An Information System to Access Status Information of the LHCb Online

M.Frank, C. Gaspar
CERN, 1211 Geneva 23, Switzerland
E-mail: Markus.Frank@cern.ch

Abstract. The LHCb collaboration consists of roughly 700 physicists from 52 institutes and universities. Most of the collaborating physicists - including subdetector experts - are not permanently based at CERN. This paper describes the architecture used to publish data internal to the LHCb experiment control- and data acquisition system to the World Wide Web. Collaborators can access the online (sub-) system status and the system performance directly from the institute abroad, from home or from a smart phone without the need of direct access to the online computing infrastructure.

1. Introduction

LHCb is a dedicated B-physics experiment at the LHC collider at CERN [1]. LHC delivers proton-proton collisions at a centre of mass energy of up to 14 TeV to the LHCb detector, at a rate of 40 MHz. The LHCb collaboration consists of roughly 700 physicists from 52 institutes and universities located mostly in Europe. Most of the collaborating physicists are not permanently based at CERN, but rather visit CERN more or less frequently for data taking- or subdetector shifts and meetings to discuss problems or physics results. They though want to participate in the collaboration's life centered around the experimental setup, which requires close monitoring and constant maintenance.

The LHCb experiment consists of 10 subdetectors each having a largely independent infrastructure which consists of the subdetector control infrastructure for cooling, high voltage control, front end control, readout control etc. To ensure the proper functioning these subsystems need to be monitored. Most of this monitoring activity is automated by the experiment controls system. However, not all possible problems are known a priori and human checks from time to time help to spot unclassified failures. For this purpose in the LHCb control room each subdetector displays detailed information about the status of the individual components on dedicated screens or subdetector experts start the required user interface to access any piece of information required. To overcome the limitation of a logon to the online computing infrastructure a system is necessary, which requires nothing but the bare infrastructure of any typical personal computer available today. In the following the mechanism allowing all collaborators to access online information such as the overall detector and data taking status or subdetector experts to access the most urgently required information on a given subsystem using normal web browsers is presented. First the architecture and the implementation is described, which is used to publish data internal to the LHCb experiment control system (ECS) and the data acquisition system to the World Wide Web. Then examples are shown how online (sub-)system status and the system performance are presented to collaborators abroad in the institute, at home or to their smart phone.

2. Architectural Concepts

The ECS, implemented using the commercial product PVSS [2], handles the configuration, monitoring and operation of all experimental equipment involved in the different activities of the experiment:

- The Data Acquisition System (DAQ): readout boards, HLT-, monitoring farm
- Timing and fast control system (TFC) and hardware trigger (L0)
- The Detector Control System (DCS): gases, high voltages, low voltages, temperatures, etc
- The Experiments Infrastructure: magnet, cooling, electricity distribution, detector safety, etc
- Interaction with the outside world: LHC Accelerator, CERN safety system, CERN technical services, etc.

The relationship between the ECS and other components of the experiment is shown schematically in figure 1. All ECS data in PVSS is organized as data points, i.e. data-values identified throughout the entire system by a unique name. The value of each data point is accompanied by its data type as a primitive atomic value such as an integer, a float, a string etc. Due to the large number of PVSS data items representing devices, I/O channels or Finite State Machine (FSM) entities, the acquisition and monitoring of the data managed by PVSS is done in parallel and distributed over several machines.

All macroscopic entities of the ECS, which belong to a given subdetector, are modeled as FSMs, grouped together in a tree-like structure. The state of every higher level node summarizes the state of its children. The same FSM tree mechanism is used to describe the functioning of the processor farms, where at the lowest level processes are modeled as FSM entities, a set of processes is grouped to a node, a set of nodes describes a subfarm and finally the high level trigger is described as the set of subfarms. The state of any of these FSM entities is stored in a PVSS data point. This architecture of the LHCb data

