Journal of Physics: Conference Series **664** (2015) 062064 doi:10.1088/1742-6596/664/6/062064

The new ALICE DQM client: a web access to ROOT-based objects

B von Haller[1], F Carena[1], W Carena[1], S Chapeland[1], V Chibante Barroso[1], F Costa[1], C Delort[1], E Dénes[2], R. Divià[1], U Fuchs[1], J Niedziela[1,3], G Simonetti[1], C Soós[1], A Telesca[1], P Vande Vyvre[1] and A Wegrzynek[1,3] for the ALICE Collaboration

[1] European Organization for nuclear Research (CERN), Geneva, Switzerland
[2] KFKI Research Institute for Particle and Nuclear Physics, Wigner Research Center, Budapest, Hungary
[3] Warsaw University of Technology, Poland

E-mail: barthelemy.von.haller@cern.ch

Abstract. A Large Ion Collider Experiment (ALICE) is the heavy-ion detector designed to study the physics of strongly interacting matter and the quark-gluon plasma at the CERN Large Hadron Collider (LHC). The online Data Quality Monitoring (DQM) plays an essential role in the experiment operation by providing shifters with immediate feedback on the data being recorded in order to quickly identify and overcome problems.

An immediate access to the DQM results is needed not only by shifters in the control room but also by detector experts worldwide. As a consequence, a new web application has been developed to dynamically display and manipulate the ROOT-based objects produced by the DQM system in a flexible and user friendly interface.

The architecture and design of the tool, its main features and the technologies that were used, both on the server and the client side, are described. In particular, we detail how we took advantage of the most recent ROOT JavaScript I/O and web server library to give interactive access to ROOT objects stored in a database. We describe as well the use of modern web techniques and packages such as AJAX, DHTMLX and jQuery, which has been instrumental in the successful implementation of a reactive and efficient application.

We finally present the resulting application and how code quality was ensured. We conclude with a roadmap for future technical and functional developments.

1. ALICE

ALICE [1] is a general-purpose detector dedicated to the study of heavy-ion collisions at the CERN LHC. It is optimized to study the properties of the deconfined state of quarks and gluons produced in such collisions known as quark-gluon plasma [2]. It is also well-suited to study elementary collisions such as proton-proton and proton-nucleus interactions. Robust tracking and particle identification in a wide p_{T} range are ensured by different types of detectors designed to cope with high particle multiplicities. The commissioning of the experiment was carried out during 2008 and 2009 in the underground experimental pit. Since the startup of the LHC in November 2009 and till the long shutdown in February 2013, the experiment has been successfully taking data, almost continuously.

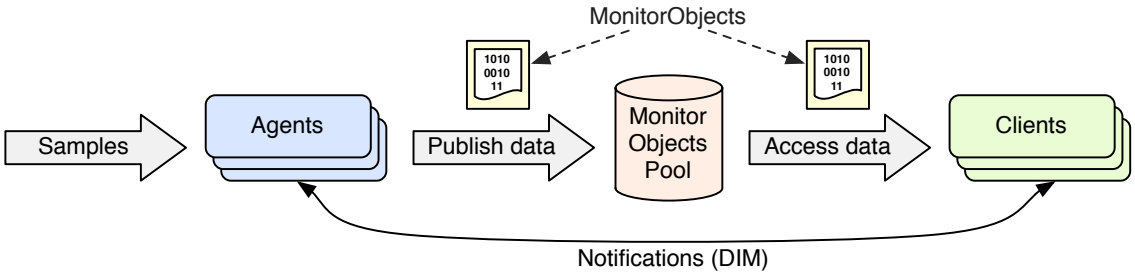

Figure 1. Publisher-subscriber data flow in AMORE.

2. Data Quality Monitoring

Data Quality Monitoring (DQM) is an important component of every high-energy physics experiment, especially in the LHC era where the detectors are extremely sophisticated devices. To ensure recording of high-quality data, one needs an online feedback on the quality of data actually being recorded for offline analysis. The DQM system provides this feedback and helps shifters and experts to quickly identify potential issues. It involves the online collection of data, their analysis by user-defined algorithms and the storage and visualization of the produced monitoring information.

