Use of Interactive SVG Maps and Web Services to Monitor CMS Tracker

Giuseppe Zito
and Maria Santa Mennea
Sezione Infn Bari
via G.Amendola 173
I-70126, Bari, Italy
Email: giuseppe.zito@ba.infn.it
maria.mennea@ba.infn.it

Abstract— **The complexity of the hep detectors for LHC (CMS [1] Tracker has more than 50 millions of electronic channels), the large number of people in charge of it (more than 100 people from 10 labs distributed worldwide) and the availability of the Grid computing environment require novel ways to visualize data coming from detector for use in online monitoring.We suggest here an approach that takes advantage of the distributed computing environment.Web services will ship the data from the control room to the user desktop in the form of SVG maps. These contain the monitoring information represented in a compact way and have interactive features that allow the user to diagnose quickly detector problems.**

I. Introduction

SVG (Scalable Vector Graphics) W3C specification [3] is slowly becoming a standard way to show graphically information on the Web in 2D. Some outstanding use of it [4] to represent statistical data on geographical maps shows how sending a SVG file plus some Javascript code may allow an user with a SVG enabled browser to display and interact with the information sent in many ways.

The quality of maps rivals the paper equivalent with a interactivity impossible on paper.Real-time update of the map is also possible through Javascript. We report here on a study done to see if SVG images can be used to access and interact with CMS tracker monitoring data. The detector map [5] of the tracker is already implemented in CMS visualization software. We have already studied another solution to this problem with a Java client used to develop the GUI and with web services used to access monitoring data. In the approach described here we replace the Java client with SVG but the web services implementation is the same. For this reason we won't repeat the detailed description of the web services reported elsewhere[6] but at the end we do a comparison between these two approaches.

Web services [2] are used to develop a standard interface to all information providers used for detector monitoring and are essential when the expert uses the map to diagnose a problem.

II. Why SVG

SVG is already being used in HEP (and outside HEP) visualisation tools as a standard way to export 2D graphics. For this use it can replace the postscript format since it has the same capability to represent vector graphics (shapes, text and embedded images).If you compress the text file, it is also more compact. Svg images can be zoomed and panned exactly like postscript images. By adding Javascript code other interactive features can be added. In our case the most important are:

- The possibility to pick a part of the image getting information about the detector part represented in the detail.
- The possibility to connect the image to web services that will provide new data refreshing the image.

Since SVG is an XML instance, it should be supported by all browsers which are XML-compliant clients. However up to now only Mozilla/Firefox [8] and Opera are about to release a version with native support. For Internet Explorer you have to install the Adobe plugin [7] (this works also with Mozilla).

III. Web access to monitoring data

Monitoring the CMS tracker requires access to the following data sources:

- construction data base (this is a single database containing for example list of dead channels in the module)
- DDD (Detector Description Database) - Description of detector geometry.
- conditions data base (working conditions for each module: voltages,temperature,etc)
- last events read from detector
- events from a Grid dataset
- histograms data base

Since each data source has a different API, we define a single XML format for tracker data and instead of accessing the data directly, we use web services that act as gateways to data sources. The data is transformed in this xml format and sent back to the SVG image when the user requires new data to display. The data about the geometry are only accessed once, at the beginning of a interactive session , and produce the sending of the SVG image itself. In some cases,when the data to be sent is too much, it can be transferred directly in the SVG format (instead of XML) and replace completely the previous image instead of only refreshing it.