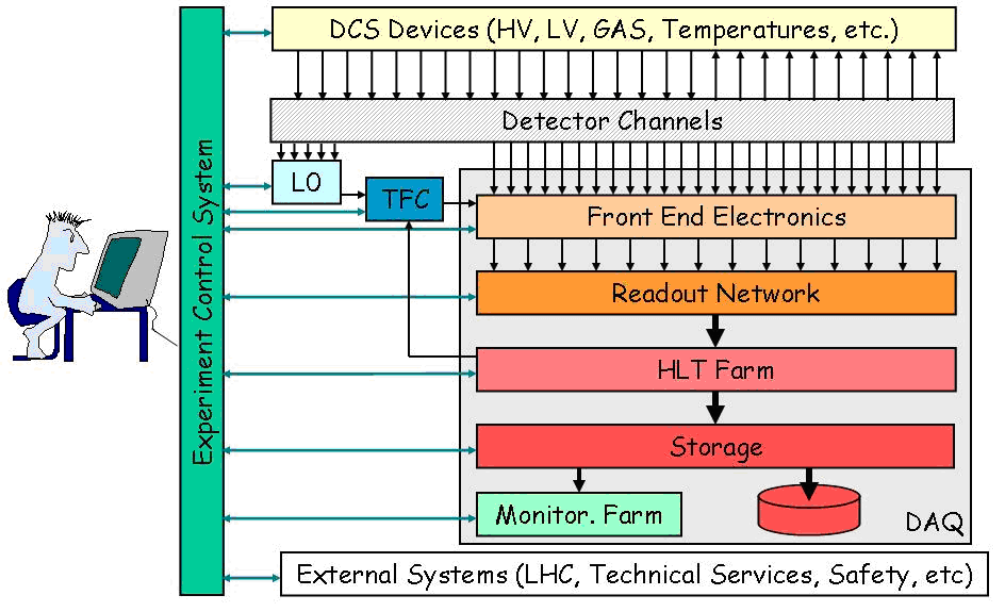

Figure 1. The scope of the Experiment Control System. All elements are controlled by PVSS, the state and the setup parameters of all components is accessible from PVSS and consequently can be published and monitored.

acquisition and experiment controls system has an important consequence: the ECS provides a unique interface between the user and all experimental equipment. In the online system users can connect to

any of these systems with graphical user interface applications and analyze the status of any system of interest. The mechanism used to extract data from the ECS and publishing this information to another medium like the World Wide Web is nothing but a specialized user application.

The functional description of the basic mechanism is as follows:

- The user requests information about the data content of an ECS data point by opening a web page.
- The user application (executing in the web browser) subscribes to updates of a given data point and sends this request back to the web server which provided the current web page. The data point is identified by its PVSS name. To limit any possible overload of the online system, the decision which data items are exported is controlled at the server side. In fact first must be actively marked for publication.
- The request is processed at the server side, the mapping between the subscriptions is remembered and any change of the data point value is sent to the web client. The transfer mechanism between the web browser and the server is based on a comet data push protocol [3] rather than the traditional pull mechanism browsers use to receive data from the server.
- The data transfer to the client terminates if the connection is closed. This happens either if the client itself terminates or moves to another web page.

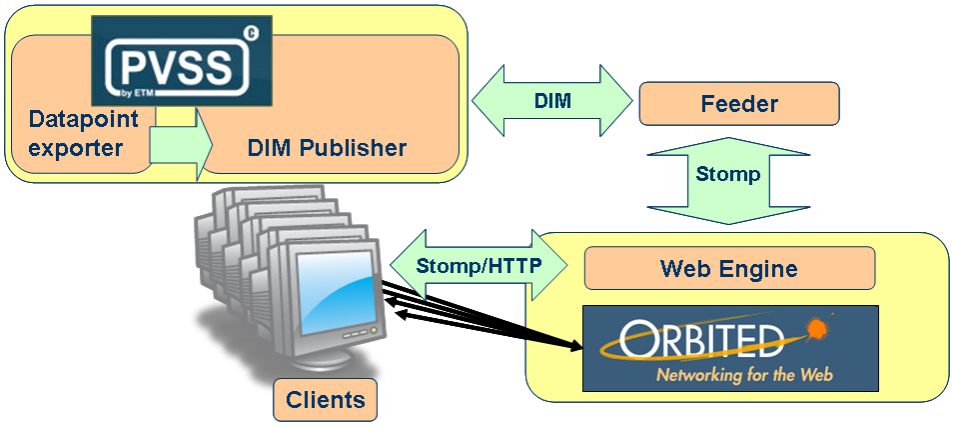

Figure 2. The different interacting components. Shown on top is the data source hosted by PVSS, which supplies the Feeder application with updates to data-point values, which in turn are fed into the Web-Engine. The Clients prior subscribe to the data items and receive the updates from the Web-Engine.

