UNIVERSITY OF WATERLOO
Faculty of Science

WEB APPLICATIONS FOR ACCELERATOR PHYSICS FACILITIES

TRIUMF
Vancouver, BC

Prepared by
Paul Matthew Jung
ID #20463518
pmjung@uwaterloo.ca
3B Mathematical Physics

January 15, 2016

Paul Matthew Jung
54 Bradley Dr.
Georgetown, ON L7G 6B4

January 15, 2016

Jeff Chen, Chair
Physics
200 University Ave W
Waterloo, ON N2L 3G1

Dear Jeff Chen:

This report entitled "Web Applications for Accelerator Physics Facilities" was prepared for TRIUMF as my 3B work term report. This is my third work term report in my undergraduate career.

TRIUMF is Canada's national nuclear and particle physics laboratory. The Beam Physics group in which I was employed is responsible for the design, study, and operation of the accelerators. The group is lead by Rick Baartman, my supervisor within the group was Suresh Saminathan.

The purpose of this report is to provide an overview of the Beam Visualization application I developed over the term and to evaluate the prospect of developing web applications for further use at TRIUMF, or other high energy facilities.

This report was written entirely by me and has not received any previous academic credit at this or any other institution. I received no other assistance.

Yours sincerely,

_____

Paul Matthew Jung, 20463518

Table of Contents

Summary . . . . . **iii**

1 Introduction **1**

2 Development **2**
2.1 Client Side 3
2.2 Server Side 6

3 Modules **9**
3.1 Tune Display 9
3.2 Beam Envelopes 9

4 Conclusions **13**

5 Recommendations **14**

1 References **15**

List of Figures

1 Client State Machine Diagram 5
2 Client Module State Machine Diagram 6
3 Server State Machine Diagram 8
4 Perl TK Beam Envelopes Calculator Application 11
5 Beam Envelopes Web Application Module 12

Summary

This report analyses the potential use of web applications in high energy physics facilities. The merits of web applications are contrasted against traditional desktop applications.

It is discovered that web applications, have limitations making them most suitable for simple programs, examples being diagnostics, graphing, monitoring, and analytic programs.

The development of web applications is more involved than traditional desktop applications however, they offer many advantages including accessibility, compatibility, ease of distribution, and version control. Web applications are discouraged for use in complex programs, as expansive testing and debugging is difficult in web development.

The development of Beam Visualization, a modular web application is analysed. The libraries and development philosophies used in the application are described. The initial release included a web application implementation of a Beam Envelopes desktop application that uses the beam transport code Transoptr for simulations. It is a suitable replacement for the desktop application as it has more functionality and is more expandable.

1 Introduction

At TRIUMF the official applications used to monitor and control the Isotope Separation and Acceleration (ISAC), and Electron Linear Accelerator (ELINAC) facilities are traditional desktop applications built using the Experimental Physics and Industrial Control System (EPICS) software suite. The open source EPICS project provides libraries for many programming and scripting languages so that users can create their own software. However, desktop applications are limited when it comes to platform dependence, version control, and distribution.

In order to maximize the amount of time beam is reliably delivered to the experiments Accelerator Physicists and Operators may need to access specific programs at any time, from any location, during beam delivery. The occasional erratic problem during operation may require the use of a specific individual's knowledge, tools, and talent, regardless of their location. The reliability of beam delivery would then depend on their ability to access information, use their tools, and communicate their results with others.

In addition, the Accelerator Physicists being able to easily share programs, especially during development, and to make accessible to others data sets and results stimulates collaborative work specifically for research and development.

The current methods used to work remotely include using a virtual private network (VPN) with a secure shell (SSH) to work in a command line environment or a remote desktop application.

2 Development

The modern solution to this problem is to develop web applications, fully featured applications which are run in a web browser and accessed through a network. This project, titled Beam Visualization, is a single page web application for displaying information about TRIUMF's particle accelerator facilities. This is so that the tools it implements can be accessed from any internet enabled device. Modern web browsers are backwards compatible so that any page developed now, will remain accessible in the future. Also, unlike a desktop application, no software packaging, distribution and update distribution is necessary since the user gets the latest version of the application when they load the page. Lastly, all of the data, configuration files and code base can remain in one location, so it is easy to manage, especially with the aid of a version control system. Developing for the internet can be complicated. However, the World Wide Web Consortium, the international web standards organization codifies standards and publishes tutorials. Also, there is a world wide community of web developers, which together form a complete knowledge base because internet is so ubiquitous.