3. AMORE

A DQM software system, called AMORE [3, 4] (Automatic MOnitoRing Environment), was developed for the ALICE experiment. It is based on the data analysis framework ROOT [5] and uses the monitoring library of DATE (Data Acquisition and Test Environment) [6]. AMORE is based on the publisher-subscriber paradigm (see Fig. 1) where a large number of processes, about 60 "agents", execute detector-specific decoding and analysis on raw data samples and publish their results in a Monitor Objects pool. Clients can then connect to the pool and visualize the monitoring results through dedicated user interfaces.

The data samples feeding the agents might come from the data acquisition nodes, from other agents or from files. They are also generated within other systems such as the High-Level Trigger or the Detector Control System and can as well feed directly the data pool without going through an agent. The resulting quantities, generally histograms although there is no restriction on their type, are published encapsulated in MonitorObjects. A MonitorObject is a ROOT object which contains metadata, for example the quality and the target audience of the object. Finally, the only direct communication between publishers and clients consists of notifications by means of the DIM (Distribution Information Management) system [7].

AMORE uses a plug-in architecture to avoid any framework dependency on users' code. Users, usually detector teams as well as the framework developers themselves, implement specific code that is built into dynamic libraries called modules. They are loaded at runtime by the framework if, and when, it is needed.

3.1. Run 1 clients

During Run 1 - the ALICE data taking period between 2009 and 2012 - DQM shifters and detector experts inspected the histograms and monitoring objects mainly using the AMORE clients developed in C++ with ROOT. More than 10000 objects were published and updated at least every minute during data taking. The main GUI was a generic client to browse and visualize the published objects organized in a tree and which could be filtered according to specific parameters (see Fig. 2). In addition, detector experts used specialized user interfaces. The ALICE electronic logbook [8] is another tool giving access to pictures of the monitoring

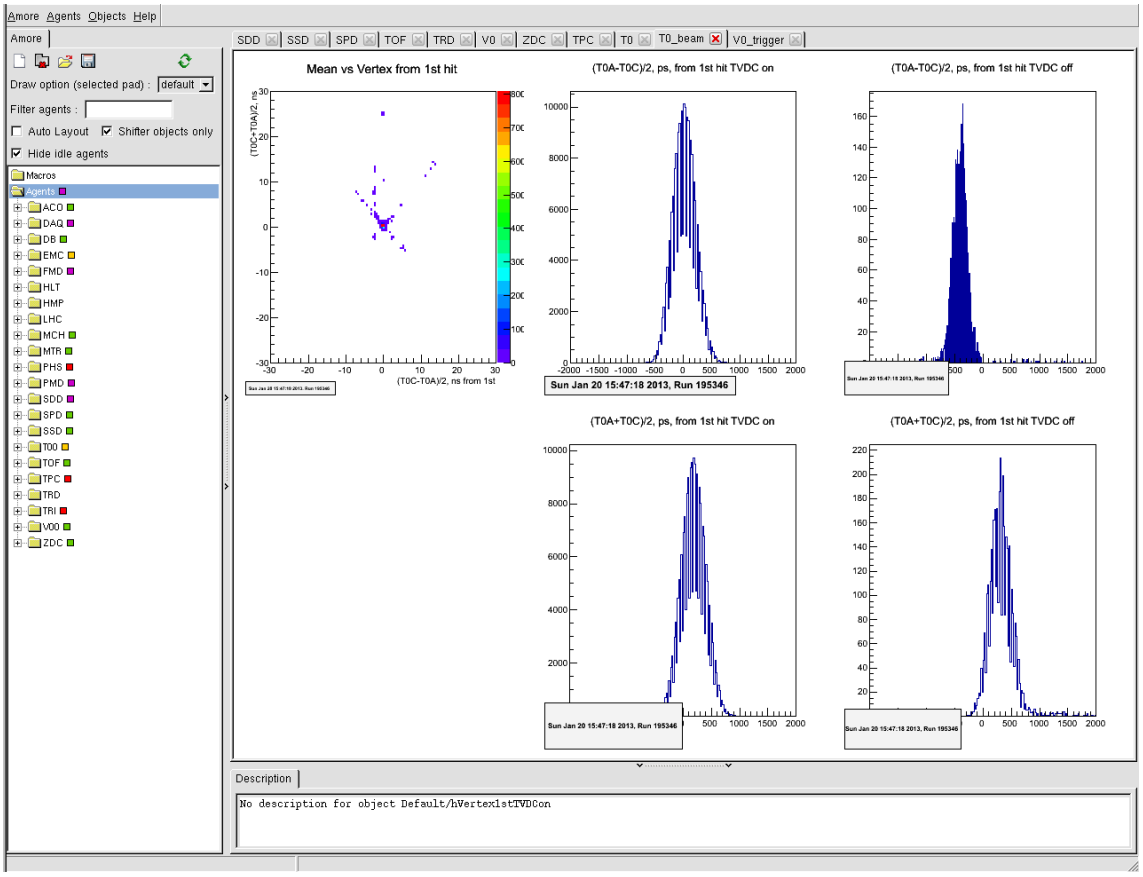

Figure 2. The generic user interface of AMORE showing information on the interaction vertex.