There are many possibilities to implement such a mechanism. In the presented implementation an attempt was made to minimize code maintenance for manpower reasons and hence to optimize the use of off-the-shelf components from open source projects. Taking into account this boundary condition, the analysis and decomposition of the above functional requirements led to the need of the software components shown in figure 2 with the following functionality:

- The *Datapoint Exporter*: A dedicated ECS/PVSS graphical user interface application (GUI) allows the administrator (in our case anybody having access to the online system) to mark a given data point or the state of a FSM entity for publication. To minimize the impact on the connected PVSS systems, which are required for data taking, the export mechanism internally copies the data values to be exported to a local PVSS project. PVSS has built-in mechanisms to automatically updated the local datapoints whenever the source value changes. The local values are always in string format, prepended by the atomic data type as a processing hint for connected web clients. The Publisher subscribes to these data points. FSM states are internally modeled as complex data points with a separate type. The GUI had to be developed; a snapshot is shown in figure 3.

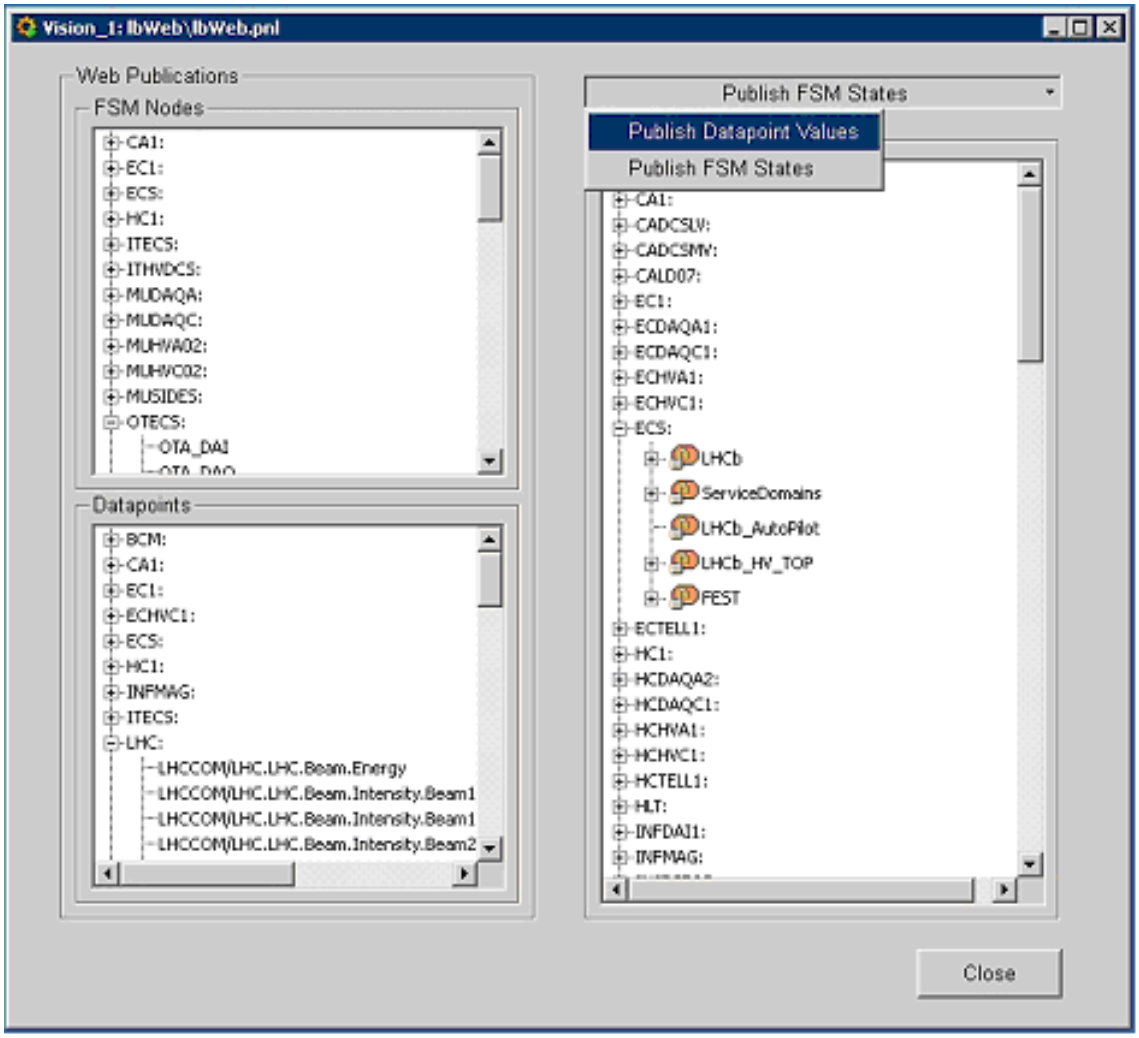

Figure 3. A snapshot of the GUI application which allows to export PVSS data points. Data points or FSM states to be exported are selected from the pane to the right. The two vertical panes on the left allow to browse entities already marked for export.

- The *Publisher*: This process publishes the value of an arbitrary PVSS data-point marked for export in a predefined format to the network on the online domain using the DIM protocol [4] - an existing publish-subscribe mechanism already used multiple times within the LHCb online system.
