University of Technology, 2008.
- [13] S. Mueller, *Design, Commissioning and Performance of the CMS Beam Condition Monitor 2 and Simulation Studies of the Radiation Environment near CMS at LHC*. PhD thesis, CERN Geneva / KIT Karlsruhe. in preparation.

-
- [14] B. Todd, *A Beam Interlock System for CERN high energy accelerators*. PhD thesis, Brunel University, 2006.
 - [15] A. J. Bell, “Design and Construction of the Beam Scintillation Counter for CMS,” Master’s thesis, University of Canterbury, Christchurch, 2008.
 - [16] A. J. Bell, “Beam and Radiation Monitoring for CMS,” in *Nuclear Science Symposium Conference Record*, no. 2322 in 3, IEEE, 2008.
 - [17] W. Lohmann, “Fast Beam Conditions Monitor BCM1F for the CMS Experiment,” vol. 15, 2009.
 - [18] L. Fernandez-Hernando, “Development of a CVD diamond beam condition monitor for CMS at the large hadron collider,” *Nuclear instruments and methods in physics research*, vol. 6, pp. 183–186.
 - [19] A. Macpherson, “Beam condition monitoring and radiation damage concerns of the experiment,” *Proceedings of the XV LHC Project Chamonix Workshop*, vol. 5, pp. 258–263.
 - [20] D. Chong, “Validation of synthetic diamond for a beam condition monitor for the compact muon solenoid experiment,” *IEEE Trans. Nucl. Sci.*, vol. 54, pp. 182–236, 2009.
 - [21] M. Hollingsworth, *Generalized Unilateral Transfer Server Framework*. CMS Detector note.
 - [22] ETM, *PVSS Concepts*.
 - [23] K. Kostro, “The controls middleware (CMW) at cern status and usage,” *Proceedings of ICALEPCS*, vol. 4, 2003.
 - [24] M. Peryt and M. M. Marquez, “Fesa 3.0 : Overcoming the XML/RDBMS impedance mismatch,” *12th International Conference On Accelerator And Large Experimental Physics Control Systems*, vol. 4, 2009.
 - [25] B. Eckel, *Thinking In Java*. Upper Saddle River, NJ: Prentice Hall PTR, 2006.
 - [26] Sun, *org.xml.sax Documentation*, February 2010.