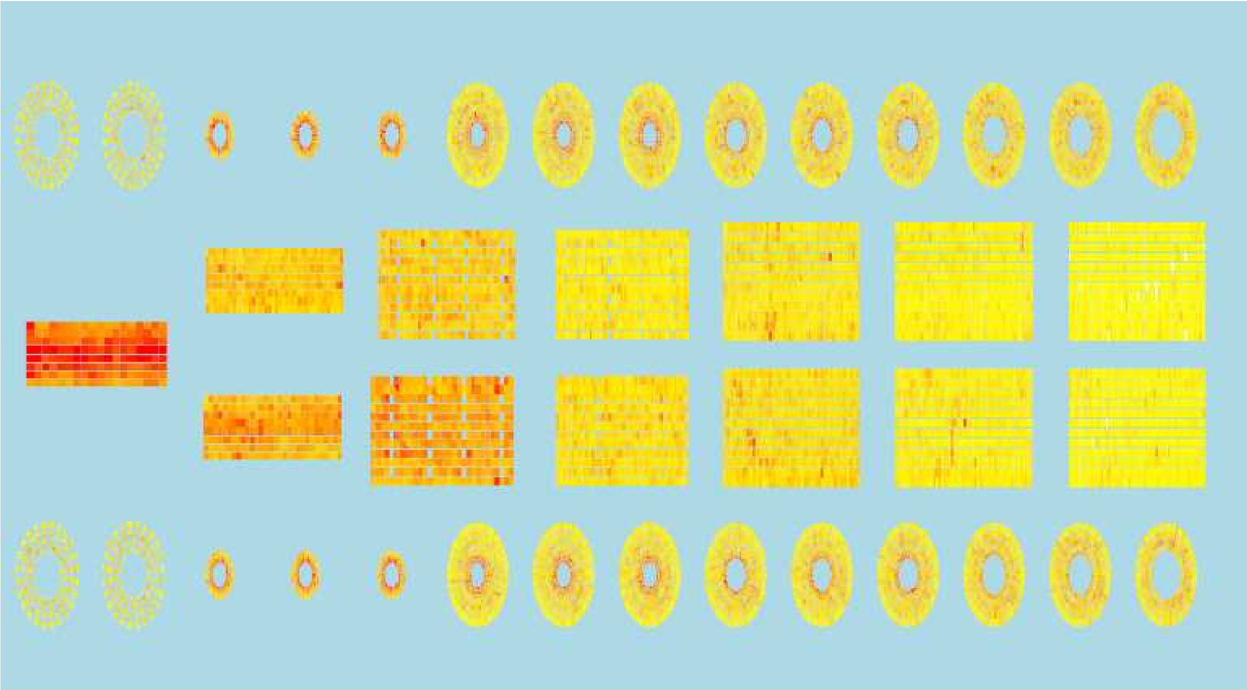

Fig. 1. Tracker Map with integrated signal from 50 events.

IV. The Tracker Map

In the past the main visualization tool for monitoring was an histogram presenter that would show a set of histograms and tables updated regularly. The CMS tracker has more than 50 milions channels organized in 16540 modules each one being a complete detector: its monitoring requires many thousands of histograms to be computed every few minutes. An histogram presenter is not enough: people in charge for monitoring need a representation that would show all modules at once in a single computer screen with single modules information coded in some way. This is the representation implemented in the SVG image. This "tracker map" [Fig. 1]is obtained in the following way: since the single module is flat , we imagine to disassemble the whole tracker and to assemble it again on a flat surface putting the single modules in positions which are connected to their spatial position. Data from each one of the data sources described in the previous paragraph can be represented on this map. For example the single module can show coded with a color:

- number of dead channels from construction database
- total number of rechits hitting the module in the last 100 events readout
- result of a comparison between the last histogram and a control histogram

Holes or hot spots in the map can pinpoint detector problems.In this case the interactive nature of the map is essential. In this use for Web monitoring, the map has the additional advantage of minimizing data transfer (of course this map can be used also locally in the control room).

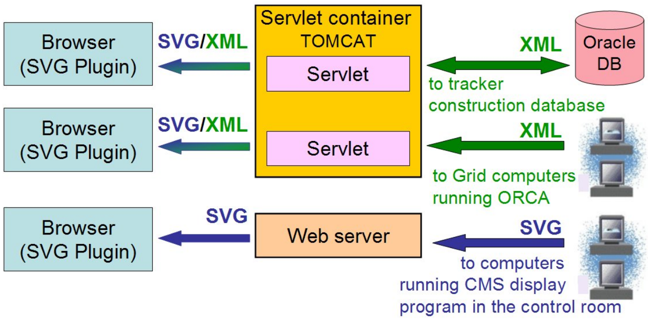

Fig. 2. System Architecture.