- The *Feeder* application has a two-fold functionality: the application listens to subscription requests from the *Clients* (web browsers). On the occurrence of a request for data the *Feeder* process subscribes to updates provided by *Publisher*. All updates for the subscribed data-feeds are forwarded to the *Web-Engine* using an externally provided software solution [5], which simplified the development of the *Feeder*. The desired reuse of existing software already used in the online system requires an implementation in C or C++.
- The *Web-Engine* [6] has two responsibilities: firstly it listens to user requests to serve html pages and comet data, secondly it listens to updates of the *Feeder* and forwards changes to the client. The *Web-Engine* is message queue system, where the *Feeder* provides the message feeds, with a fan-out functionality to the clients. The *Web-Engine* is provided by open-source components, which only

required a slightly modified configuration to support the desired functionality. The data between the *Feeder*, the *Web-Engine* and the *Clients* are transferred using the Stomp-protocol [7].

- The *Client*, which consists of a set of html pages and the corresponding JavaScript code. The code dealing with the comet communication to the *Web-Engine* was provided externally. However, the DOM tree of these pages is created programmatically and had to be developed. In fact this part required most of the effort, not due to its complication, but rather the number of different information pages. This task cannot be handled generically and similar to the development of display applications using other GUI frameworks requires extra work for each page.

Only the data publication mechanism depends on PVSS. All other components like the *Feeder*, the *Web-Engine* and the *Client* implementations do not depend on proprietory software. The system can easily be enhanced to also serve data from other data sources. An example of such an extension can be seen at the bottom of figure 4: the extract of the logbook is published from a python script reading the latest entries from the online log-book.

