DETECTOR CONTROLS MEETS JEE ON THE WEB

F. Glege, M. Janulis, A. Andronidis, O. Chaze, C. Deldicque, M. Dobson,
A. Dupont, D. Gigi, J. Hegeman, R. Jiménez Estupiñán, L. Masetti, F. Meijers,
E. Meschi, S. Morovic, C. Nunez-Barranco-Fernandez, L. Orsini, A. Petrucci, A. Racz,
P. Roberts, H. Sakulin, C. Schwick, B. Stieger, S. Zaza, P. Zejdl (CERN, Geneva, Switzerland),
U. Behrens (DESY, Hamburg, Germany), O. Holme (ETH Zurich, Switzerland),
J. Andre, R. K. Mommsen, V. O'Dell (Fermilab, Batavia, Illinois, USA), G. Darlea,
G. Gomez-Ceballos, C. Paus, K. Sumorok, J. Veverka (MIT, Cambridge, Massachusetts, USA),
S. Erhan (UCLA, Los Angeles, California, USA),
J. Branson, S. Cittolin, A. Holzner, M. Pieri (UCSD, La Jolla, California, USA)

Abstract

Remote monitoring and controls has always been an important aspect of physics detector controls since it was available. Due to the complexity of the systems, the 24/7 running requirements and limited human resources, remote access to perform interventions is essential.
The amount of data to visualize, the required visualization types and cybersecurity standards demand a professional, complete solution.
Using the example of the integration of the CMS detector controls system into our ORACLE WebCenter infrastructure, the mechanisms and tools available for integration with controls systems shall be discussed. Authentication has been delegated to WebCenter and authorization been shared between web server and control system. Session handling exists in either system and has to be matched. Concurrent access by multiple users has to be handled. The underlying JEE infrastructure is specialized in visualization and information sharing. On the other hand, the structure of a JEE system resembles a distributed controls system. Therefore an outlook shall be given on tasks which could be covered by the web servers rather than the controls system.

INTRODUCTION

The Compact Muon Solenoid (CMS) detector is one of the four experiments of the Large Hadron Collider (LHC) at CERN. The Detector Controls System (DCS) is needed to bring the detector in a state to collect physics data. The scale of the system is illustrated with the following numbers:

- ~3 million parameters
- ~700.000 lines of code
- ~35000 finite state machine nodes
- 34 SCADA systems
- 29 redundant PC pairs (Windows)
- ~50 DB schemas (ORACLE)
- O(TB) of data in database schemas

The system is running in production since several years with a high efficiency. It is built using the SCADA system WinCC OA from the Austrian company ETM. At CERN the Joint Controls Project (JCOP) created a toolkit based on WinCC OA to facilitate the development of the experiment controls systems. Today this is used by all LHC experiments and for several other projects at CERN.

PART1: REMOTE CONTROL VIA WEB

Visualization in WinCC OA

In WinCC OA visualization is done through so-called panels. Using a graphical editor, graphics objects (buttons, text, lines, etc.) can be placed on a window. Each element can be enhanced by scripts, which allow to change the properties of the element itself. The scripts have access to the process data and can therefore reflect the process information through the visualization. The graphical editor produces a file describing the panel layout and functions. This file is then used by a WinCC OA process to visualize these panels. This process uses Qt as a graphics library, with all graphics elements being drawn using the Qt graphics API. The Qt Library itself translates the drawing of the panel into primitive paint commands which are sent to the device used for visualization.

Remote Control

Around 4000 members of the CMS collaboration should be able to access the DCS information remotely in a read only manner. The data should ideally be live to ensure a good user experience. Experts should in addition be able to interact and remotely control their systems. Authorization and authentication are required to enable interaction.
The means for remote control currently used in CMS are

- snapshots taken regularly of selected panels being served as images vie the web server
- HTML/java script pages providing live information collected
  - from the logging database
  - through protocols supported and used by the controls system

A full discussion of the current remote control solution in CMS DCS can be found in [1].