V. Implementation

The architecture of the system can be seen in this image [Fig. 2]. The web browser can connect to many data servers: one for each data source. To implement the data servers as web services we used Tomcat that provides web services following the W3C definition with XML and WSDL. Up to now the data servers can provide data from two different sources :

- datasets of Monte Carlo events for the full detector(on the Grid)
- data from the construction database(this is a single Oracle database)

The implementation of these Web services is described in detail elsewhere [6]. The SVG approach makes also possible another way to access data that is shown on the bottom of the figure. A program running CMS code on a computer in the control room (for example the event display) can save the data directly in SVG format on a disc area served by a Web server. In this way a computer connected to Internet can see the same image seen in the control room (thus implementing a push type Web service). But this push type service can only be useful to inform people everywhere in the world about the situation in the control room. When the expert needs to diagnose some problem web services of the pull type like those implemented in Tomcat are essential.

A. The SVG map

When the user requests data to the service, the first time he gets an SVG image of the tracker containing Javascript code. This image is built using the geometrical data of the detector and can be of two differend types:

1) Tracker map with overlayed modules:this is good if you want to see exactly how modules overlap and how the signal is positioned inside them.
2) Tracker map with separated modules: in this case each module (also in a stereo pair) is represented by a separate polyline. This is useful to visualize quantities defined for each module.

In addition to the normal interactivity of SVG images (zoom and panning) we added through Javascript functions [Fig. 3]:

- The thumbnail at bottom left makes more easy the panning since it allows the selection with the mouse of the zone to view.
- Special buttons are present to zoom in, zoom out and return to initial image.
- When you move the mouse on the image you get information about which tracker part you are.

This is the minimum interface needed by the experts to diagnose problems.

B. Real time update of the map

Further requests of data by the experts are done using Javascript. This request is done using a different API for Javascript embedded in the Adobe plugin and Javascript embedded in the browser. These tests were done using a Adobe plugin, so we used the first method, but after the release of Mozilla/Firefox with SVG native support, we will require the second method. (We have not yet implemented this capability in the interface: we have only written some temporary code to do our tests.)

In the first method we use the geturl function. If the XML file is successfully received, we use the parseXML function to parse the file attaching it to the DOM document containing the SVG map. Then we can traverse the tree updating the SVG with the new data.

In the second method the XMLHttpRequest object is used to contact the remote server. It has many methods that allow to contact the data server, to receive the data asynchronously and to extract them directly from the document's DOM tree (no need to do parsing).

VI. Performance

The test was done on a PC with a Pentium IV CPU and with both the Linux and Windows operating system. The version of the Adobe plugin to view SVG was 3.01 on Linux and 3.02 on Windows. The size of the Ascii file containing the tracker map in SVG is around 3 MB. The time necessary to have the map loaded in the browser is around 20 seconds. This time is due mostly to extract the data and build the DOM tree for the SVG image which contains around 100,000 nodes. The time necessary to transfer the data from the remote computer was negligible in our test (it depends on the speed of the Internet connection). After these 20 seconds all the local image manipulations provided by the interface (zoom , pick,etc) are fast. Also printing the image on screen is done excellently by the Adobe plugin.

Unfortunately the real time update of the map with Javascript is too slow and can be used only if you have to change a small portion of the image. The traversal of the complete DOM tree requires 5 minutes on Windows and 7 minutes on Linux! For this reason it is a lot faster to send new data directly in SVG format and update the SVG image by loading a new one which replaces completely the previous image (because this only requires 20 seconds).

At last for this kind of application it is unrealistic to send a new event in XML format and use these data to update the image.

Does this limit the use of SVG in our application? Not necessarily but you must design the SVG interface carefully in order tominimize the local processing of the DOM tree of the image . This requires the development of a special strategy where for example:

1) When one clicks on a module, a request is sent to a server for an histogram of the module and this, after the data is received, is represented in a special window below the tracker map.
2) To get information from other data servers you must click for example on a button that links the data server. The result is a new SVG image that replaces completely the actual image but has the same interface (Of course this can also be opened in another window of the browser).
3) To get more detail you reserve a special area reserved for such detailed zoom and when one requests it by clicking on a module, the data about the module is requested to a server and displayed by updating the content of this area.