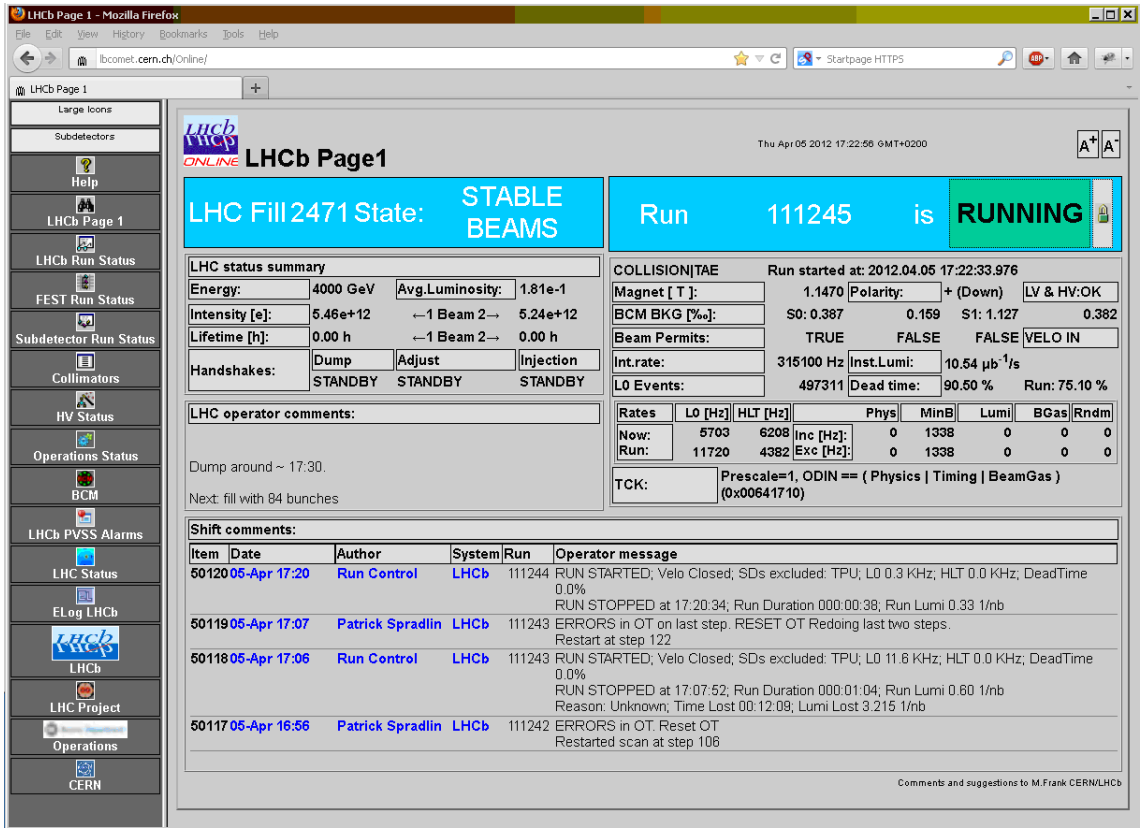

Figure 4. A snapshot of the LHCb Page 1.

3. Client Page Development

Various client pages were developed. Collaboration members, who are not directly involved in the operation of the experiment, are mostly interested in the experiments status summary (LHCb Page 1). This page shows basic performance numbers of the LHCb detector and the LHC accelerator, such as status, fill and run number, beam conditions and the detector state. Trigger rates, background conditions and the latest entries of the shift log-book shown in the display allow to quickly assess the status of the data taking activity. A snapshot is shown in figure 4.

