A GRAPHICAL TOOL FOR VIEWING AND INTERACTING WITH A CONTROL SYSTEM

J. Forsberg, V. Hardion, D. Spruce
MAX IV Laboratory, Sweden

Abstract

We present a graphical interface for displaying status information and enabling user interaction with the Tango based control system for the MAX IV synchrotron. It focuses on bringing an intuitive view of the whole system, so that operators can quickly access the controls for any hardware based on its physical location.

The view is structured into different layers that can be selectively shown, and various live updated information can be displayed in the form of e.g. color or text. Panning and zooming is supported, as well as invoking commands. The interface is defined by an SVG (Scalable Vector Graphics) drawing which can be edited without programming expertise.

Since our system is based on modern web technologies, it can be run as a web service accessible by standard browsers, but it can also be integrated in GUI applications.

INTRODUCTION

The MAX IV laboratory is a synchrotron facility under construction in Lund, Sweden. It has a TANGO [1] based control system consisting of thousands of different pieces of software that provide users with access to equipment. The control system tends to be structured in a way that makes sense to the software developers that build it, but not necessarily to accellerator physicists and synchrotron operators. This project, called "SVG synoptic", is an attempt at making a control system interface that is intuitive to the users.

The basic idea is to display live control system information in the form of a picture that somehow represents the physical layout of the machine, in a schematic way. Items in the picture may contain live information about corresponding control system variables, as e.g. color or text. The user is able to iteract with the drawing in order to bring up specific controls in a window, or to perform specific actions.

It may not be realistic to fit every detail of a complex system into a single image. By allowing the synoptic to contain several different detail levels, we can provide a seamless interface where the user can choose what to see simply by zooming and panning using the mouse.

A control system is usually logically split into several separate sub-systems such as vacuum, cooling, and so on, and for many use cases only some of these may be of interest to the user. SVG synoptic provides a way to structure the information into layers that can be toggled on or off depending on what subsystems are relevant at the time.

The size of the control system has to be taken into account from a performance point of view. Communicating with thousands of items is time consuming, and since most of the information is not visible at any given time it is not helpful. Therefore, SVG synoptic has the ability to only keep updated about the parts currently visible to the user.

CHOICE OF TECHNOLOGY

The development of this project was initially motivated by requirements from the users that were not easily achievable using available tools. The choice was between extending an existing system or creating a new one.

SVG

The standard tool for displaying synoptics in the TANGO community is based on a specialized Java based drawing tool called JDraw (part of the ATK [2] suite of tools). It has some of the features of SVG-synoptic, but the current viewers can only provide a static view. The JDraw format, while simple, is also fairly limited. Since it is not a widely used standard, there is not much tooling to work with the file format.

SVG (Scalable Vector Graphics) [3] has the following characteristics:

- Being a vector format (like JDraw) it produces resolution independent images suitable for free zooming.
- It is a mature, widely used and open standard, adopted by web browsers (currently all major browsers except IE 8 or older) and many graphics software packages.
- The file format is XML and therefore straightforward to generate and manipulate programmatically.
- SVG can easily be accessed from javascript since it becomes part of the DOM (Document Object Model) once loaded. It can also be styled through CSS (Cascading Style Sheets).
- The FOSS (Free and Open Source Software) drawing application Inkscape [4] uses SVG as a native format.

Considering these features, we chose to go with SVG instead of extending JDraw.

JavaScript

Part of the reasons SVG was chosen as an image format was that it enables the use of a web browser as viewer, even if embedded in an application. This cut down development time, but it did mean that the user interaction had to be implemented in JavaScript [5], the "native" programming language supported by web browsers.

JavaScript has a large community around it, resulting in a lot of high quality libraries. In particular, for interacting

ISBN 978-3-95450-148-9

with SVG, D3.js [6] is very powerful. Although mostly used for generating data visualisations, it can also be used to manipulate an existing SVG. It is used for implementing the interactive features of the synoptic.

An added benefit of having most of the application running inside a web browser is that it is straightforward to implement it as a web service.

Python

The Python [7] programming language was an obvious choice for the "back-end" part of the application, since it is already the main language used for control system work at MAX IV. Through the PyTango [8] bindings it can access the control system at the low level, and the Taurus library provides Qt4 [9] widgets and other functionality on top.

Python also has good representation in web frameworks, which means that it's also a reasonable choice considering future developments.

ARCHITECTURE

The result of this project is a library, "svgsynoptic", that can be used to easily build visualisations as described above. The implementation is as a Qt widget communicating with the underlying control system and embedding a WebKit component to render the view.

The QWebView widget in Qt makes it possible to embed a full WebKit browser environment in a Qt widget, and the PyQt bindings allow intercommunication between Python and JavaScript.