VII. Conclusion

Given the fact that the tracker is now in construction and data taking will start in 2007, this was first of all a feasibility study for real time monitoring from Internet using web services and SVG maps. The results show that monitoring a detector like the CMS tracker from Internet should be feasible but with some limitations due to poor performance of Javascript in the browser. At least the following use case

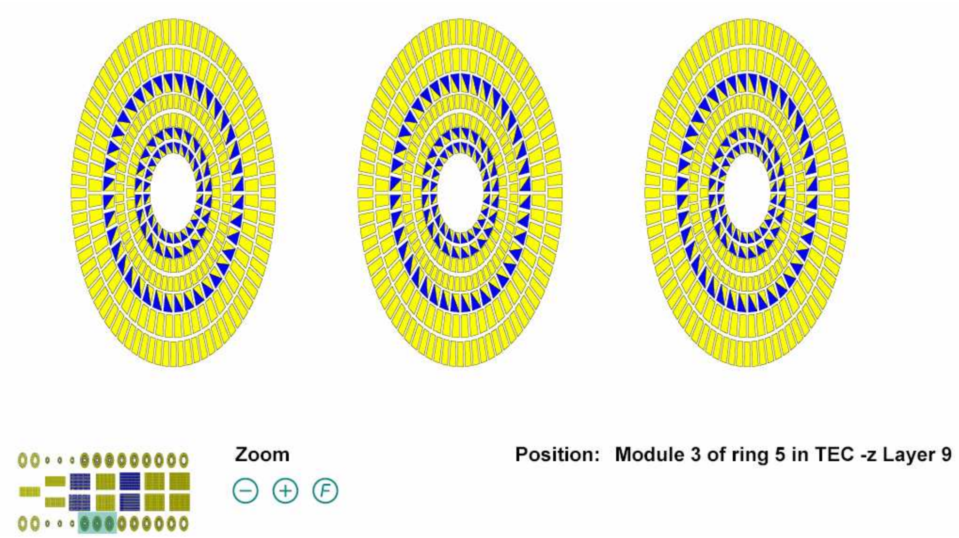

Fig. 3. SVG map interface.

TABLE I
COMPARISON BETWEEN SVG AND JAVA CLIENT APPROACH

	SVG	**Java client**
Real time update	5min	10sec
Software development	very easy (mostly copied from the web)	difficult
Requirements for the user	Plugin installation	Java runtime installation and Java client download
Graphics performance	very good	sufficient
Interactivity	sufficient	sufficient

is feasible without problems. A monitoring program working in the control room can save every few seconds on a server Web an SVG image with the integrated signal map for the last events Anywhere in the world, an expert can in less than 20 seconds with a normal browser access it and check that everything is ok with the 17000 modules of the tracker. In case of problems he knows exactly which parts of the detectors are misbehaving and he can request using the same image more informations about these modules(for example histograms).

A. Comparison between the SVG and the Java client approach

Table I shows how the two recipes compare. From the table you see that the main strength of the Java client solution is the real time update speed. At the moment there is no clear winner. But in the future better SVG support and performance increases in PC could make the SVG solution superior to the Java solution also considering how easy is its implementation.

ACKNOWLEDGMENT

The authors would like to thank Juliana Williams and Andreas Neumann for the kind permission to use their Javascript code in the website: http://www.carto.net/

REFERENCES

[1] CMS Collaboration, *The Compact Muon Solenoid, Tecnical Proposal*, CERN/LHC C 96-45, CERN 1996
[2] David Booth et al., *Web Service Architecture*, W3C Working Group Note, 11 February, 2004. Available: http://www.w3.org/TR/ws-arch/
[3] Ola Andersson, et al., *Scalable Vector Graphics (SVG) 1.1 Specification W3C Recommendation* , W3C Recommendation 14 January 2003 February, 2004. Available: http://www.w3.org/TR/ws-arch/
[4] See for example the material available on the site: http://www.carto.net/
[5] M. S. Mennea, G. Zito, A. Regano and I. Osborne, *CMS Visualisation Tools*, CHEP'04 Conference Proceedings, Interlaken Switzerland, 2004. Published on InDiCo.
[6] M. S. Mennea, G. Zito and A. Regano , *Monitoring CMS Tracker construction and data quality using a grid/web service based on a visualization tool*, IEEE NSS Conference Proceedings, Rome Italy ,October 16-22 2004
[7] http://www.adobe.com/svg/
[8] http://www.mozilla.org/