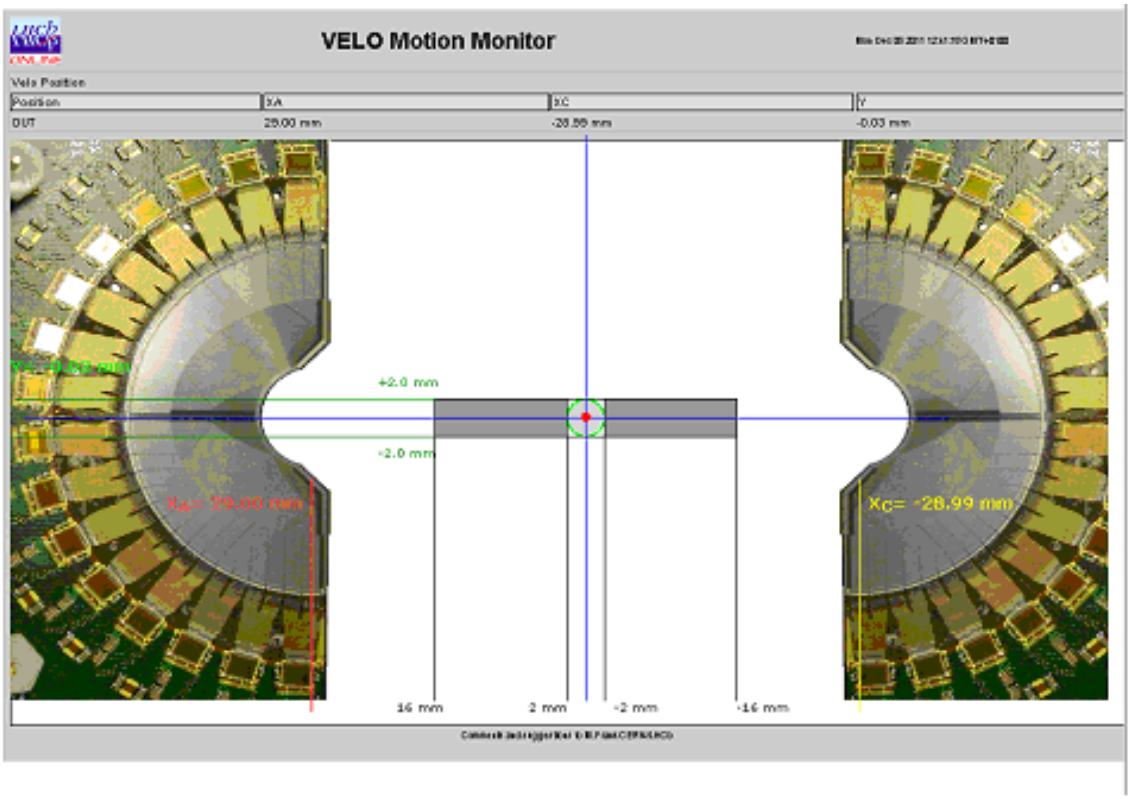

Figure 5. The motion monitor display of the VELO detector. In the display only two sensors corresponding to one layer are shown. The complete detector has 21 layers.

For more detailed information expert pages are provided, which display for example the state of the run-control [8] or mimic the real panel of Detector Controls System (see figure 6). Other pages are availible which show the status of the readout boards for the various subdetector, the trigger status, the LHC status or the alarms reported by the various PVSS systems to the central console. Figure 5 demonstrates that the JavaScript implementation is sufficiently flexible to also implement animated displays. In the motion display of the vertex detector (VELO), the picures of the 2 sensors move according to their true location with respect to the LHC beam.

4. Extensibility and Technology Choices

The choice of Orbited [6] for the Web-Engine was driven by the following considerations:

- The possibility to implement the Feeder in C/C++.
- Orbited embeds a fully functional message queuing implementation (MorbitQ), which requires only small configuration. It intrinsically understands the Stomp protocol.
- It provides a JavaScript web client library supporting the communication between the clients and the server.
- It supports the extension to a deferred message queuing mechanism. This option is important since the performance of the MorbitQ is limited. In the LHCb implementation currently an ActiveMQ instance [9] is used to serve the fan-out of data point updates. ActiveMQ supports data feeds using a large variety of protocols. *Feeders* can be implemented using a whole palette of client libraries.