objects via its web interface (see Fig. 3). It also gives the possibility to download the AMORE objects for further manipulation. The main limitation of the clients which were used during Run 1 was the restriction to access results from outside the ALICE Control Room or the loss of important features by doing so.

4. DQM Web client

A web client for the DQM results appeared to be the ideal solution to provide access to the monitoring results to any member of the Collaboration at any moment. The main requirement though was to provide the same features as with a desktop based ROOT application. Indeed pictures generated from the objects and available in the electronic logbook proved to be unpractical and heavily cpu consuming for the agents. The new ROOT Javascript I/O library (JSRootIO) [9, 10], and its companion web server, provided the ideal solution. Table 1 summarizes the different alternatives to access the monitoring results and their advantages and disadvantages.

4.1. Description

The ALICE DQM client is made of a hierarchical view of the monitoring agents and their published objects (see Fig. 4), and a panel where the objects can be drawn and manipulated. The display can be organized in several tabs and the objects laid out as it fits best the user. The arrangement of tabs and objects can be saved as dashboard to be reloaded at a later time

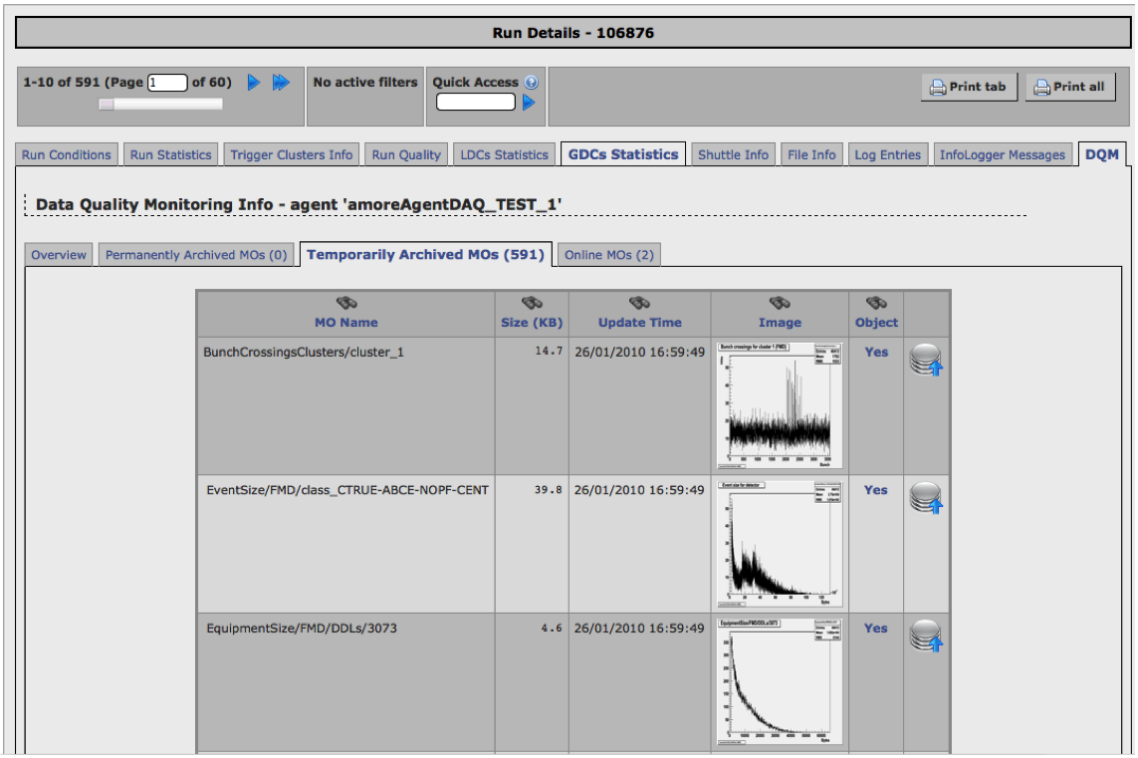

Figure 3. The ALICE electronic logbook displaying DQM information.