Live Web Access

Thousands of panels have been created for the detector controls of the LHC experiments. Reimplementing all of them at once to provide a pure HTML solution which

ISBN 978-3-95450-148-9

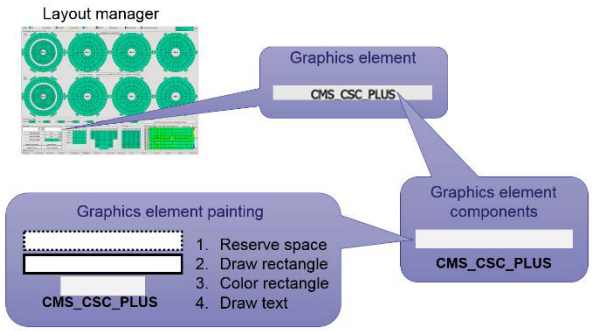

Figure 1: The graphics stack.

would offer live data access using standard web tools is not an option. Giving access to WinCC OA panels directly through remote access tools like remote desktop, VNC, etc. would require too many computing resources caused by the large number of possible clients.

A good compromise represents the usage of the Qt platform abstraction mechanism [2]. Graphics in applications are usually created on several layers (see Fig. 1). On the highest layer a layout manager will position graphics objects according to the rules the designer has set. Each graphics object is composed of graphics primitives. Those primitives are drawn using basic commands to reflect its geometry.

The Qt platform abstraction mechanism allows to intercept the basic drawing commands (see Fig. 2). It happens that those commands are often similar for different visualization devices. Also the commands used in an HTML5 [3] canvas are similar to those used in Qt. The solution is to intercept those commands in Qt and send them to a web browser, where they are applied to an HTML5 canvas. Doing so creates an exact image of the Qt graphics inside the browser. Sending all events of the HTML5 canvas (mouse movement, clicks, button press, etc.) back to Qt allows to have exactly the same control remotely as one would have on the local Qt graphics display.

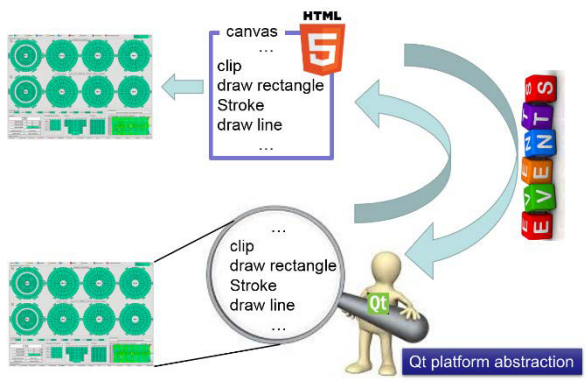

Figure 2: Paint command redirection.

Web Server as a Gateway

To make the above mentioned mechanism usable for many clients in a real environment, a web server offering JEE (JAVA Enterprise Edition)[4] has been added as a gateway between Qt and the browser. Doing so provides

- authentication and authorization based on standards
- multi user access
- firewalling using well tested, widely used software
- business logic implementation and process modelling

A resource broker is required to handle the graphics sources (in our case WinCC OA user interface managers) and steer the data exchange. On request the resource broker starts a new user interface manager, if the requested panel is not yet served by another manager, and tells the Web socket proxy to set up the connection (see Fig. 3). In response to the initial browser request an HTML client containing the HTML5 canvas and all necessary Java Script code will be returned. This client will open a web socket to the web socket proxy which will then open the connection to the user interface manager and start proxying data. The web server infrastructure will ensure that the client is authenticated and authorized to do this action.

Two modes are possible using this mechanism:

- One client per panel, allowing for read write access to the panel. This is to be used by system experts, who need to interact with their system
- Many clients per panel, allowing only read access. This is to be used by any authenticated user to gather the current system status.

In our case the resource broker and the web socket proxy have been implemented as a proof of concept and run in an ORACLE portal infrastructure.