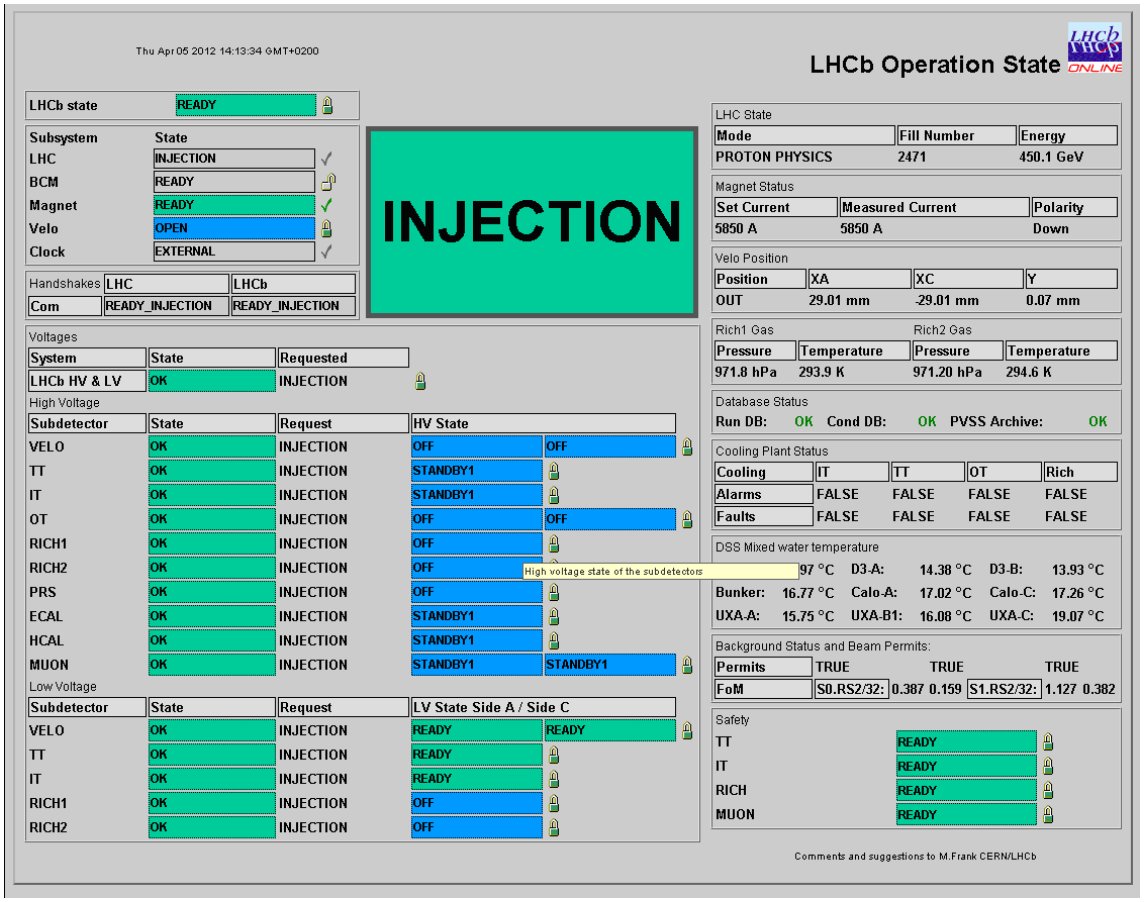

Figure 6. A snapshot of the Detector Controls Status display.

5. Conclusions

The status of the LHCb experiment was made available to colleagues in the collaboration, which cannot be present in the control room. The publishing mechanism requires only standard web browsers with JavaScript capability. Pages were provided, which show the overall status of the experiment as well as specialized expert pages. For the realization of this project special emphasis was put on the leverage of existing software or existing open-source components. This minimized the work required during the development, but also the maintenance since many of the components only require some configuration, which tends to be stable. The system is now used daily as soon as the experiment is taking data. Typically O(20) clients are connected, but tests have shown, that also several hundred clients can be served without problems.

References

[1] LHCb Collaboration, LHCb the Large Hadron Collider beauty experiment, reoptimised detector design and performance CERN/LHCC 2003-030

[2] PVSS-II, [Online]. Available: http://www.pvss.com

[3] Comet Maturity Guide, http://cometdaily.com/maturity.html

[4] C. Gaspar et al., DIM, a portable, light weight package for information publishing, data transfer and inter-process communication Computer Physics Communications 140 1+2 102-9.

[5] CMS: C++ Messaging Service, http://activemq.apache.org/cms

[6] Orbited: Networking for the Web, http://labs.gameclosure.com/orbited2

[7] Stomp: Simple Text Oriented Messaging Protocol, http://stomp.github.com

[8] F.Alessio et al., The LHCb Run Control, Computing in High Energy Physics 2009, Proceedings
[9] Apache ActiveMQ, http://activemq.apache.org