Table 1. Features of the different alternatives to access the DQM results. Background colors indicates whether it is positive (green) or negative (red).

	ROOT desktop app	Pictures on web page	Web access to ROOT objects
Impact on agents	None	Heavy CPU usage	None
Image resolution	Good	Limited	Good
Worldwide access	Control room only	Yes	Yes
Full software stack required	Yes	No	No
Drawing option, zooming	Yes	No	Yes

or shared amongst users.

When a tree item is double-clicked the corresponding object is displayed in the selected canvas. The look and feel is very similar to the one of a ROOT standalone application. A contextual menu gives access to functions such as zooming, changing the scale of the axes or adding the statistics box. One can also change the drawing options. The objects have an optional description attached to them that is displayed below the tabs.

The objects in the tree can be filtered by name and by attributes. Their background color indicates their quality as defined by the agents, and their icons their types.

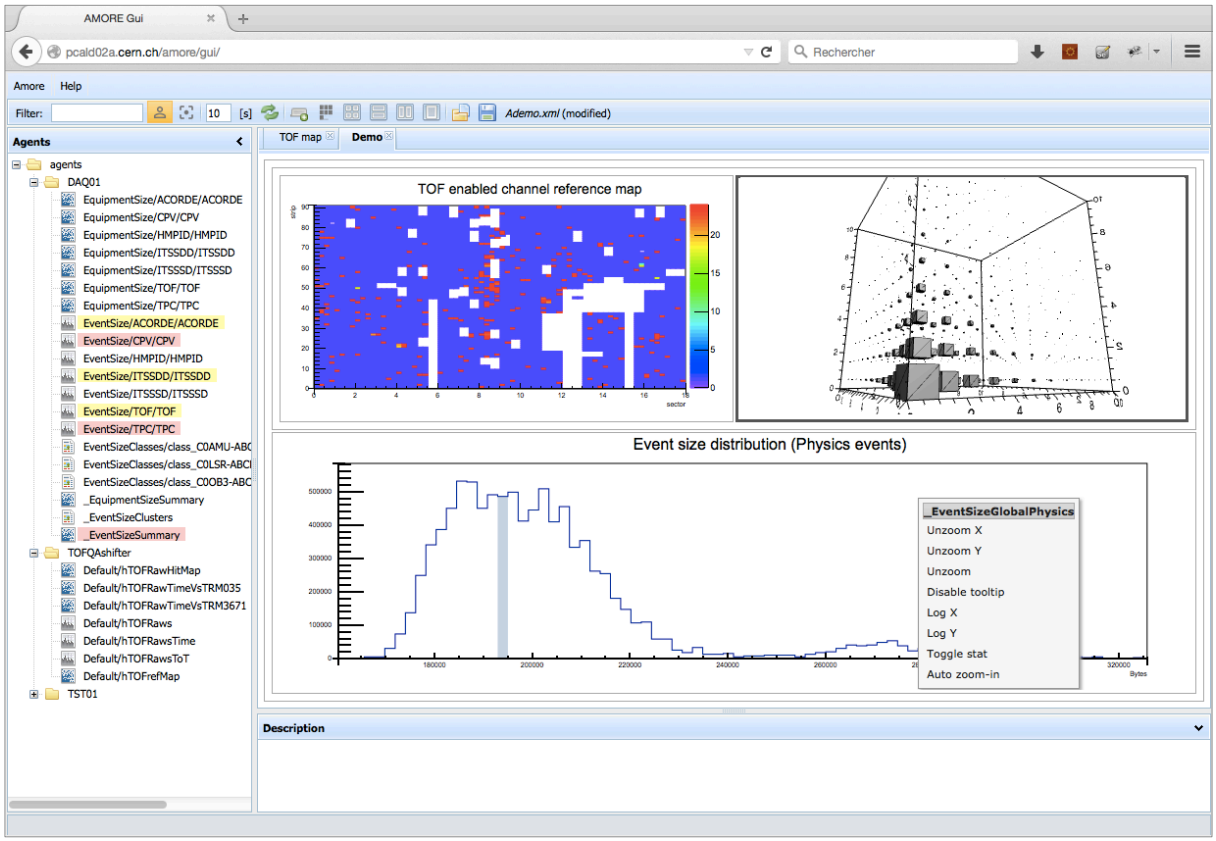

Figure 4. The ALICE DQM web client.