In the following section, the general architecture of the library is laid out. A few concepts will be used in this description:

model Items in the drawing must be connected to control system entities. For this we use *models*. A model is simply a string, representing some control system aspect in a standard way. For TANGO, a model may currently be the name of a device (e.g. `"sys/tg_test/1"`), or a device attribute (e.g. `"sys/tg_test/1/ampli"`). It's likely to be more generalised in the future. A drawing item may have several models.

section A section is an item in the SVG that is not necessarily directly represented in the control system, such as a logical part of a machine. The idea is to allow the user to navigate the synoptic by e.g. clicking a section in order to move the view to that part.

layer The drawing may be structured into different **layers**, each representing some global grouping of control system items, e.g. "Vacuum" or "Magnet". The user is able to toggle each of these layers on or off at any time, in order to restrict the visible information to what's relevant at the moment.

zoom level If needed, the drawing may be further structured into an arbitrary number of **zoom levels** which successively present more detailed views of the system. The synoptic will automatically switch between these levels as the user changes the scale, displaying only the current one.

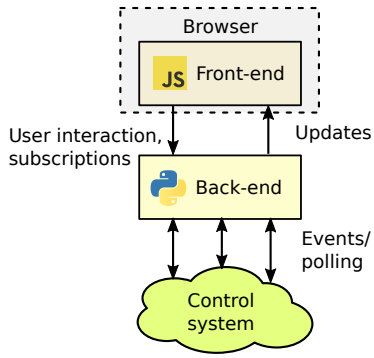

Figure 1: Architecture overview.

In order to separate concerns, the system is composed of two major pieces (Fig. 1):

- a "back-end" that communicates directly with the control system
- a "front-end" which presents a user interface to the user.

We will be discussing TANGO specific details as an example. The current implementation runs both of these parts in the same process.

Back-end

The back-end handles all communication with the control system. The synoptic itself is only concerned with presenting information about the underlying system, relying on opening panels or external applications for direct user interaction such as writing attributes or running commands. Therefore, the information flow is essentially one-way, from control system to back-end.

The back-end receives a list of models currently visible in the synoptic from the front-end (see below). This list is intended to be used to keep the front-end updated about changes in the control system. The TANGO back-end dynamically sets up change event subscriptions to these models, pushing updates to the front-end. It also unsubscribes to attributes no longer in view.

The back-end received notification of user interactions with the front-end, e.g. mouse clicks and hover. It can take appropriate actions, such as visually selecting a clicked item, opening an application when right-clicking certain models, or updating the tooltip with relevant information about the item hovered over, using the front-end's API if needed. The TANGO implementation reacts to right mouse button clicks on models corresponding to devices, by bringing up an appropriate window for direct interaction.

For the web-based implementation, a small part of the back-end will run in the browser, to handle communication across HTTP, using SSE (server-sent events) to push updates.

ISBN 978-3-95450-148-9

Front-end

The front-end (Fig. 2) is written in JavaScript and runs entirely in a web browser, usually embedded as a Qt widget. It is concerned with loading the SVG and presenting it to the user in an interactive way, reporting control system related actions to the back-end. The front-end is not directly aware of the control system and can only associate model names with items. It relies on the backend knowing what to do with the models. It has a simple API which can be used from the back-end, e.g. to inform the user about changes in the control system state of a particular model.

The front-end needs to be configured by the developer. At least an SVG image must be supplied, using the correct structure.

The UI can be extended with various optional "plugins" that add functionality. There are currently plugins for adding information "popups", a "thumbnail" view, and buttons to toggle the visibility of each layer.

Embedding

Because of its implementation as a Qt widget, it is easy to use the synoptic as part of another application. For example, it has been used as part of a specialised vacuum application (Fig. 3).

CURRENT STATUS

Applications based on the library is in use at MAX IV since around one year, for the linear accellerator, the 3 GeV storage ring and several beamlines. It has gone through several revisions in order to handle the demands of a growing control system.

Currently the storage ring synoptic contains roughly 2000 magnets, 1000 sensors, 200 beam position monitors, 85 ion pumps and 80 vacuum valves. The SVG for this synoptic is too large and complex to be drawn manually, so it is created by a script taking equipment lists as input.

The current focus of development is making the library a stable and flexible tool. The project source code is available at `https://github.com/maxiv-kitscontrols/lib-maxiv-svgsynoptic`

CONCLUSION

We have provided a library that allows creating interactive visualisations of a control system by drawing an SVG file and inserting references to control system entities. The system has been successfully used during the commissioning of the MAX IV synchrotron facility and several beamlines.

ACKNOWLEDGMENT

The authors wish to acknowledge the entire team at MAX IV, including controls and software, IT, accellerator group and operators for invaluable feedback.