Nevertheless, a web application requires dedicated server hardware and there may be resulting performance restrictions based on the hardware chosen. Web based applications may also face performance variations based on the variety of the hardware that clients may be accessing the page on; a mobile phone will perform differently than a desktop computer. Web browsers are also not optimized for intensive graphics and will consistently perform worse than a desktop application that is properly optimized, in high performance situations. For example, a three dimensional animation will easily run on a desktop with graphics processor acceleration but may not run as well in a

browser on the same computer. Also, developing for the internet is difficult because some features of the standards are not guaranteed to be supported by all web browsers. Lastly, it is either time consuming or expensive for a developer to debug their web application on all of the platform combinations it may be used on.

The recent proliferation of web applications may be due to the technology that enables web developers to create dynamic web pages. It is a collection of technologies and standards referred to as Asynchronous JavaScript and XML (AJAX). This standard lets JavaScript make asynchronous requests to the server to retrieve information, usually used to update a part of a web page, without navigating away. This is commonly used to create pages with near real time information but can be used in more general applications as well.

An important requirement of this project was to use a modular design. This is because there are separate versions of the application, which may look different and implement different modules depending on the section of beam line it looks at and the preferences of the users and developers of that particular section. The core module that handles the application state and universal communication should be common for all versions of the application. The caveat being that the modules themselves should be independent of the version of the page it is in.

2.1 Client Side

The client side of the application is a standard web page. It consists of a main hypertext mark-up language (HTML) file which includes links to a cascading style sheet (CSS) file for modifying the look of the page as well as JavaScript files. The joint use of these technologies is referred to as dynamic hypertext

mark-up language (DHTML). The associated JavaScript files control all of the logic and most of the data of the application; the rest being in the document object model (DOM) of the web page. JavaScript is a high-level interpreted programming language. It is unique since the interpreter is built into all modern web browsers, which execute the code while simultaneously displaying the page. Since the internet contains many competing approaches to development it is reasonable for a web development project to take longer than an equivalent desktop application, for a single platform.

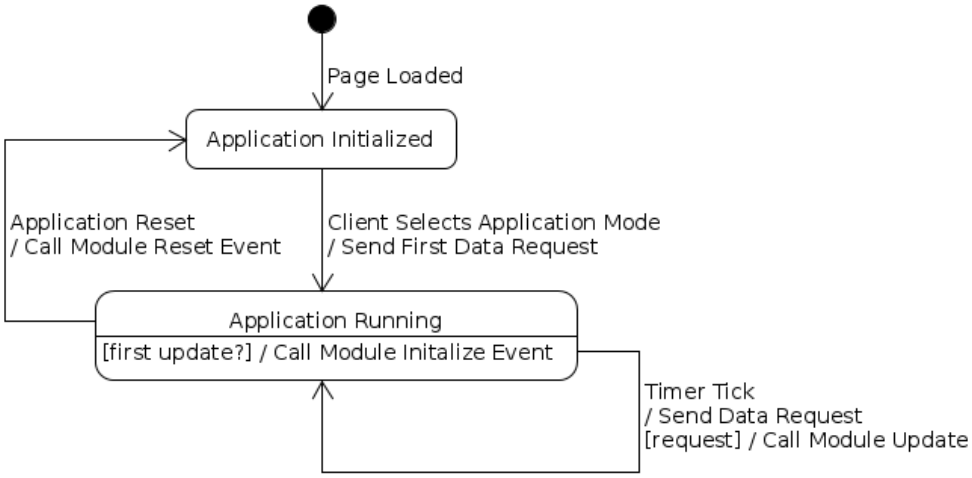
Libraries including JQuery and JQuery UI were used to expedite development. JQuery acts as a layer of abstraction for many JavaScript functions and features, that would otherwise be web browser dependant. This solved many of the compatibility concerns involved in developing JavaScript code. JQuery UI was used to quickly implement advanced user interface features not natively supported in the HTML and CSS specification, an interactive overlay menu, for example. Lastly, the JavaScript library HighCharts was used in the modules to create dynamic, interactive, graphs.