4.2. Architecture

The User Interface has been developed using DHTMLX [11] which provides a set of widgets greatly simplifying the implementation. For example the tree showing the agents and their objects can seamlessly display thousands of items using lazy loading and partial data loading, with only a couple of lines of code. The javascript library jQuery [12] was used to ease the Javascript development and to handle AJAX calls such as the ones to update plots or refresh the layout.

A standard Apache web server runs on port 80 and handles most of the requests coming from the web client, such as loading the index page and the agents tree or using saved layouts (see Fig. 5). It uses PHP and relies on the pre-existing AMORE RESTful API [13].

The recent developments in ROOT, with regards to the web technologies, offer the possibility to run a simple web server directly from within a ROOT application. Using the new class *THttpServer*, the AMORE web server has been developed to handle requests to load and display ROOT monitoring objects stored in the AMORE database. It runs separately from the Apache web server and takes care of the specific requests for ROOT objects. Whenever it loads an object from the database into memory, it makes it available for the clients; they retrieve and display it by using the library JSRootIO. As a result, when a user selects an object, a first call is made to load the ROOT object in memory. Once the call returns, an iframe is used to load a standard page that the AMORE web server provides for any object in memory. This second call will result in loading a basic html, the javascript libraries and the JSON data.

The authorization and authentication relies on the CERN single sign-on solution (SSO) [14].

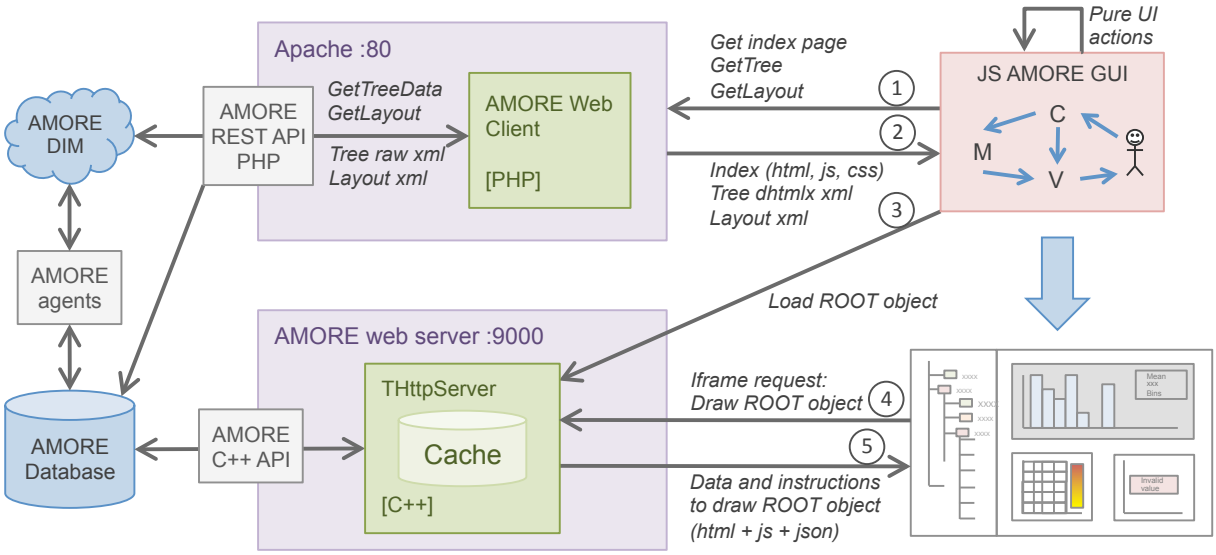

Figure 5. Architecture of the DQM web client. Numbers indicate the order of a typical request-response between the browser and the web servers.

The integration with Apache is simple and it relies on the CERN e-groups to have flexible access rights. Rather than developing an interface between the AMORE web server and the CERN SSO, we plan on using Apache as a unique entry point (see Sec. 5).

4.3. Software quality assurance

The quality of the software was ensured and improved with a number of procedures and tools.