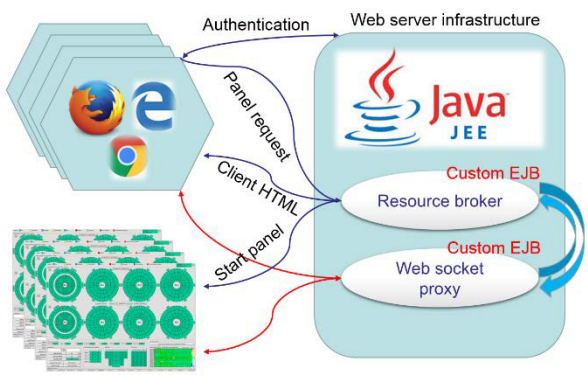

Figure 3: Work flow.

Why JEE?

Our web server infrastructure provides JEE. The JEE concept contains EJBs (enterprise JAVA beans) which have several features that ease the implementation of the described remote access mechanism:

- EJBs model the business logic of a JEE application
- EJBs are not limited to a request
- EJBs provide intercommunication
- EJBs allow modularity and scalability
- EJBs allow for distributed systems

ISBN 978-3-95450-148-9

By implementing a resource broker and a web socket proxy as an EJB, they profit from the inherited infrastructure. The reception of the initial browser request is very simple in an EJB as is the communication amongst different EJBs. These components can be distributed over several web server instances (e.g. to do load balancing) using the same communication mechanism.
Thus the use of JEE significantly eased the development for CMS and allows building complex system architectures out of the box to fulfil functional requirements.

Performance

The first implementation simply expressed all paint commands intercepted in Qt as HTML5 canvas paint commands and sent them to the browser. The bandwidth required to show a reference panel was ~1.1MB/s. This did not allow for a good user experience since drawing on the canvas was lagging behind and even crashing the browser. Several optimizations have been applied to improve this:

- Command indexing:
  - The paint commands have been replaced by a numerical representation.
- Command sequence caching:
  - All commands are aggregated in a continuous buffer. If the server identifies a command or sequence of commands which has already been sent, it just sends a pointer and the length of the sequence to be repeated.
- Image caching.
- Optimized TCP packet usage:
  - Commands are sent in chunks to minimize the network protocol overhead.
- Update frequency Reduced to human perceivable rate
- Optimized (reduced) event sending:
  - A mouse movement on the client side will create a move event for every pixel it moves. Small movements are collected, combined and sent as a single movement.

After applying the previously mentioned optimizations, the required band width could be reduced to ~20kB/s. This provides a sufficiently good user experience where the remote graphics are nearly in sync with the local one.

Next Step

Many panels provide navigation mechanisms: either they contain tabs or provide access to other panels. Ideally navigation should be available to all users and not only to the expert, with rights to interact, since the full information is only obtained through this mechanism. In the current mechanism it is not possible to implement this since the navigation items (tabs, buttons, etc.) are not identifiable on the client side. (The HTML5 canvas only knows about an accumulation of geometrical objects)
Furhermore, accepting events on navigation items from all users would mean that the resulting changes in the panel would also be shown to all users.
To overcome this problem and allow for basic navigation, the graphics elements used for navigation could be replaced by HTML objects. Now the browser would identify actions on those elements and could request the underlying action to the server. In case of tabs, the resource proxy would redirect the connection of this browser to a panel showing the requested tab.
Since the navigation elements will no longer be transferred as graphics primitives but as high level HTML code, the required bandwidth is reduced and performance increased. Using object based authorization the access to those elements could be restricted on a per user basis.

Final Goal

The final goal is to replace all panels with pure HTML panels. The advantages would be maximum performance through minimal required bandwidth and multi user read/write access. As mentioned before this will require a sizable development effort and is therefore not achievable in short term.

Summary Part1

We implemented a solution for web based remote control mirroring WinCC OA panels using Qt platform abstraction. No modification of panels is needed to provide single user read/write or multi user read only with sufficient performance. An evolution towards a fully HTML based remote visualization is possible and foreseen.

PART2: JEE GRAPHICS AND BEYOND?

The simplified architecture of a controls system could be sketched as shown in Fig 4.