JavaScript as a language imposes no structure for organizing code or separating it within a file, a convenient feature for short scripts contained in a script tag in a HTML document however, this can create problems for large programs depending on many separate JavaScript files. One major problem is variable name collision, it occurs when reusing variable names and can cause erratic behaviour even across script files or between scripts and the interpreter itself. Hence, a self imposed structure is needed to approach the development of large web programs. There are many commonly used formats for this. The code structure that was utilized for this project is the Revealing Module Pattern [1]. This code structure enables the creation of objects which give the programmer most of the features of object oriented programming, the excep-

tion being inheritance. Objects can be created with both private and public variables and functions, and are globally accessible. The revealing module pattern works well with the modular design of the application.

The design choices for the JavaScript lead to having one core script file for the main application, common to all versions. This script controls the state of the application, houses the main timer so that the application updates at a regular interval. Each update retrieves common data that may be used by all modules. The script also triggers JavaScript events, using JQuery, to provide the current state of the application to the modules. See Figure 1 for a state machine representation of the core client application script.

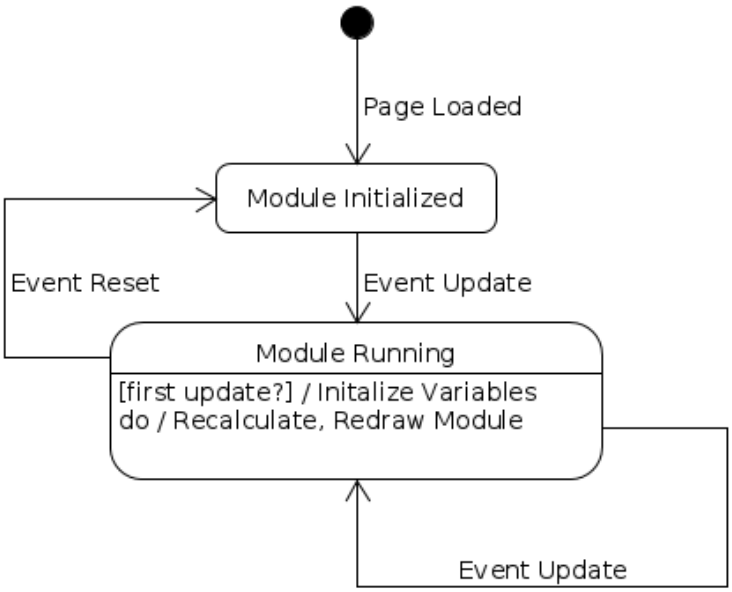
Figure 1: Client State Machine Diagram



The format of the modules are excerpts of HTML body code that may include JavaScript, CSS or any other links, specific to that module, as well as HTML code. See Section 2.2 for how the HTML excerpts are assembled into the full application page. The typical JavaScript for a module is much like the core JavaScript file as it follows the same design pattern. The difference is, the modules listen for the events from the core module, and use the events to

change state and get information, although, it is not mandatory. An example of a state machine for a simple module that uses information solely from the core script, is Figure 2. Otherwise, a module can be arbitrarily complex, it may contain it's own timers and communicate with other modules though events, get user input, make its own AJAX requests or it may just display a static image. The essence of the module is that it is a part of the web application that may be reused in other versions of the page, to eliminate code redundancy. Examples of modules can be found in Section 3.1 and Section 3.2 .

Figure 2: Client Module State Machine Diagram



2.2 Server Side

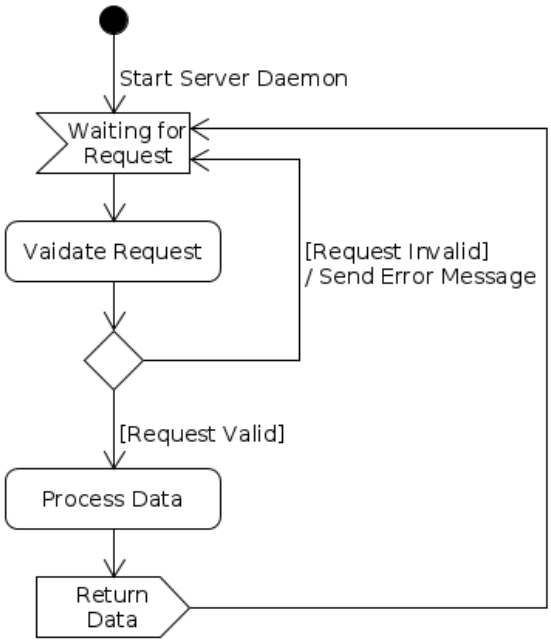
Development of the server part of the application is much more straightforward than the client part. The server's role is to return the web page when a web browser requests it, as well as return data when an AJAX call from a running instance of the web application requests it. However, there are many server

architectures that fulfill this role. This project utilized the representational state transfer (REST) set of constraints [2]. The the most notable constraints applicable to this project include: the client server model of data transfer and statelessness of the server. The client server model dictates that only the client can initiate communication (make requests) to the server which can then, and only then, respond. The statelessness of the server mandates that the server stores no contextual information about the client, instead the client must provide context with each request. This architecture provided many advantages including quicker development time, as opposed to a server with state information, and simplicity.