REFERENCES

[1] The TANGO Control system website: `http://www.tango-controls.org`

[2] F. Poncet, J.L. Pons, 10th ICALEPCS Int. Conf. on Accelerator, & Large Expt. Physics Control Systems. Geneva, 10 - 14 Oct 2005, FR2.1-6O (2005).

[3] W3C, *Scalable Vector Graphics (SVG) 1.1 (Second Edition)*, `http://www.w3.org/TR/SVG/`

[4] Inkscape website: `https://inkscape.org/en/`

[5] Ecma International, *ECMAScript 2015 Language Specification*, `http://www.ecma-international.org/publications/standards/Ecma-262.htm`

[6] D3.js website: `http://d3js.org/`

[7] Python Software Foundation. *Python Language Reference, version 2.7*, Available at `http://www.python.org`

[8] PyTango website: `http://www.esrf.eu/computing/cs/tango/tango_doc/kernel_doc/pytango/latest/index.html`

[9] Qt website: `http://www.qt.io`

ISBN 978-3-95450-148-9

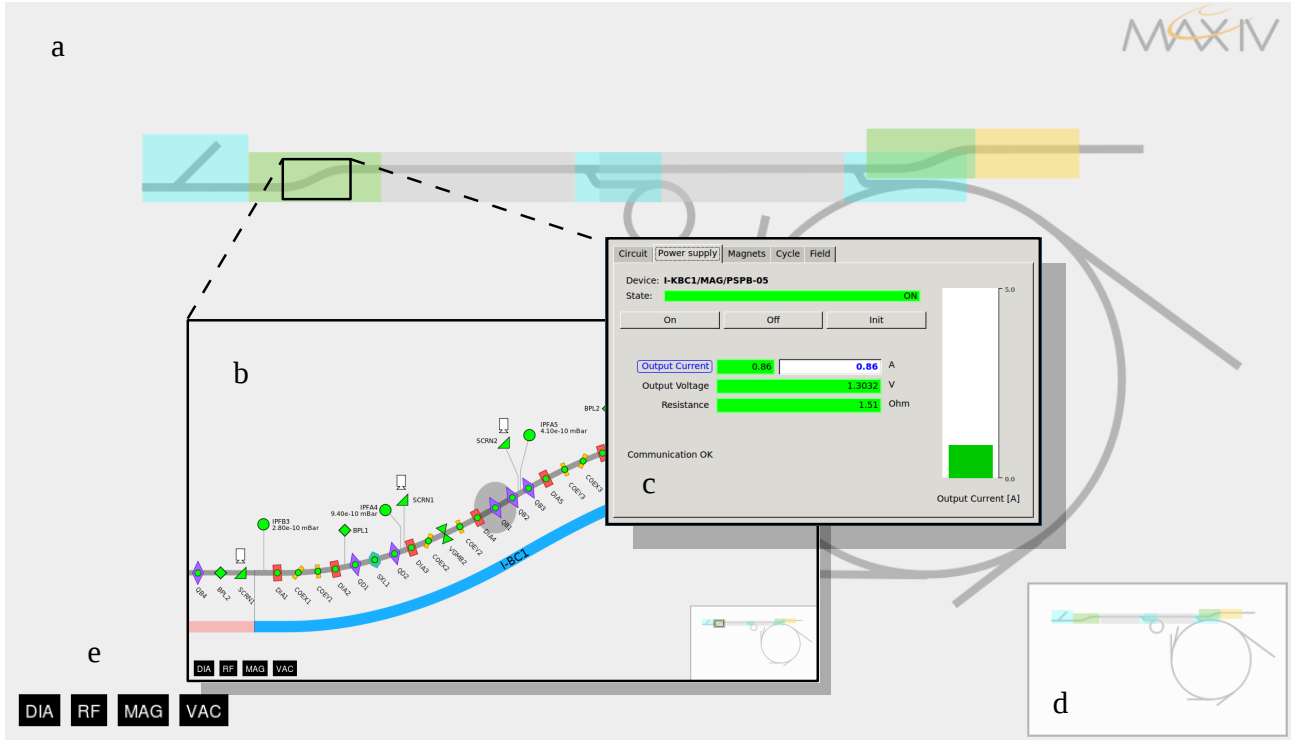

Figure 2: Synoptic for the linear accellerator of MAX IV. Initially an overview of the entire machine is shown (a). The user can zoom in on a particular region (b) to reveal individual equipment such as magnets. To access a particular device, a specific GUI (c) can be opened. A "thumbnail" (d) shows the current view in relation to the whole. A row of buttons (e) allow the user to control the visibility of individual subsystems.

Figure 3: An example of embedding the SVG synoptic widget in a dedicated vacuum monitoring application. The widget responds to user actions such as selecting a device in the tree to the left, by displaying the chosen device, and vice versa.

ISBN 978-3-95450-148-9