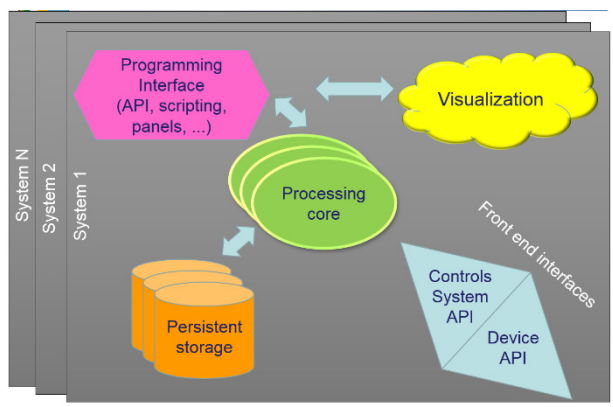

Figure 4: Simple sketch of a controls system architecture.

Apart from the programming interface, all parts are logically independent. The separation of processing core and programming interface might not be as clear as shown and depends on the implementation. The connection to the devices is often device specific. Interface concepts like OPC UA and TCP based bus systems should generalize the front end connections and remove the need for specific hardware interfaces.
After some reshuffling and abstraction the JAVA EE architecture looks like in Fig. 5.

ISBN 978-3-95450-148-9

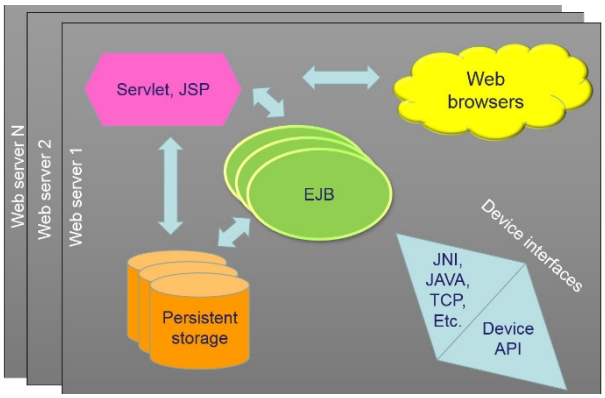

Figure 5: JEE architecture.

It's obvious that a web server can be used for visualization. (The first part of this paper proofs this). The persistent storage is the same in both cases. Through the Java native interface (JNI) any device library can be connected to JAVA. According to their definition the EJBs are meant to be used to model business logic. That is what the processing cores in any controls system do. Also here the borderline between EJBs, servlets and JSP is not a clear separation. Some tasks might be implemented with either type. The general concept therefore looks the same.

JEE for Controls?

Seeing that the concept fits, could one use JEE to build a control system?

A web browser as visualization is certainly very powerful and provides a large amount of freely available components. This is certainly more than most SCADA (Supervisory controls and data acquisition) systems provide.

The access to persistent storage systems is part of JEE. Usually different types of storage are supported by the web server infrastructure. JEE even provides a possibility to declare data items to be made persistent, which is done transparently to the user. The achievable performance has to be tested but the modular nature of the JEE concept should allow to scale up sufficiently to reach the required performance.

The connection to front ends is possible by design, since any library can be connected to JAVA through JNI.

The processing core in JEE, i.e. JSPs (JAVA server pages), servlets and EJBs provide inbuilt local and distributed communication means allowing for a distributed design. Also here a certain performance has to be reached but since JAVA is a standard programming language, this is not more or less difficult than with any other language. The ease of factorization, however, is an advantage over bare C++ for instance.

Summary Part 2

JEE is not a readily built SCADA software. JEE is a concept for which different implementations exist and provides many components and interfaces required to build a controls system. It is well documented, widely used and often free of charge.

For those reasons, JEE seems well suited as a basis for the development of a new controls system.

REFERENCES

[1] Lorenzo Masetti et al., a scalable and homogenous web-based solution for presenting CMS control system data, THCOAAB01, proceedings ICALEPCS 2013.
[2] http://wiki.qt.io/Qt_Platform_Abstraction
[3] http://www.w3.org/TR/html5
[4] http://www.oracle.com/technetwork/java/javaee/overview/index.html

ISBN 978-3-95450-148-9