A generalized server script state machine diagram is given in Figure 3. The constraints that make this application RESTful imply that all server scripts for this application must follow this straightforward behaviour. Implementing a script which models this behaviour can be done in almost any programming language. There are many programming languages which officially support server specific programming, however TRIUMF widely utilises the scripting language Perl, which has an active developer community and a long history of being used for web server applications.

The server code stack consists of Perl common gateway interface (CGI) scripts for returning both the web page and any module data. The main Perl script, when opened in a web browser, returns the web page. It dynamically inserts HTML of the selected modules into the page before returning it to the client. The separate Perl script that handles the main application's data transfer accepts CGI parameters from the client which specify the client's state, in this case what section of the beam line the client has selected as well as what mass and energy it is operating at. This script then returns an XML file which contains static information about the beam line, specifically, what

Figure 3: Server State Machine Diagram



hardware is installed, how is it referenced by the control system, as well as dynamic information, in this case the values the control system is reading from the hardware. This communication is initiated by the core JavaScript module, as described in Section 2.1.

The server side script has no limitations on what can be done. An example being running simulations using input data from the client and returning the resulting graph data, as in the Beam Envelopes module in Section 3.2. The Perl language is a useful choice for this feature since it supports executing any shell command in GNU/Linux and has the ability to process the output.

3 Modules

During development two modules were created to illustrate the capability of the web based application framework. These modules were not intended to be independent tools, they are however, implementations of tools already being used at TRIUMF.

3.1 Tune Display

Tune Display is a recreation of a web application of the same name originally created by Olivier Shelbaya an accelerator operator at TRIUMF. It uses the HighCharts JavaScript graphing library [3] to display a graph of the percentage deviation of each optical element along the beam line from the theoretical value. This is the simple case of a module that relies on the core module for both events and data. The module's JavaScript file works exactly like the example client module state machine as depicted in Figure 2.

3.2 Beam Envelopes

The Beam Envelopes module is a graphical user interface for the second order beam transport code called Transoptr [4]. This Fortran program is used at TRIUMF for calculating the size of the beam for beam line design and monitoring. Previoulsy, a traditional desktop user interface was developed by Chris Gong at TRIUMF usng the Perl language and the Tk graphical user interface toolkit [5]. A picture of the desktop application user interface can be seen in Figure 4. It is limited with respect to graphing options and interactivity. This module was created to replace the Perl and Tk desktop application, a picture of the web module can be seen in Figure 5. It includes all of the capabilities

of the desktop application, however, it has slightly better performance for optimization problems since it is running on dedicated server hardware. Also, unlike the Perl and Tk user interface, the graph is interactive, for example, the axes are automatically scaled, the series can be hidden or shown by the user, the mouse pointer hovering over a data point will show the values of that data point, and the mouse pointer can be used to select a section that the graph will zoom in to. The state of the application can be saved to a file on a computer locally and then the state can be restored by loading it from the file at a later date. This module illustrates the potential of web applications.