The code has been regularly statically analyzed to detect the trivial, yet unavoidable, mistakes which can appear during development. The PHP code was tested first with PHP Mess Detector [15] that looks for several potential problems such as possible bugs, suboptimal code and unused expressions. A tool called RIPS [16] was used to detect vulnerabilities and security issues. The Javascript code was tested with JSHint [17]. It reports about commonly made mistakes and potential bugs like a syntax error, a bug due to implicit type conversion or leaking variables.

Performance evaluation was carried out using Firebug [18] and YSlow [19]. The aim was to know how long the application was taking for typical use cases, in particular its initial loading, and in which part(s) of the code the time was spent.

Finally, the application as a whole was tested using Selenium [20]. This tool gives the possibility to record and to code a series of actions, sending requests and catching the responses. A copy of the database is used and reinstalled before each test to reproduce the results. It is the most complex, but also the most realistic tests, as they behave like a user and exercise the full software chain. The Selenium tests are now ran within the Jenkins [21], a continuous integration system.

5. Future

Having two web servers made the development faster and allowed to have a clean separation of concerns. However, it also creates cross-domain requests. Using fastCGI [22] for the AMORE web server will avoid this issue by directing all the requests via the Apache web server. Moreover, it will considerably simplify the authentication and authorization as Apache will be the single

entry point of our system and thus can handle it for any request.

The use of iframes is satisfactory and leverages the existing and default ROOT object display. Nevertheless, an approach with only `divs` and Javascript will be tested as it could possibly bring a significant performance improvement. Along with this test, a number of more complex commands will be provided for the users which will require the execution of some specific code on the server side. Typically one will be able to make a fit or to add extra elements on a plot such as a line or a pave.

On the short term, and following the public tests of the client, the DQM web client will be commissioned in production and users encouraged to use it extensively.

6. Conclusions

The new ALICE DQM web client allows users to assess the quality of the data being recorded using a hierarchical view and the possibility to display and manipulate ROOT monitoring objects directly within a web browser. It requires no prior or extra installation. The ongoing tests have proven that the system is stable and that the client provides the same functionality and performance as the existing desktop-based application.

The software quality was ensured by a number of methods and tools, which will also ease future developments and maintenance.

The new software will be installed in production during 2015 in order to be used during the LHC Run 2.

References

[1] The ALICE Collaboration et al. 2008 The ALICE experiment at the CERN LHC *JINST* **3** S08002
[2] Cabibbo N and Parisi G 1975 *Phys. Lett. B* **59** pp 67-69
[3] von Haller B et al. 2009 *Real Time Conference, 2009. RT '09. 16th IEEE-NPSS* pp 354-360
[4] von Haller B et al. 2014 *J. Phys.: Conf. Series* **513** 012038
[5] Brun R and Rademakers F 1997 ROOT - An Object Oriented Data Analysis Framework *Nucl. Instr. Meth.* **A389** pp 81-86
[6] Carena F et al. 2011 *J. Phys.: Conf. Series* **331** 022028
[7] Gaspar C, Dönszelmannand M and Charpentier P 2001 *Computer Physics Communications* (Padova) **140** pp 102109
[8] Carena F et al. 2010 *Real Time Conference (RT), 2010 17th IEEE-NPSS* (Lisbon) pp 1-5
[9] Bellenot B 2012 *J. Phys.: Conf. Series* **396** 052011
[10] Bellenot B and Linev S 2014 *J. Phys.: Conf. Series* **513** 052005
[11] DHTMLX - `http://www.dhtmlx.com`
[12] jQuery - `http://www.jquery.com`
[13] REST - `http://en.wikipedia.org/wiki/Representational_state_transfer`
[14] Ormancey E 2008 *J. Phys.: Conf. Series* **119** 082008
[15] PHPMD - `http://phpmd.org`
[16] Dahse J and Schwenk J 2012 *RIPS-A static source code analyser for vulnerabilities in PHP scripts* `http://php-security.org/downloads/rips.pdf` Ruhr-University Bochum
[17] JSHint - `http://jshint.com`
[18] Firebug - `http://getfirebug.com`
[19] YSlow - `http://yslow.org`
[20] Selenium - `http://www.seleniumhq.org/`
[21] Jenkins - `https://jenkins-ci.org/`
[22] Brown M 1996 FastCGI: A High-Performance Gateway Interface *Fifth International World Wide Web Conference* (Paris)