Figure 4: Perl TK Beam Envelopes Calculator Application

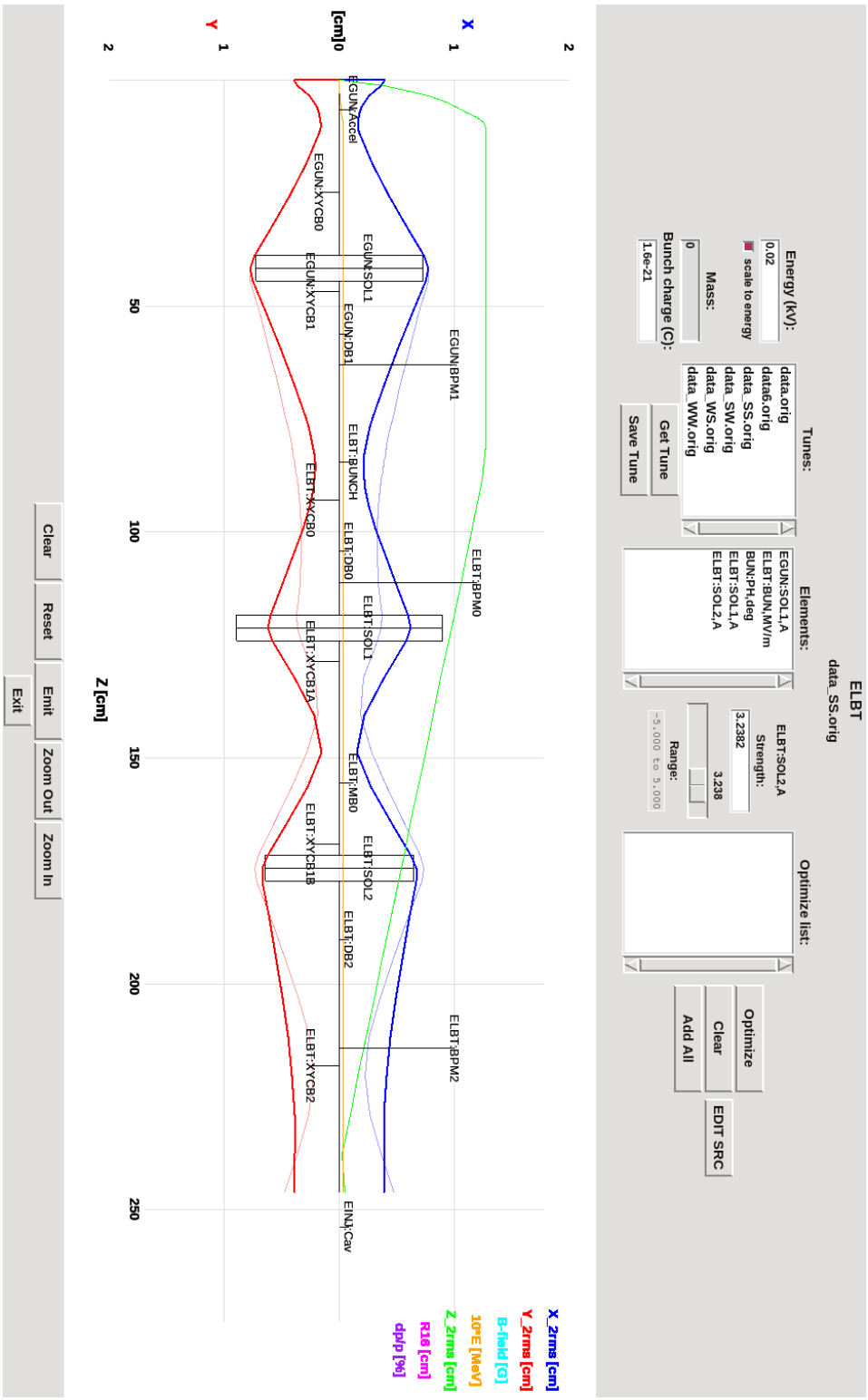
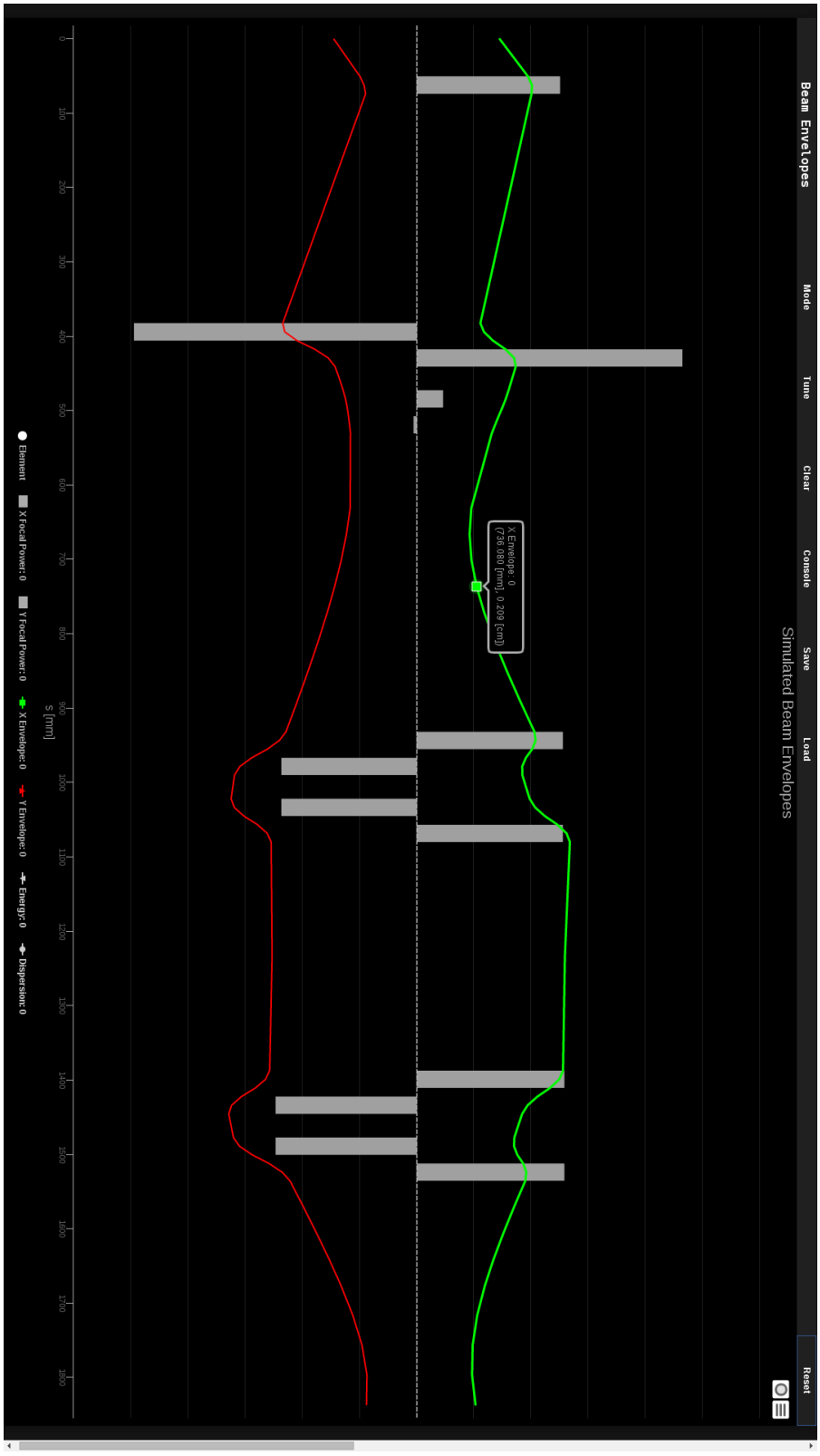


Figure 5: Beam Envelopes Web Application Module



4 Conclusions

Web applications can be very useful in a particle accelerator facility, as opposed to other methods of working remotely.

Web development is generally more difficult than traditional application development and thus is not worth the development effort for complex or high performance applications.

Web applications are most suited towards diagnostics, analytic, and monitoring tools.

Development of web applications is most efficient when using commercially available or open source libraries like JQuery and Highcharts, provided they are extensively supported and actively being developed.

5 Recommendations

TRIUMF should continue using traditional desktop applications for important tasks, specifically in the control room.

TRIUMF should continue to create modules and refine the Beam Visualization application for both external and control room use.

High energy laboratories should consider creating web interfaces that compliment their control room software.

1 References

[1] A. Osmani, *Learning JavaScript Design Patterns*, vol. 1.6.2. O'Reilly Media, 2015.

[2] R. Fielding, *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of California, Irvine, 2000.

[3] "Interactive javascript charts for your webpage — highcharts." http://www.highcharts.com/. Accessed: 2016-01-13.

[4] E. Heighway and R. Hutcheon, "Transoptr a second order beam transport design code with optimization and constraints," *Nuclear Instruments and Methods in Physics Research*, vol. 187, no. 1, pp. 89 – 95, 1981.

[5] "Tcl developer site." https://www.tcl.tk/. Accessed: 2016-01-13.

[6] J. J. Garrett *et al.*, "Ajax: A new approach to web applications," 2005.

[7] "The perl programming language." https://www.perl.org/. Accessed: 2016-01-13.

[8] "Jquery." https://jquery.com/. Accessed: 2016-01-13.

[9] "Jquery ui." https://jqueryui.com/. Accessed: 2016-01-13.